2021

良好的计算机科学考试之路

Prof Marco Gottardo Ph.D.

SIIC Strada per un buon esame di informatica. 1 e 2 Liceo 2020/2021.

prof.Marco Gottardo
SIIC –G-Tronic
15/04/2021

“SIIC Strada per un buon esame di informatica”
1 e 2 Liceo 2020/2021

Questa edizione è stata edita e pubblicata a Aprile 2021 da Marco Gottardo.∴.

Marco Gottardo, C.F. GTTMRC68R06G224I,

Via Colombo 14, 30030 Vigonovo (VE) Italia.

E-mail: ad.noctis@gmail.com

ISBN-13: 9798741106747

ISBN-10:

B J

Edizioni Gottardo 2021

良好的计算机科学考试之路

Prof Marco Gottardo Ph.D.

SIIC Strada per un buon esame di informatica

1 e 2 Liceo 2020/2021

OBIETTIVI SPECIFICI DI APPRENDIMENTO DELL'INFORMATICA

- Sviluppo delle capacità intuitive e logiche;
- Sviluppo delle attitudini sia analitiche che sintetiche
- Acquisizione della capacità di deduzione e di analisi
- Acquisizione del rigore espositivo e del corretto uso dei termini informatici
- Apprendimento dei componenti del sistema di elaborazione
- Apprendimento dei processi di astrazione e di formazione dei concetti
- Saper utilizzare strumenti informatici per rappresentare dati e oggetti relativi a diversi
- ambiti disciplinari
- Conoscere strategie algoritmiche per risolvere problemi di diverso genere

COMPETENZE IN USCITA DAL PRIMO BIENNIO
COMPETENZE LINGUISTICHE:

- ➢ acquisire il concetto di pseudolinguaggio
- ➢ acquisire il concetto di algoritmo e le sue rappresentazioni
- ➢ acquisire il concetto di linguaggio di programmazione
- ➢ saper passare da un "testo" di un problema a un programma
- ➢ saper utilizzare un linguaggio rigoroso nell'esposizione sia scritta che orale
- ➢ saper interpretare e costruire un grafico

COMPETENZE OPERATIVE:

- ➢ saper utilizzare consapevolmente le funzioni dei pacchetti applicativi di Office
- ➢ saper individuare consapevolmente le diverse fasi di realizzazione di un programma
- ➢ saper utilizzare la tecnica top-down per descrivere gli algoritmi
- ➢ saper utilizzare un ambiente di sviluppo di programmi
- ➢ saper utilizzare strumenti informatici di diverso tipo

COMPETENZE PROCEDURALI:

- ➢ saper risolvere problemi nei vari ambiti disciplinari
- ➢ aver appreso la tecnica dell'analisi dei problemi fino ad arrivare alla stesura dei programmi
- ➢ saper risolvere problemi utilizzando strumenti informatici
- ➢ saper risolvere problemi utilizzando uno specifico linguaggio di programmazione

INFORMATICA CLASSE PRIMA OPZIONE SCIENZE APPLICATE

L'INFORMATICA E IL PENSIERO ALGORITMICO:

Analizzare un problema e trovare la strategia risolutiva *:*

Che cos'è l'informatica:

L'informatica e la scienza che si occupa della rappresentazione di un problema in forma computabile da una macchina usando procedimenti iterativi di calcolo chiamati algoritmi.

L'informatica si divide in sviluppo di software e sviluppo di hardware capace di creare un'immagine memorizzabile del problema detto modello.

L'informatica non risolve lo specifico problema ma classi del problema.

Questo significa che il computer non è progettato per risolvere il problema 1+1=2 ma bensì per risolvere A+B=C ovvero tutte le somme e non la specifica somma.

Affinché questo sia possibile è stato concepito il concetto di tipo di dato astratto, che assegna un nome un indirizzo e un formato alle locazioni di memoria, rimangono così definite le variabili.

L'informatica si basa sull'utilizzo di machine calcolatrici che negli anni hanno avuto un notevole sviluppo.

I primi procedimenti di calcolo si basavano sul conteggio delle dita delle mani, e di li a poco all'associazione del concetto di gruppi di 10 sassolini da abbinare concettualmente agli oggetti da quantificare.

Di li a poco, già 3000 anni or sono, in Cina, si riunirono questi sassolini in aste scorrevoli costruendo i primi abachi.

L'abaco è senza ombra di dubbio il primo strumento per il calcolo costruito dall'uomo dotato anche di effetto memoria rappresentato dalla posizione in cui gli elementi venivano spostati.

Solo nel quattordicesimo secolo si diffuse l'uso delle cifre arabe che grazie al concetto dei pesi e delle posizioni delle cifre renderanno possibile la costruzione di un'algebra universale applicabile anche a sistemi di numerazione con base diversa da 10, ovvero il predetto numero delle dita della mano.

Nel 1617 il matematico scozzese John Napier, detto Nepero, inventò un dispositivo di calcolo basato sui principi dei logaritmi.

Contemporaneamente un inglese, Henry Briggs, pubblico le tavole logaritmiche e a breve comparve il regolo calcolatore.

Passarono circa vent'anni e il francese Blaise Pascal costruì la prima macchina calcolatrice meccanica in grado di fare le somme e le differenze.

Il tedesco Gottfried Leibniz nel 1673 costruì una macchina in grado anche di moltiplicare e dividere.

Nei successivi due secoli non ci furono novità nel campo dell'informatica.

Charles Babbage, nel 1822, progetto la macchina automatica per il calcolo detta alle differenze, che calcolava i polinomi. Da questa derivò il suo progetto della macchina "analitica", che mostrava alcune delle caratteristiche essenziali di un moderno computer.

Venne introdotta l'idea delle schede perforate in cui memorizzare lo stato di segnali pari a presenti o mancanti (true e false), già studiate dallo scienziato Jacquard.

Le macchine di Jacquad vennero utilizzate per il controllo delle macchine per la tessitura molto in uso negli anni della rivoluzione industriale.

Nella macchina analitica era già presente una memoria e un'unità centrale di calcolo e la funzione di stampa dei risultati.

Il progetto di Babbage rimase però prototipale o addirittura sulla carta perché nessuno finanziò la costruzione.

Per questa macchina, la signorina Lady Ada, figlia del noto Lord Byron, contessa di Lovelace, scrisse molti programmi, e si può considerare la prima programmatrice di computer "moderni" della storia.

Esiste il linguaggio di programmazione Ada il cui nome è stato assegnato in suo onore.

Negli stati uniti, alla fine del diciannovesimo secolo, serviva una macchina in grado di accelerare le operazioni di spoglio delle cartelle elettorali a cui ci pensò lo scienziato Herman Hollerith basandosi sulle idee della macchina di Babbage.

Venne applicata nelle operazioni di censimento americano del 1890.

Da questa data in poi vennero sempre più impegnate le macchine a schede perforate.

Il punto più debole di queste macchine era l'impossibilità di rielaborazione dello stesso dato, in caso di errore o necessità di ricalcolo tutto il pacco di schede, detto batch o lotto, doveva venire riperforato.

Arriviamo al 1930, con l'avvento dell'elettronica quando l'americano Howard Hathaway Aiken cominciò la costruzione del primo calcolatore elettromeccanico il MARK 1, che funzionava con relè e la cui costruzione terminò nel 1943.

Il programma era memorizzato su un nastro perforato e la memoria dati, costituita da ruote poteva memorizzare fino a 72 numeri di ben 23 cifre.

Per eseguire una addizione bastavano 3 decimi di secondo ma le moltiplicazioni e le divisioni richiedevano tra i 4 e i 10 secondi.

L'università di Harvard mantenne in uso questo calcolatore per 15 anni.

Il primo calcolatore elettronico fu l'ENIAC costruito nel 1946 con 18 mila valvole termo ioniche, la cui potenza di calcolo, paragonata a quella del predecessore Mark I era di 1 ora contro una settimana di elaborazione.

La memoria era minore che passava da 72 numeri a 20 numeri ma i tempi di elaborazione e calcola scendevano a circa 200 milionesimi di secondo per le somme e 3000 fino a 6000 milionesimi di secondo per le moltiplicazioni e le divisioni.

Le macchine moderne hanno però un'architettura che rispecchia gli studi del tedesco naturalizzato americano Johon von Newmann abbinate alle idee del matematico inglese Alan Turing, nella seconda metà degli anni 40.

Le università di Cambridge e di Pennsylvania lavoravano in maniera congiunta su progetti che dovevano portare allo sviluppo di macchine basate sui lavori di Von Newmann con programma residente in memoria.

Nel 1949 furono completati in Inghilterra il calcolatore EDSAC e in America il calcolatore EDVAC entrambi in grado di eseguire un programma residente in memoria.

Con ciò si intende che tutte le istruzione che il computer doveva eseguire sui dati erano memorizzate in un'area dedica della memori.

Questi computer, grazie a questa soluzione, potevano operare elaborando il programma senza l'intervento dell'operatore.

Il 6 maggio 1949 il computer EDSAC produsse una tavola dei quadrati dei numeri da 0 a 99, e questo storico tabulato è oggi esposto al museo della tecnica di Londra.

I primi computer commerciali, intesi per essere utilizzati dalle grandi compagnie comparvero negli anni 50 con il modello UNIVAC prodotto dalla Reminghton –Rand Company e il IBM 701 seguito dell'IBM 1401 e da IBM 1620.

L'azienda IBM (International Bussines Machine) nel 1950 divenne leader mondiale nel settore delle macchine automatiche per il calcolo.

Nel 1947 venne inventato il transistor, un componente elettronico allo stato solido destinano a sostituire le ingombranti e onerose di energia valvole termoioniche.

Il concetto di funzionamento era molto simile ma basava il suo funzionamento sulle proprietà chimeriche dei materiali semiconduttori, il silicio e il germanio, piuttosto che sull'accelerazione degli elettroni emessi per effetto termoionico.

Il transistor risultava subito pronto ad eseguire il calcolo mentre la valvola doveva prima arrivare alla temperatura di emissione ionica.

Il transistor poteva essere costruito centinaia di volte più piccolo della stessa valvola.

Il transistor, parola che deriva dall'unione delle parole cambio di resistenza trans-resistors era anche termicamente piu stabile e più facilmente trasportabile.

Le potenzialità di memoria aumentarono notevolmente con l'invenzione dei nastri magnetici.

La precedente scheda perforata può contenere circa 28 caratteri per decimetro mentre un nastro magnetico arrivava oltre 2000.

Già negli anni 70 entrarono in uso massiccio i circuiti integrati che eliminavano tutte i fili di interconnessione creando un intero circuito logico all'interno dello stesso componente.

Grazie ai circuiti integrati si ridusse notevolmente il parametro MTBF (tempo medio tra due guasti) aumentando notevolmente l'efficienza e l'affidabilità dei computer.

La codifica simbolica rappresento la svolta nel progresso del software, in cui alcuni simboli vennero abbinati a specifiche funzioni e operazioni della macchina.

Dato e informazione

La rappresentazione dei dati all'interno del computer sta nella possibilità di gestire locazioni di memoria formattate (significa con la misura e distribuzione dei bit preassegnate), all'interno sia della memoria di massa (i dischi rigidi SSD, oppure i rimovibili USB, oppure la RAM o ROM).

I vari formati sono stati assegnati da specifici comitati scientifici allo scopo di rappresentare dati e numeri.

Ad esempio, il numero P-greco, spesso indicato con Pi oppure il simbolo π, con valore 3.14 viene espresso con 32 bits detti double word in una specifica locazione di memoria generalmente con origine ad un indirizzo pari.

Le suddivisioni di memoria secondo gli standard sono:

1. Il singolo bit, detto unità fondamentale o boolean.
2. Il carattere della tabella ASCII, che rappresenta con 8 bit oggetti stampabili e non stampabili (ad esempio il rumore bit che può generare il computer in fase di errore o segnalazione di un evento), detto BYTE, che può anche contenere un short INT, ovvero un numero espresso in binario puro compreso tra 0 e 255 ovvero 2^8=256 a cui si sottrae 1 per la necessità di esprimere lo zero.
3. La word, pari a 2 byte, di estensione 16 bit. Questa può contenere un numero intero compreso tra -32768 e più 32767, la cui asimmetria è dovuta alla necessità di rappresentare lo zero togliendo un valore da quelli disponibili sul lato positivo.
4. La double word, è una locazione di memoria a 32bits corrispondente a 4 byte ovvero 2 word. In una double word possiamo allocare (allocare significa memorizzare o fissare in un indirizzo di memoria) una variabile di tipo DINT (doppio intero, molto più grande di 32767) oppure un numero REAL, che in tanti linguaggi viene chiamato float. Questo nome deriva dal fatto che la sua rappresentazione avviene spostando la virgola a destra o a sinistra del valore originale dando luogo a quello che si chiama il processo di normalizzazione. I numeri REAL detti FLOAT sono normati dalla regola IEEE

secondo cui il primo bit a destra è il segno, 0 se positivo 1 se negativo, il successivo byte corrisponde alla rappresentazione del valore dell'esponente per il quale abbiamo elevato la base 10 per poter spostare la virgola avanti o indietro allo scopo di normalizzare la mantissa. La mantissa è l'insieme dei rimanenti 23 bits e rappresenta il numero che si ottiene spostando la virgola all'interno del numero fino a che si ottiene 0,valore ad esempio π quando normalizzato diventa 0,314 ma per tornare al valore originale deve essere moltiplicato per 10^1 quindi π = 3,14 = 0,314 * 10^1

5. I dati possono però essere anche strutturati, ovvero composti da più tipi di dato predefinito, detto anche tipo di dato astratto perché viene adattato, secondo le regole dei punti 1,2,3,4 alla modalità di memorizzazione sul calcolatore. Il tipo di dato composto si chiama struct o a volte record. La parola chiave struct apre una area di memoria in cui creare una variabile di un nuovo tipo di dati creato vincolando i tipi di dato predefiniti a stare assieme, anche se di tipo diverso, all'interno di un unico nome identificativo. Ad esempio, la variabile anagrafica, rappresenta una persona all'interno della memoria del computer mettendo assieme nel nuovo tipo persona, i campi età=int, nome = string[], cognome = string [], altezza = float, peso = int; //ecc.

Ecco un esempio di struttura dati che rappresenta una persona all'interno della memoria del computer.

```
#include <iostream>
#include <conio.h>

using namespace std;  //abilita in flusso verso il monitor o dalla tastiera

struct dati_persona{
        char nome[25];
        char cognome[25];
        char gender;
        int anno;
        int mese;
        int giorno;
        char citta[25];
};

void leggi_persona(){
        dati_persona persona;
        cout<<"Inserisci dati anagrafici della persona\n";
        cout<<"Nome -> ";
        cin.getline(persona.nome,25);
        cout<<"Cognome -> ";
        cin.getline(persona.cognome,25);
        cout<<"gender (m/f) ) -> ";
        cin>>persona.gender;
        cout<<"inserire la data di nascita -> ";
        cin>>persona.anno;
        cin>>persona.mese;
        cin>>persona.giorno;
```

```
	cin.ignore();
	cout<<"Luogo di nascita -> ";
	cin.getline(persona.citta,25);

	// per leggere i dati se sono locali
	system("cls");
	cout<<"Estrazione la scheda della persona\n";
	cout<<persona.nome<<" ";
	cout<<persona.cognome<<" ";
	cout<<"sesso "<<persona.gender<<"\n";
	cout<<"data di nascita ("<<persona.giorno<<"/"<<persona.mese<<"/"<<persona.anno<<")\n";
	cout<<"luogo di nascita ("<<persona.citta<<")";
	getch();
}

void splash(){
	cout<<"**********************************\n";
	cout<<"*    prof. Marco Gottardo PhD             *\n";
	cout<<"* seconda classe liceo 2021               *\n";
	cout<<"*Definizione struttura anagrafica        *\n";
	cout<<"**       15/03/2021                      **\n";
	cout<<"**********************************\n"<<endl;
	cout<<"\n Press any key \n";
	getch();  //il prototipo è in conio.h
}

void	byebye(){
	system("cls");
	cout<<"**********************************\n";
	cout<<"*programma eseguito correttamente     *\n";
	cout<<"*          Bye Bye !!!                 *\n";
	cout<<"*       struct anagrafica              **\n";
	cout<<"**         15/03/2021                   **\n";
	cout<<"**********************************\n"<<endl;
	cout<<"\n Press any key \n";
	getch();  //il prototipo è in conio.h
	}

int main(int argc, char** argv) {
	splash();
	leggi_persona();
	byebye();
	return 0;
}
```

Ragionare per algoritmi

Un algoritmo è un procedimento iterativo pensato per risolvere uno specifico problema. Consiste in una sequenza che deve essere limitata di istruzioni, non ambigue, che possono essere eseguite sia da computer che macchine che persone, portando alla soluzione del problema in un numero finito di passi.

Si può anche definire più semplicemente un algoritmo come un "metodo di elaborazione da applicare a certi dati iniziali per ottenere dei dati finali o risultati".

Per ottenere dei risultati il problema deve avere un insieme di dati iniziali ben definito e deve avere una formulazione tale in modo che l'obbiettivo finale, ovvero l'uscita dai calcoli iterativi non abbiano nessuna ambiguità ovvero che i valori finali da raggiungere siano chiari.

A questo fine si devono avere:

- Un chiaro l'obiettivo da raggiungere
- Un set di dati di partenza noti e sufficienti
- il problema non sia impossibile.

Se cade una di queste condizioni l'algoritmo non è formalizzabile.

Non si deve comunque mai confondere l'output dell'esecuzione di un algoritmo (il risultato) con quanto si voleva ottenere dall'applicazione dell'algoritmo stesso.

Un algoritmo deve allora avere le seguenti caratteristiche:

- Generalità
- Finitezza
- Realizzabilità;
- Completezza;
- Riproducibilità;
- Non ambiguità (o precisione).

Per **Generalità** si intende il fatto che l'algoritmo deve risolvere la classe dei problemi e non lo specifico problema nel senso che non è fatto per eseguire la specifica somma 1+1=2 ma bensì tutte le somme a+b=c

Per **Finitezza** si intende il fatto che l'algoritmo sia composto da un numero finito di istruzioni e termini in un numero finito di passi (per istruzione si intende la descrizione di una certa operazione, per passo si intende l'esecuzione di quella istruzione).

Per **Realizzabilità** si intende il fatto che l'algoritmo sia realizzabile (e quindi prima di tutto comprensibile) da parte di chi lo deve eseguire.
questo comporta anche che l'algoritmo deve essere molto dettagliato cioè composto di istruzioni molto elementari (cioè non ulteriormente scomponibili). Il grado di dettaglio che si deve raggiungere nella stesura di un algoritmo non è definibile con precisione ma deve essere comunque tale da assicurare la realizzabilità delle singole istruzioni da parte di un computer.

Completezza significa che devono essere previste tutte le possibilità che possono verificarsi durante l'esecuzione e per ognuna devono essere definite le azioni da svolgere.

Riproducibilità significa che successive esecuzioni dell'algoritmo con gli stessi dati di partenza devono condurre agli stessi risultati.

La **Non ambiguità** è molto importante perché il computer non è in grado di ragionare su cosa gli viene chiesto di fare, lo esegue e basta.

Gli algoritmi non possono essere ambigui e non possono contenere istruzioni vaghe; quindi non potranno essere descritti con il linguaggio naturale (la lingua parlata) perché si presta spesso a diverse interpretazioni.

Si deve quindi ricorrere ad altri strumenti per descrivere algoritmi che dovranno essere eseguiti da macchine non intelligenti; un esempio è costituito dai linguaggi macchina, che oggi vengono creati dalla compilazione dei linguaggi ad alto livello, ad esempio il C++, già più vicino a un linguaggio naturale.

Qualunque sia il linguaggio con cui si descrive un algoritmo, le istruzioni che lo compongono devono essere eseguite una alla volta e nell'ordine in cui sono scritte.

Le proprietà dei buoni algoritmi

Un algoritmo, inteso come procedimento iterativo di calcolo, risulta migliore o peggiore di un altro algoritmo, avente lo stesso scopo finale, in funzione di quanti passi richiede prima di produrre un output.

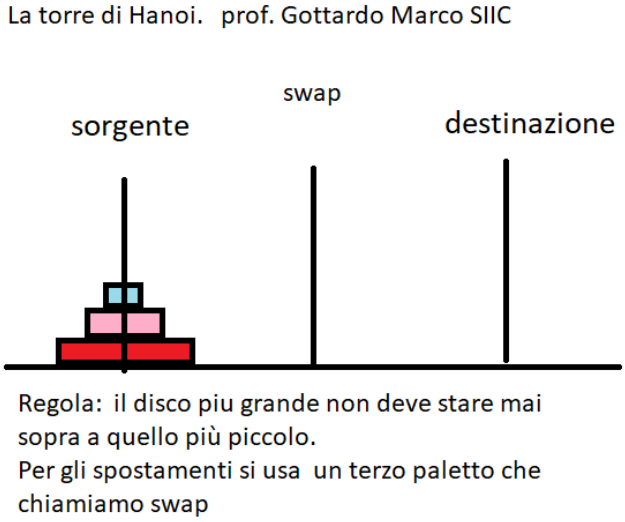
In generale, è la dimensione dei dati in input che determina la necessità di eseguire più passi.

Questo parametro è definito come "dimensione dell'input" e la quantità di passi è chiamata "complessità computazionale".

Gli algoritmi peggiori hanno una complessità esponenziale, o peggio ancora fattoriale, per cui all'aumentare della dimensione dell'input si esegue un numero di passi che cresce con la potenza.

Questi algoritmi, benché eseguiti su potenti e odierni calcolatori potrebbero non giungere mai a produrre un output finale perché il tempo necessario è maggiore di quello della vita dell'universo, quindi sicuramente maggiore di quello della vita del programmatore che lo ha messo in esecuzione.

Ne è un esempio il gioco della torre di Hanoi.



Il problema della torre di Hanoi ha complessità $O(2^n)$, esponenziale, e qualunque altro algoritmo risolutivo non potrà avere complessità minore... (è un problema intrinsecamente esponenziale).

Se nella torre vi sono inizialmente 64 dischi da ordinare, il tempo machina necessario, in funzione delle mosse da eseguire per ordinare i dischi in modo decrescente è pari a 2^{64}-1 = 18.446.744.073.709.551.615 per una durata di circa $\sim 5*10^{11}$ anni se un uomo esegue una mossa al secondo quindi per un computer che elabora a 1GHz il tempo è ~500 anni.

Ne consegue che è un pessimo algoritmo sfruttabile solo per bassi valori dell'input.

Co soli 10 dischi, se non sbagliassimo mai la mossa, un uomo che esegue un movimento al secondo impiegherebbe 17 secondi mentre il computer circa 1u secondo.

Ma se i dischi salgono a 30 allora il tempo necessario sale a 34 anni.

Segue la soluzione per 2 dischi e per tre dischi.

Input : 2

Output : Disk 1 moved from A to B

Disk 2 moved from A to C

Disk 1 moved from B to C

Input : 3

Output : Disk 1 moved from A to C

Disk 2 moved from A to B

Disk 1 moved from C to B

Disk 3 moved from A to C

Disk 1 moved from B to A

Disk 2 moved from B to C

Disk 1 moved from A to C

```
// C++ recursive function to
// solve tower of hanoi puzzle

#include <bits/stdc++.h>
using namespace std;

void towerOfHanoi(int n, char from_rod,
                  char to_rod, char aux_rod)
{
    if (n == 1)
    {
        cout << "Move disk 1 from rod " << from_rod <<
                        " to rod " << to_rod<<endl;
```

```
    return;
  }
  towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
  cout << "Move disk " << n << " from rod " << from_rod <<
                  " to rod " << to_rod << endl;
  towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

// Driver code
int main()
{
  int n = 4; // Number of disks
  towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of rods
  return 0;
}
```

in conclusione, ci sono algoritmi che hanno complessità computazionale esponenziale (non buoni), algoritmi che hanno complessità lineare (a volte accettabili), e algoritmi che hanno complessità logaritmica, per i quali anche alla crescita dell'input si ha una compressione della crescita in output, questi sono i migliori.

L'informatica e le altre scienze

L'informatica può essere considerata come una scienza ausiliaria in riferimento ad altre discipline scientifiche, in modo particolare per quanto concerne la fisica, la medicina, le discipline finanziarie ed in genere tutte quelle discipline dove si esige l'utilizzo della logica applicata.

Le scienze informatiche di avanguardia riguardano l'utilizzo delle procedure di intelligenza artificiale, quali machine- learning e/o deep-learning, che utilizzano neural networks, per applicazioni computeristiche standard, ma si spingono all'utilizzo di elementi quantistici per applicazioni ancora più complesse, che non fanno più riferimento al concetto di bit, in quanto tale, ma si riferiscono al "quantum-bit", come unita elementare di elaborazione dei dati.

LA CODIFICA DIGITALE DEI DATI

Come si rappresentano i dati all'interno del computer

Il computer è in grado di leggere solo lo stato alto o basso (TRUE o FALSE) dell'unità fondamentale di memorie detta bit.

Si rende necessario correlare tra loro i bit al fine di rappresentare i dati e le informazioni.

Le informazioni possono essere di tipo numerico ma anche di tipo letterale quindi caratteri, quando questi sono rappresentati insieme si dicono alfanumerici.

I dati assumo significato quando questi sono composti da bit che rispettano certe regole.

Le prime più basilari regole sono il raggruppamento dei bit.

Il singolo bit è usato per le decisioni di deviazione del flusso dell'elaborazione del programma, ad esempio, se un numero è maggiore di un altro eseguo un task altrimenti ne eseguo un altro.

- Bit -> on or off
- Nibble -> 4 bit, servono per rappresentare il codice binario BCD ovvero quello che esprime i numeri da 0 a 9 per poter essere inviati a display a led. Quando il numero è maggiore di 9 si fa un riporto alla cifra successiva, ad esempio 9+1 = 0 con riporto di 1 nel successivo digit.
- Byte -> il byte è l'unità di riferimento per la memoria. La quale si esprime in potenze di 2 ma riferite al byte, ad esempio il kbyte è una quantità pari a **1024 byte** e non 1000 perché 1000 non è rappresentabile come potenza di due. il singolo byte può rappresentare sia un carattere, ad esempio quelli che arrivano dalla tastiera, oppure un numero intero corto, compreso tra 0 e 255. Infatti il numero 255 è espresso con 8 bit tutti a 1.
- Word -> composto da due byte. Sono 16 bit, in questo spazio sono di solito contenuti gli interi naturali, ovvero qui numeri che non hanno la virgola. Spostando lo 0 al centro del range rappresentabile si potranno esprimere i numeri interi dal valore -32768 fino al numero +32767. Sul lato positivo ce n'è uno di meno perché bisogna rappresentare anche lo zero.
- Double Word -> composta da 2 word, ovvero 4 byte, quindi sono 32 bit. Con questa estensione possiamo rappresentare i numeri Reali, ovvero quelli con la virgola che in C e C++ si chiamano float.

All'interno delle locazioni di memoria, con l'estensione mostrata sopra, le informazioni, dette i dati, possono essere rappresentate in varie maniere.

Quelle più comuni sono il formato IEEE reale, ovvero quello secondo cui il numero è diviso in parti, che sono, il bit più a sinistra rappresenta il segno, i successivi 8 l'esponente del valore normalizzato prima della conversione, ovvero una potenza di 10 equivalente a quante posizioni abbiamo spostato la virgola per ottenere la prima cifra a destra del punto decimale diversa da zero, ad esempio se dovessimo scrivere nella memoria del computer il numero Pi-greco, dovremmo prima spostare la virgola di una posizione indicando 0,314 invece che 3,14. Ma i due numeri sono uguali se eseguo la moltiplicazione per 10 elevato alla 1. Ne consegue che 0,314 è uguale a 3,14 se faccio il calcolo 0,314*10.

Ne consegue che i secondi 8 bit dopo il segno dovranno esprimere il numero 1, ma siccome esistono le potenze anche negative si preferisce sommare a questo numero 127 così che il numero 1 vale in realtà 128.

128_{10}=1000 0000_2 questa stringa di 8 bit rappresenta in accesso 127 il numero 1 per esprimere la potenza del numero normalizzato.

Quanto rimane a destra della virgola del numero 0,314 è 314 e si chiama **mantissa normalizzata**

La mantissa normalizzata viene espressa in binario puro secondo la conversione diretta.

314_{10}= 00011111010100111010_2

π = 1 **1000 0000** 00000011111010100111010

Dove, per esprimere 3,14 da sinistra verso destra

- 1= segno
- 1000 0000 = esponente della mantissa normalizza
- 00000011111010100111010 = mantissa normalizzata.

Solo in pochi casi il numero potrà essere espresso in binario puro, oppure in altre forme di conversione.

Ma i numeri interi potranno essere rappresentati dividendo a metà il numero 65296 e ponendo lo zero al centro, in effetti, sommando il valore negativo -32648 a quello espresso.

Quando vogliamo rappresentare i caratteri, ad esempio quelli della tastiera o delle stampanti, ci atteniamo a una tabella le cui celle sono invidiate da 4 bit per riga e 4 bit per colonna totale 8 bit.

ASCII - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange è un codice standard a 7-bit che fu proposto dall'ANSI nel 1963 e diventò definitivo nel 1968.

ASCII (si pronuncia "askii") è il codice standard per i microcomputer e consiste di 128 numeri decimali che vanno da 0 a 127.

I numeri che vanno da 128 a 255 costituiscono il set di caratteri estesi che comprendono caratteri speciali, matematici, grafici e di lingue straniere.

Queste due tabelle si chiamano ASCII American standard code for information interchange e sono adottate da tutti i costruttori mondiali di computer o periferiche per computer.

È possibile convertire il numero da binario a decimale per ottenere la casella corrispondente del codice, ad esempio, la lettera a (minuscolo) è il numero decimale 97.

TABELLA COMPLETA

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Convertire un numero da un sistema di numerazione ad un altro

Definiamo innanzitutto il numero.

Un numero è una somma ripetuta di prodotti i cui fattori sono la cifra e la base elevata alla posizione.

Questo vale per tutte le basi.

Ad esempio quando scrivo 123 devo specificare la base altrimenti non ha significato.

- Rappresentare i diversi tipi di dati
- Il sistema binario, bit e byte
- Il sistema di numerazione esadecimale.
- La rappresentazione dei numeri

Conversione da binario a decimale e viceversa

La conversione numerica dalla base 2 alla base 10 è abbastanza semplice perché le cifre sono solo 2 e i pesi facili da ricordare. In effetti si tratta di sommare le potenze di 2 per i bit posti a 1 logico nel numero binario.

Partendo dal più leggero questo algoritmo prevede la somma dei valori 1,2,4,8,16,32,64 ecc, tenendo presente che qualsiasi numero elevato alla 0 vale 1.

```
#include <iostream>
#include <math.h>
#include <conio.h>
#define B1  1 // 2 alla 0
#define B2  2 // 2 alla 1
#define B3  4 // 2 alla 2
using namespace std;

int calc(){

        int cifreB[3];
        int cifreC[3];
        int indexC;
        int numF;
        system ("cls");
        cout << "Inserire 3 cifre in base binaria" << endl;
        cin >> cifreB[1] >>cifreB[2]>>cifreB[3];
        cifreC [1] = cifreB[1] * B3;
        cifreC [2] = cifreB[2] * B2;
        cifreC [3] = cifreB[3] * B1;
        numF = cifreC[1]+cifreC[2]+cifreC[3];
        cout << "\n\nLa conversione ha dato il risultato di ->\t" << numF << endl;
        cout << "\n\nPremere 'q' +ENTER per terminare il programma" << endl;
        cout << "Premere ogni altro tasto + ENTER per continuare" << endl;
        return 0;
}

void hello (){
```

```
	cout << "Maxim Shapiro" << endl;
	cout << "3a media" << endl;
	cout << "Conversione binario-decimale" << endl;
	cout << "\nPremere ogni tasto + ENTER, tranne che 'q', per continuare" << endl;
}

void byebye (){
	system ("cls");
	cout << "\n\nProgramma eseguito!" << endl;
	cout << "3a media" << endl;
	cout << "Conversione binario-decimale" << endl;
	cout << "Premere qualsiasi tasto per terminare il programma" << endl;
}

int main (){
	char forward;
	hello();
	cin >> forward;
	while (forward != 'q'){
		calc();
		cin >> forward;
		if (forward == 'q'){
			byebye();
			getch();
			break;
		}
	}
	return 0;
}
```

Conversione da decimale a esadecimale e viceversa

I simboli numerici decimali sono anche detti "cifre arabe", che a loro volta sono una evoluzione dei numeri brahmi indiani (300 aC) che si evolvettero con le codifiche indiane dei numeri gwallor (500dC) e costituisco la base del sistema numerico decimale: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

I numeri in base esa-decimale, sono una conseguenza delle codificazioni binare (codice binario) e (ottale), ovvero composto da otto bit, quindi con la codifica esadecimale, sono necessari sedici simboli, tutti quelli che compongono la codificazione decimale più le prime 6 lettere dell'alfabeto.

Base esadecimale: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Come si puo notare la cifra "A" esadecimale equivale al "10 decimale" , la "B" equivale a "11" la "C" al "12", la "D" al "13", la "E", al "14" ed in fine la "F", al "15" in valore decimale.

Da quanto sopra, di conseguenza il valore “10” in base esadecimale equivale al valore “16”, in base decimale.

Quindi, per analogia, se una cifra qualsiasi scritta in un valore decimale assumerà per esempio, la seguente struttura:

2021, significa eseguire una somma con la seguente caratteristica: $2*10^3 + 0*10^2+2*10^1+1*10^0 = 2021$

La stessa cifra scritta pero in valori esadecimali sarà cosi scomponibile (conversione esadecimale-decimale):

$(2021)_{16} = 2*16^3+0*16^2+2*16^1+1*16^0 = 8.192 + 0 + 32 + 1 = (8225)_{10}$

Altresì, per effettuare una conversione da decimale a esadecimale, con cifre superiori a “15”, bisogna:

dividere il numero decimale per “16”, calcolando il quoziente ed il resto.

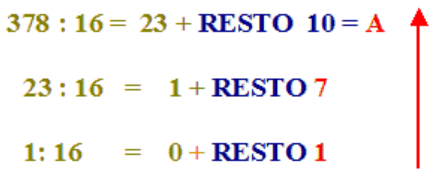
Se il quoziente della divisione e diverso da “0”, si deve continuare a dividerlo finche sarà minore di “16”

Scrivere i resti delle divisioni in ordine inverso rispetto a come sono stati ottenuti e separarli tra loro.

Convertire i numeri maggiori di “9”, nelle cifre alfabetiche.

La successione numerica cosi eseguita sarà la cifra corrispondente al valore esadecimale della cifra decimale in origine.

Esempio:



Quindi $(378)_{10}$ -> $(17A)_{16}$

Per la conversione inversa vale il medesimo metodo applicato per tutte le basi derivante dalla definizione di numero:

$$\sum_n C \cdot b^p = numero$$

Con C indichiamo la cifra, con b la base in cui il numero è rappresentato, con p il peso relativo alla posizione, ovvero a partire da zero la cifra più a destra e sommando 1 verso sinistra.

Per il valore convertito prima allora si ha:

$$numero = 1 \cdot 16^2 + 7 \cdot 16^1 + A \cdot 16^0 = 1 \cdot 256 + 7 \cdot 16 + 10 = 256 + 112 + 10 = 378_{10}$$

Conversione da esadecimale a binario e viceversa

I numeri con basi diverse da quella decimale sono di solito finalizzate all'utilizzo nelle funzioni informatiche nei computer tradizionali che si basano sull'algebra binaria.

E' interessante, prima di parlare di conversioni tra una base e l'altra, vedere come vengono rappresentati i numeri, in relazione alla base con la quale vengono esposti, nella fattispecie, solitamente si usano le seguenti rappresentazioni principali ponendo come esempio il numero $(255)_{10}$:

Questo numero intero ci sta in un byte, ovvero 8 bit, come mostrato nella lista qui sotto.

- 255 (in base decimale)
- 0377 (in base ottale) * *quasi mai usata*
- 0xff (in base esadecimale)
- 0b11111111 (in base binaria)

In particolare si noti che vengono utilizzati dei prefissi particolari come "0b", seguito dalle cifre binarie oppure "0x", per indicare che la cifra che segue ha base esadecimale, oppure "0o", per la base ottale.

E' anche importante notare che le cifre multiple della base dieci (in decimale), non vanno confuse con le cifre simili scritte con basi divere:

$(100)_{10}$ = cento …. $(1000)_{10}$ = mille …

$(100)_{16}$ = uno-zero-zero (in esadecimale) …. $(1000)_{16}$ = uno-zero-zero-zero (in esadecimale).

$(100)_{2}$ = uno-zero-zero (in binario) …. $(1000)_{2}$ = uno-zero-zero-zero (in binario)

Quindi, In generale per effettuare una conversione tra una base ed un'altra bisogna dividere il numero da convertire per la base b fino a quando l'ultimo quoziente è minore della base stessa (b), dopodiché il numero convertito si ottiene prendendo l'ultimo quoziente e tutti i resti delle divisioni, procedendo dall'ultimo resto al primo e scrivendoli da sinistra verso destra.

Quindi l'algoritmo di conversione è sempre lo stesso qualunque sia la base di partenza.

Per convertire un numero esadecimale in un numero binario, potrebbe essere molto pratico convertirlo dapprima in un numero decimale e poi procedere con la conversione a binario, è consigliati quindi fare due successive conversioni.

Osservazione:

La conversione dalla base 2 alla base 16 e/o 8, e viceversa, è più semplice e veloce di quella da decimale ad altre basi.

Infatti basta considerare che per rappresentare le sedici cifre diverse del codice esadecimale occorrono 4 bit (24 = 16) mentre per rappresentare le otto cifre diverse del codice ottale occorrono 3 bit (23 = 8).

La regola è che per convertire un numero binario in esadecimale o in ottale, si devono raggruppare le cifre binarie in gruppi di quattro o tre cifre (bit) a partire da quelle "meno significative": si ricava il numero sostituendo i bit così ricavati con la cifra esadecimale o ottale corrispondente.

Per effettuare delle conversioni rapide si può utilizzare questa tabella:

ESADECIMALE	DECIMALE	BINARIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Rappresentazione delle informazioni alfanumeriche

Le informazioni alfanumeriche sono quei simboli, che solitamente vengono immessi da una tastiera e che si possono riprodurre a video, ne fanno parte ovviamente tutti i caratteri e tutti i numeri, ma anche tutta una serie di simboli speciali, pensiamo ad esempio ai simboli matematici, che molto spesso utilizzano le lettere greche, oppure, i simboli utilizzato come operatori matematici, o ancora dei simboli speciali, che solitamente non vengono scritti a video, poiché utilizzati quali comandi o combinazioni di tasti particolari.

Per inciso, va detto che le rappresentazioni di qualsiasi tipo, su un supporto informatico, vengono scritte attraverso delle "codifiche binarie", ovvero una serie di "0" e "1", che identificano, per cosi dire, un numero al quale e stato associato un particolare "carattere".

Quindi, esistono delle convenzioni, di rappresentazione, come abbiamo ad esempio già visto per la scrittura dei numeri, nelle varie basi.

E' anche utile ribadire, che per ogni simbolo scrivibile, ci possono essere molte versioni, si pensi ad esempio proprio ai caratteri che state leggendo, che vengono presentati, a video, cosi come nel libro, utilizzando un determinato tipo di "layout" che fa riferimento ad una particolare grafia.

Quindi, le rappresentazioni "alfanumeriche" sono variegate, una prima grande divisione però viene fatta tra i caratteri in genere e le cifre (numeriche).

Assume allora una grande importanza il concetto di "variabile", che identifica il "contenuto" di un certo "contenitore", in un determinato frangente di tempo.

In particolare, potremmo dire che vi sono variabili adibite a contenere qualsiasi tipo di carattere letterale e variabili adibite invece a contenere soltanto i caratteri numerici, in una delle varie forme possibili di rappresentazione, come abbiamo accennato nei paragrafi inerenti le conversioni.

Con questo approccio, si distinguono allora le rappresentazioni di "stringhe" e di "numeri":

- "ambarabbaciccicocco" es una stringa
- "CCCRMO11ZXXXN" es una stringa
- **0x1ABCDEF** es un numero (in esadecimale)
- **"@" "#" "%" "&"** sono stringhe (ma anche comandi)

I caratteri ASCII e Unicode

La trasmissione dei dati tra computer e computer, oppure tra computer e periferica in maniera bidirezionale ha reso necessario normare e standardizzare insiemi di caratteri abbinati a valori numerici che ne siano le coordinate in una tabella

Già nel 1967 è stato formalizzato il codice ASCII (American Standard Code for Information Interchange), che ancora oggi costituisce la base per la rappresentazione in forma numerica dei caratteri che compongono qualunque testo.

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

Binary	Oct	Dec	Hex	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

I codici numerici compresi tra 0 e 31 non vengono riportati, in quanto rappresentano caratteri speciali che non visualizzabili o stampabili. un esempio significativo è dato dal codice numerico 10, un carattere speciale che indica il punto in cui una linea di testo rappresentata in binario come una lunga sequenza di “0” e “1” è interrotta per «andare a capo». (in alcuni linguaggi di programmazione ad esempio si usa una combinazione di simboli come questa: “/n”.

I programmi che “leggono” i dati numerici introdotti da un dispositivo di input, come ad esempio la tastiera, devono preventivamente convertire le sequenze in codici ASCII, che rappresentano i caratteri nel corrispondente formato ASCII numerico. Per poi “scrivere” un risultato numerico su un dispositivo di output che convertirà a sua volta questi codici ASCII assieme ai comandi occorrenti alla loro rappresentazione.

Si osservi che ad esempio, sommando o sottraendo 32 al codice ASCII di un carattere alfabetico, se ne ottiene rispettivamente la trasformazione da maiuscolo a minuscolo o, viceversa, da minuscolo a maiuscolo.

Come si può immaginare, dall’osservazione della tabella, il codice ASCII originale, disponeva di relativamente scarsa potenzialità informatica, avendo a disposizione solo 7 bit, fu per questo motivo che venne introdotto il termine ASCII esteso (in inglese extended ASCII o high ASCII) che designa una codifica a 8 bit o più, in grado di rappresentare molti altri caratteri oltre ai tradizionali 128 dell'ASCII a 7 bit, ma ancora non sufficientemente in grado di poter rappresentare tutti i simboli necessari allo sviluppo dell’informatica, anche per l’inclusione di simbologie diverse dai caratteri latini.

L’introduzione di una nuova codifica denominata “Unicode”, servì per estendere i caratteri verso gli asiatici e non latini.

Nel 1991 per poter codificare molti più caratteri, in modo standard, e permettere di utilizzare più set di caratteri estesi (es. greco e cirillico) in un unico documento entò in vigore l’Unicode.

Questo set di caratteri è oggi largamente diffuso e prevedeva inizialmente 65.536 caratteri (code points).

In seguito anche questo set di codici e stato esteso a 1.114.112 (= 220 + 216) e finora ne sono stati assegnati circa 101.000 code point.

I primi 256 code point ricalcano esattamente quelli dell'ISO 8859-1.

La maggior parte dei codici sono usati per codificare lingue come il cinese, il giapponese ed il coreano.

LA CODIFICA DEI DATI IN DIGITALE

Nei sistemi computerizzati, che nello stato attuale della tecnica si possono assimilare ai sistemi digitali, le informazioni possono essere di tipo immagine, di tipo sonoro oppure di tipo alfanumerico.

Spesso la loro rappresentazione, in modo che la macchina possa manipolare il dato corrisponde a una conversione da analogico a digitale.

Ciò che preleva il dato o l'informazione dal mondo fisico per crearne una sua immagine è solitamente detto trasduttore.

Ad esempio, un'immagine che colpisce la retina dei nostri occhi non è altro che un raggio di fotoni, particelle quantiche di natura ondulatoria, che colpisce i coni ricettori i quali li convertono in un insieme, assimilabile a una matrice di stimoli successivamente interpretati, per confronto con ciò che già ricordiamo, all'interno del nostro cervello, ovvero la memoria.

Un processo di questo tipo è da molti anni emulabile con sistemi digitali, quali sono i computer, dopo che l'immagine sia stata trasdotta (convertita) da insieme di frequenza analogiche a stimoli, presente o non presente sui coni dell'occhio (bits).

In ogni caso l'immagine, o ogni altro stimolo, ad esempio il suono che arriva come onda di pressione al nostro timpano, deve essere "digitalizzato" allo scopo di essere memorizzato e successivamente confrontato con qualcosa di noto al fine di essere riconosciuto.

La codifica delle immagini deve operare sull'informazione affinché i dati possano essere con facilità, archiviabili nella memoria RAM o di massa, Dovranno essere facilmente modificabili/elaborabili dal processore centrale, e devono essere facilmente gestibili per le comunicazioni con le periferiche tramite i protocolli standard, ad esempio la serializzazione RS232 o USB oppure inviati tramite sistemi di trasmissione in etere come il bluethoot o WiFi.

Le informazioni memorizzabili si possono riassumere nelle seguenti.

- Le immagini in formato digitale
- I suoni in formato digitale
- La compressione dei dati

Vediamo prima come viene convertita un'immagine facendo un semplice esempio per un carattere in bianco e nero.

Tecnica "**Bitmap**" per la lettera **A maiuscola** si verifica:

In una matrice di 6 righe per 5 colonne, che ne definisce la risoluzione, rappresentiamo in nero quelle occupate dalla lettere e in bianco quelle libere, poi quelle nere saranno un bit TRUE (1 logico) mentre quelle bianche saranno FALSE (0 logico).

Ogni elemento della matrice definisce un pixel per la risoluzione e il tipo di immagine che abbiamo scelto.

Si noti che il pixel è definito come l'unità più piccola in cui si può suddividere l'immagine ma non è definito la sua dimensione fisica che può variare, ad esempio quando posta un'immagine più piccola dello schermo al centro di esso, e stirata al fine di riempirlo tutto, i pixel cambieranno la dimensione con una ovvia perdita di "definizione" dell'immagine.

In sostanza l'immagine mantiene la stessa risoluzione ma cambia la definizione degradando l'aspetto se la si ingrandisce e migliorandolo, nel limite del compromesso, riducendola.

La risoluzione rimane definita come la densità di informazione contenuta nell'unità di superficie in esame che solitamente viene presa come riferimento, il pollice quadrato.

Il numero di **P**ixel che sono posti in **P**ollice quadrato assume l'unità di misura di **PPI** (**p**ixel **p**er square **i**nch) ed è un parametro fondamentale per definire la qualità di stampa.

Se ogni pixel fosse un bit, quindi un boolean, l'occupazione in memoria dell'immagina "A" che stimo andando a creare sarà di 30 bit, pari a 4 byte (di cui 2 bit saranno inutilizzati), ciò definisce una risoluzione di 30bits.

	A	B	C	D	E	F	G
1							
2				1			
3			1		1		
4		1				1	
5		1	1	1	1	1	
6		1				1	
7		1				1	
8							
9							

Nella stessa matrice inseriamo i bit che non sono a 1 logico ovvero tutti i rimanenti li metto a 0 quindi FALSE.

Si ottiene questa matrice:

	A	B	C	D	E	F	G
1							
2		0	0	1	0	0	
3		0	1	0	1	0	
4		1	0	0	0	1	
5		1	1	1	1	1	
6		1	0	0	0	1	
7		1	0	0	0	1	

Ora, a partire da in basso a sinistra numero i bits che saranno quindi 30, e poi li pongo in sequenza, ottenendo un vettore.

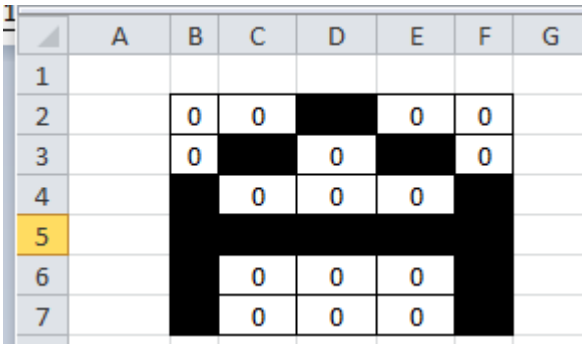
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	1	0	0

Ora posso mettere in memoria RAM o SSD l'immagine digitalizzata oppure trasmetterla con un protocollo ad esempio seriale a una periferica.

Questi bit vengono poi riconvertiti nell'immagine originali sostituendo il colore nero al bit = 1 e restando bianco il bit =0.

Ecco il risultato finale visualizzato sul monitor o display della periferica che riceve.



Vediamo un esempio di programma C++, svolto con il compilatore contenuto nella piattaforma DevC++, in cui si introduce in memoria la codifica sotto forma di vettore della lettera A e successivamente la si estrae per vedere se il calcolatore riesce a individuare nella stringa di 1 e 0 il carattere A.

```
#include <iostream>
#include <conio.h>

using namespace std;

void splash(){
        cout<<"*********************************\n";
        cout<<"* Scuola SIIC 2021 cl.Liceo 1a        *\n";
       cout<<"*   prof. Marco Gottardo PhD          *\n";
       cout<<"* Codifica digitale lettere B/N        *\n";
       cout<<"**       17/03/2021                   **\n";
       cout<<"*********************************\n"<<endl;
       cout<<"\n Press any key \n";

getch();
```

```
}

void byebye(){
	system("cls");
	cout<<"**********************************\n";
	cout<<"*  programma eseguito correttamente  *\n";
	cout<<"*		codifica digitale lettere	**\n";
	cout<<"**			17/03/2021		**\n";
	cout<<"**********************************\n"<<endl;
	cout<<"\n Press any key \n";
}

void letter_A(){
	int lettera[30];
	//10001100011111110001010100010 = A letter	(codificare anche le altre lettere)
	lettera[30]='10001100011111110001010100010';
/*	for (int index=0; index<30; index++){  // possibilità manuale di introduzione codifica runtime
		cin>>lettera[index];
	}
*/
	if (lettera[30]=='10001100011111110001010100010'){
		cout<<"hai codificato la lettera A\n";
	}
	else cout<<"Code have no sense !!! ";
	getch();
}

int main(int argc, char** argv) {
	splash();
	letter_A();
	byebye();
	return 0;
}
```

Le immagini in bianco e nero sono molto facili da codificare, specialmente quando sono in bianco e nero e a bassa risoluzione, come avviene per i caratteri ASCII.

Le fotografie in bianco e nero

Le foto in bianco e nero sono in realtà create in toni di grigio e ogni pixel richiede almeno un byte, quindi 8 bit per pixel.

8 bit, all'interno di un byte possono esprime fino a $256=2^8$ sfumature di grigio, dal bianco 00000000, fino al nero completo 11111111.

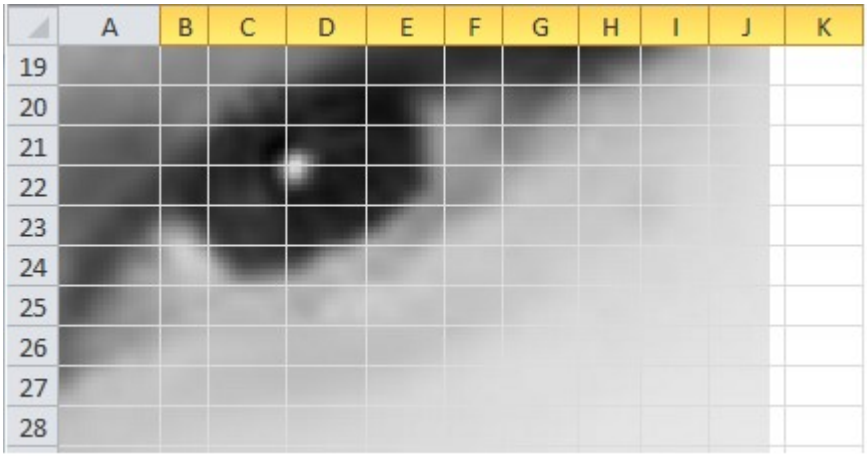
Consideriamo la foto mostrata in B/N ovvero in sfumature di grigio, mostrata sotto.

Nel complesso è una distribuzione omogenea di pixel, ciascuno di 8bit= 1 byte il cui numero si può ricavare dalla quantità presente in una riga, moltiplicato per il numero di righe, di fatto numero delle colonne per il numero delle righe.

Per semplicità espositiva catturiamo un particolare dell'intera foto e sopra di questo poniamo la nostra griglia al fine di identificarne i pixel.

Il programma in C++ rimane lo stesso proposto sopra, dato che i pixel erano già definiti INT, benché ci fosse uno spreco di memoria perché sarebbero bastati definiti *bit lettera[30]; invece che int lettera[30];*

Catturiamo il particolare dell'occhio sinistro (quello a destra nella foto), e mettiamolo in una griglia pari a 10 righe per 10 colonne.



In primo luogo questa porzione di immagine avrà, senza applicare nessuna compressione, una occupazione in memoria pari a 10 byte per 10 byte = 100byte.

Se osserviamo la cella D22, potremmo stimare un valore piuttosto alto, perché vicino al nero, ad esempio 200 su 255 possibili, mentre il pixel J28 sarà molto basso perché piuttosto vicino al bianco.

D22=11001000 J28=00101101

Con questa codifica bisogna però dare per scontato che la posizione dei pixel rimane fissata all'interno della matrice e identificabile dal suo indice di riga e di colonna, se invece è ridotta a un vettore monodimensionale dal solo indice progressivo.

Per la risoluzione più ampia, ad esempio lo standard VGA costituito 640 x 480 punti, il concetto non cambia, è solo maggiore la dimensione della mappa.

Le immagini a colori

Per le immagini a colori, una volta associato a ciascun colore un numero intero, e definito il pixel come unità della matrice 640x480, è possibile trasmettendo tre sole informazioni, ovvero coordinata x, coordinata y, profondità del colore come numero intero, codificare, memorizzare, trasmettere, ricostruire l'immagine originaria.

La tecnica **RGB**, R=red, G= green, B=blue è la più nota tecnica di codifica delle immagini a colori benché la più anziana e la meno compressa.

Richiede tre byte per ogni pixel, quindi 24 bit per ogni pixel e questo formato è noto in bibliografia come **true color.**

Dato che le sfumature dei colori, percepite dall'occhio, saranno date dalla sovrapposizione delle tonalità rossa verde e blu, tra loro amalgamate, il numero di colori generabili o rappresentabili sarà pari a 22^4=16777216 colori, che genericamente viene arrotondato e citato come 16 milioni di colori.

Come detto, questi 16 milioni di colore sono da attribuirsi al mix di 256 sfumature di Rosso con 256 sfumature di verde G, con 256 sfumature di blu.

n°	Red	Green	Blue
0	0	0	0
1	35	22	76
2	44	33	75
3	255	3	6
4			
5			
6			
7			

Un'altra tecnica è quella della tavolozza da pittore, che si traduce in **palette**, che utilizzando la tecnica RGB ma pescando i colori come numero espresso dai 24 bit dalla tavolozza dei colori disponibili.

In questo caso il numero del colore è detto "ordine".

Va però ricalcolato il peso dell'immagine, quando non compressa, perché al peso dell'immagine i per se stessa va addizionato il peso della tavolozza.

In sostanza vengono occupate due aree di memoria, una per l'immagine grezza, ovvero le coordinate dei pixel e una per le profondità del colore dato appunto dalla tavolozza.

Il calcolo del peso di un'immagine archiviata con il metodo palette (tavolozza) richiede l'applicazione di una formula, questa è:

n° colori * 3 byte (uno per R, uno per G e uno per B, tot 3 byte) + numero pixel dell'immagine * 1 (oppure2)

Il fattore moltiplicativo 1 si applica se l'immagine ha un numero di colori <= 256, mentre il fattore moltiplicativo 2 si applica se l'immagine ha un numero di colori > di 256 colori.

Ad esempio, se un'immagine fosse composta di soli 10 colori, espressi in RGB, estesa 500 pixel, il peso sarebbe:

$$10 \cdot 3 + 500 \cdot 2 = 1030 Byte \text{ [metodo palette]}$$

A titolo comparativo, la stessa immagine rappresentata con il metodo tre colors, ovvero bitmap, avrebbe avuto un peso pari a:

$$500 \cdot 3 = 1500\ Byte$$

La differenza di peso per i due metodi di rappresentazione e memorizzazione è di ben 470 Byte anche per un'immagine così piccola essendo di soli 500 pixel.

Aumentando la risoluzione diventeranno necessari anche strategie, o meglio algoritmi, che consentano di compattare le immagini.

I formati più in uso oggi sono GIF, JPEG, PNG e TIFF usano infatti metodi di compressione diversi; a parità di dimensioni in pixel, le dimensioni del file possono variare molto.

Esistono anche i formati vettoriali, quelli con cui lavorano i CAD e le macchini operatrici come le frese e i torni a controllo numerico, questi sono DXF (Drawing Exchange Format), DWG (Drawing) e SVG (Sacalable Vector Graphic).

I formati vettoriali possono essere scalati senza perdita di informazione e risoluzione, ma richiedono molti calcoli e perdono la qualità del colore e delle sfumature, sono necessari per questo algoritmi specifici.

Il sensore in uso più diffuso oggi, per la trasduzione delle immagini si chiama CCD, Coupled Charged Device.

I suoni in formato digitale

Un suono è l'effetto di variazione di pressione dell'aria che tramite un'onda colpisce il nostro timpano, che ne riproduce l'oscillazione. E' compito del cervello il dare un significa comprensibile a questi segnali resi elettrici dal sistema nervoso. In sostanza il cervello interpreta i segnali grazie a meccanismi cognitivi molto complessi.

Il computer, pur non capendo il contenuto dei suoni, ma nel migliore dei casi confrontandoli con ciò che in precedenza sia già stato codificato, può associarli a reazioni per il quale sia stato addestrato come avviene nei sistemi informatici a reti neurali che sono i promotori dell'intelligenza artificiale detta A.I.

Il suono deve essere trasdotto (convertito dalla sua forma naturale in un segnale elettrico che il computer possa almeno leggere), campionato (catturato a intervalli regolari per ottenerne un numero che ne indica la proporzione tra un minimo e un massimo), e quantizzato.

Il trasduttore più comune per catturare un suono è il microfono la cui membrana oscillante nuove meccanicamente una bobina all'interno di un magnete permanente creando grazie ana legge fisica elettrica, chiamata legge di Lentz una forza magneto motrice che può essere amplifica e che corrisponde alla forma dell'onda di pressione che ha colpito la membrana.

Il suono deve essere quantizzato, ovvero il segnale trasdotto, continuo nel tempo, diventa valori numerici presi a intervalli ti tempo fissi detti tempi di campionamento.

Il numero di campioni del segnale presi ogni secondo sono la frequenza di campionamento.

Se campioniamo ogni millesimo dio secondo avremo una frequenza di campionamento di un chilo Hertz che si scrive 1kHz.

Il numero di bit con cui posso rappresentare l'onda sonora campionata sta sull'asse verticale e esprime quanto deve variare il suono per avere un numero che lo rappresenta diverso rispetto al campione precedente.

Ad esempio, se il mio segnale ha un livello massimo di 200mV, e posso campionarlo con 10 bit, significa che la risoluzione sarà pari a 2^{10}=1024 livelli diversi.

Quindi dividendo il fondo scala del segnale di 200mV per questa risoluzione ottengo che la più piccola variazione di cui si accorge il mio sistema è 200mV/1024=0,19mV

Il peso in byte di un brano musicale campionato si ottiene considerando il numero di campioni per secondo (la frequenza di campionamento) e alla quantizzazione (il numero di bit in cui si trasduce, ovvero che si usano per ogni campione) si avranno diversi valori di BitRate avvero bit che dovremmo trasmettere e che il canale dovrà supportare in trasmissione ogni secondo.

Ad esempio, considerando un vecchi lettore CD, che produce un formato WAW, con un campionamento a 44.100Hz, con una quantizzazione monofonica a 16 bit, si avrà un BitRate di 1,4Mbit al secondo.

Per un'ora di audio bisogna moltiplicare questo 1,4 Mbit per 3600 che è il numero di secondi che compongono un'ora.

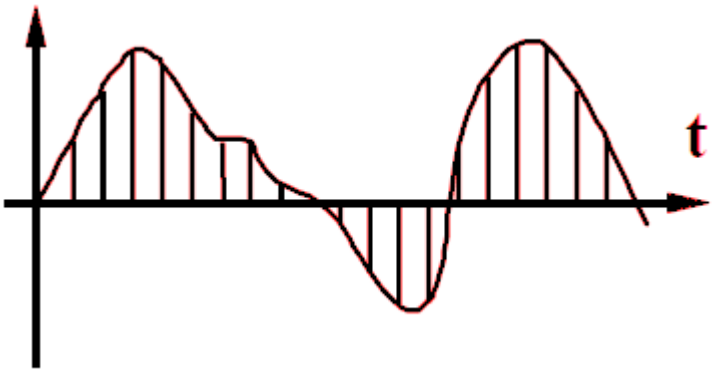
Per quanto riguarda la frequenza minima di campionamento questa dipenderà dal segnale che vogliamo memorizzare e poi riprodurre nella maniera più affidabile possibile.

Il campionamento dovrà essere almeno il doppio della frequenza massima presente nel segnale da campionare, questa è detta la frequenza di Nyquist.

Se ad esempio volessimo campionare voce umana la cui frequenza massima è 4kHz sarà necessaria una frequenza minima di campionamento di 8000 kHz ovvero campioni al secondo

Per avere una buona percezione dei segnali udibili dall'essere umano, essendo la banda passante del nostri orecchio estesa da 20 a 20kHz dovremmo campionare il segnale almeno a 40.000 campioni al secondo.

Per fare un esempio il tasso di campionamento dei compact disc è di 44.100 campioni/sec, ed è quindi sufficiente a rappresentare in alta fedeltà segnale audio di partenza.



La compressione dei dati

La compressione dei dati permette di aumentare la densità di informazione per unità di memoria quindi un sistema di archiviazione avrà una qualità maggiore di un altro in funzione della quantità di dati che riesce ad archiviare per unita di spazio.

Questa qualità di compressione diventa essenziale quando i si debba non solo archiviare i dati, ovvero teneri fermi nella memoria, ma anche quando questi debbano essere trasmessi, riducendo i tempi a parità di banda.

Supponiamo di dovere archiviare un documento di solo testo di dimensione A4, nel caso dovessimo archiviare ogni puntino che compongono le lettere del testo, i dati da archiviare sarebbero una quantità enorme, nella modalità bitmap anche se monocromatica.

In questo caso benché i pixel siano solo o bianchi o neri la quantità di questi è molto elevata anche per una singola pagina.

Se ci interessa archiviare solo il testo, non avremmo bisogno di memorizzare anche i pixel bianchi presenti nel foglio, sarà sufficiente memorizzare la sequenza dei caratteri che compongono le parole, gli spazi bianchi che separano le parole, i segni di punteggiatura, e qualche altro dato relativo al codice specifico relativo font usato.

La compressione dei file, potrebbe avvenire con due condizioni distinte, la prima con una perdita di informazione, anche se considerata accettabile, come ad esempio nel comprimere una immagine bitmap estremamente dettagliata

In una foto questo si può fare stimando quanti pixel uguali o simili si possono togliere per millimetro quadrato, riducendone sicuramente la quantità di byte necessari per la descrizione, ma a costo di perdita di nitidezza, mentre, con algoritmi diversi, applicati per esempio a dei file di testo, e possibile non perdere la quantità di informazione originale, a patto che il file, venga da prima compresso e poi trasmesso e infine archiviato.

Una volta trasmesso è necessario però applicare il medesimo algoritmo ma complementare al file per ottenerne l'espansione riportandolo allo stato originale.

In mancanza di questa procedura inversa il file risulterebbe illeggibile anche per il programma che lo ha generato.

Ci sono molti metodi per comprimere i files ma tutti hanno lo scopo comune di ottenere un unico file da cartelle contenute in una cartella principale o files sparsi all'interno di queste.

Anche se ora non viene molto usato è possibile spezzare l'archivio finale in più volumi al fine di poterle spostare o archiviare per parti. Ad esempio alcuni vecchi sistemi operativi non riescono a maneggiare file di dimensione superiore a 2 Giga Byte.

Le immagini ISO sono un metodo di compressione che permette di archiviare interi dischi rigidi o il contenuto di CD rom o DVD rom, come anche di programma, in un unico volume ricostruibile.

Ci sono molti compressori standard tipici per in vari campi informatici, quali ad esempio la fotografia, i video, i file audio, oppure i classici come il formato ZIP o il formato 7z, RAR ecc per Windows mentre per i sistemi linux, “tar.gz” o “tar.bz”.

Vediamo un semplice esempio in C++ che permette di compattare un file di test piuttosto che memorizzarlo come immagine bitmap.

Viene lasciato al letto il programma di esempio che realizza una matrice compatta

L'ARCHITETTURA DEL COMPUTER

Riconoscere le caratteristiche logico funzionali di un computer e il ruolo strumentale svolto nei vari ambiti(calcolo, elaborazione, comunicazione)

Il termine "computer" tratto dall'odierna lingua anglosassone, in realtà ha radici ben più profonde, ed in particolare prende origine dal termine latino "computare", un verbo composto da <cum> e <putare> che grosso modo poteva significare " tagliare e mettere insieme", quindi esplicabile anche con sinonimi come comparare, confrontare o contare… ed e quindi una apparecchiatura atta ad elaborare dei dati, infatti in altre lingue come il francese viene denominato "ordinateur", o anche in italiano, "elaboratore elettronico".

La storia dell'evoluzione delle "macchine da calcolo" ha origini molto antiche, ma in riferimento all'era moderna, fu lo "**Z1**", il primo di un'innovativa serie di calcolatori elettromeccanici basati sul sistema binario e programmabili, funzionanti prima a memorie elettromeccaniche e poi a relè (Z2, Z3).

La macchina presentava una struttura già del tutto analoga a quella dei computer moderni, con la distinzione tra unità di memoria ed unità di calcolo e funzionava alla velocità di clock di un solo Hertz, generata da un motore elettrico.

Gli studi di Zuse e quelli di John Vincent Atanasoff, inventore della memoria rigenerativa, furono la base principale per l'elaborazione dell'architettura di von Neumann.

Evidentemente esistono diverse categorie di computer, una volta venivano denominati "home" e "Personal", adesso invece si parte dai più piccoli a scheda singola (SBC) che sono molto apprezzati sia dai tinker che dagli hobbisti ed offrono molte funzionalità in un fattore di forma molto piccolo.

Un SBC ha la CPU, la GPU, la memoria, le porte IO, ecc... su un piccolo circuito stampato e gli utenti possono aggiungere funzionalità aggiungendo nuovi dispositivi alle porte GPIO (acronimo di General Purpose Input/Output).

Poi ci sono i "desktop" o portatili, ivi compresi i "tablet" ed i "palmari", poi i PC "fissi", rimanendo sempre nell'ambito di utilizzo comune denominati anche "Microcomputer" perché sono computer caratterizzati dalla presenza di un singolo microprocessore, poi ai "Minicomputer" che sono computer con prestazioni e potenza intermedie tra un "Maiframe" e un microcomputer, in grado di consentire l'accesso a più utenti

Salendo di categoria invece si incontrano le " Workstation" che sono computer monoutente ad elevate performance di lavoro utilizzati per l'esecuzione di software professionali.

I Mainframe invece sono computer ad elevate performance in grado di svolgere elaborazioni dati molto complesse, e poi si passa ai "Supercomputer" ovvero, computer ad elevata potenza di calcolo specializzati nell'esecuzione di specifiche operazioni, prevalentemente utilizzati in ambito scientifico.

Nel mondo di internet si parla anche ovviamente di "reti di computers".

Le differenziazioni sono quindi molteplici, basate su svariati fattori costitutivi quali ad esempio:

- la velocita di "clock" dei microprocessori (CPU e GPU)
- il "bus" caratteristico (tipico dei microprocessori usati)

- la quantità di memoria RAM utilizzata per l'elaborazione e la grafica
- Le caratteristiche delle periferiche di I/O (memorie interne, memorie di massa, schede grafiche ecc.)
- La quantità e la qualità delle porte di comunicazione
- Il "firmware" ed "Kernel" che in informatica costituiscono il nucleo o core di un sistema operativo, ovvero il software che fornisce un accesso sicuro e controllato dell'hardware ai processi in esecuzione sul computer.
- Il "sistema operativo" che non è altro che un insieme di software che fornisce all'utente una serie di comandi e servizi per usufruire al meglio della potenza di calcolo di un qualsivoglia elaboratore elettronico

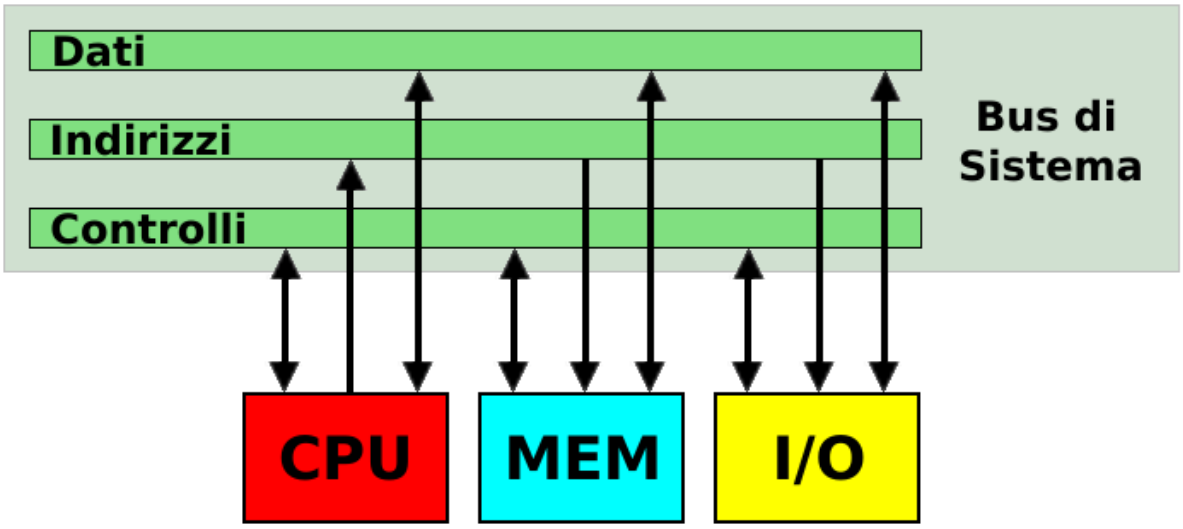
Quindi, i computer, sono delle apparecchiature che differiscono, sia per le loro peculiarità hardware, ma anche per le loro caratteristiche software, vi saranno quindi computer con migliori prestazioni di calcolo ed altri con migliori prestazioni grafiche, altri ancora saranno più adatti alle comunicazioni come quelli denominati "server", la cui caratteristica principale e quella di rendere il più veloce possibile lo scambio dei dati in rete.

Da notare inoltre che ci sono computer specifici per l'ambiente industriale, con particolari caratteristiche costruttive quali grado "IP", robustezza del contenitore ecc. ed altri computer utilizzati in ambito medicale, dove anche il software installato ha delle particolari certificazioni.

Il modello di von Neumann

Il modello di Von Neumann schematizza il principio di funzionamento dei moderno computer come macchine che devono avere tre tipi di funzione minima, queste sono:

1. Capacità di accedere a una memoria in lettura e scrittura.
2. Capacità di indirizzare e controllare periferiche
3. Capacità di calcolo nell'unità centrale di processo CPU



Secondo Von Nuemann un computer è un sistema capace di elaborare dati nell'unità di processo di memorizzare e prelevare questi dati dalla memoria, e infine di comunicare con dispositivi hardware detti periferiche.

Hardware e software

Il computer è formato da una parte hardware e una parte software, l'hardware è di solito costituito da una scheda madre che contiene il processore centrale (CPU) la memoria RAM, il chipset, la ventola di raffreddamento del processore, le porte multimediali tra cui la USB per il collegamento seriale di periferiche, la periferica audio, con canali sia di input che di output, ad esempio le cuffie e il microfono, la scheda video che permette la connessione con il monitor, e altri connettore per ad esempio collegare la tastiera e il mouse.

il tutto è contenuto in una scatola che si chiama chassy, che può avere varie forme, ad esempio il minitower o il desktop.

Se si tratta di un PC domestico fisso oppure è un portatile che ha la struttura classica con monitor integrato.

la parte software necessita sempre di un sistema operativo.

I più comuni sono Windows e Linux ma ce ne sono molti altri, a seconda dell'uso che si vuole fare del computer.

Il case e la scheda madre

Cabinet per PC (detti anche "case" o "chassy"). Esistono varie forme per il cabinet del PC, ma essi hanno tutti delle funzionalità in comune, queste sono:

- Fori di ancoraggio della mainboard predisposti sulle misure standard ATX, mini ATX con fori filettati per i distanziatori metallici.
- Alloggiamento di almeno un alimentatore ATX nella parte superiore.
- Aperture posteriori per le schede aggiuntive con passo standardizzato. Tipicamente sette.
- Pulsanti comando e LED di segnalazione sul frontale
- Flusso d'aria precalcolato.
- Formato Tower, middle tower, mini tower, desktop

Vista interna di un cabinet middle tower

Sul lato destro vediamo una torretta interna in cui potranno essere alloggiati gli Hard drive e sopra di questi il lettore DVD.
Le viti collegate alle parti metalliche condurranno anche il calore verso il cabinet allungando la vita di questi componenti.
Per questa ragione termica è bene fissare tutte le viti degli hard disk anche se 4 sembrano tante ed inutili.

Aggiungere ventole al PC spesso porta svantaggio invece che una miglioria del raffreddamento. Il PC può diventare molto rumoroso, e vibrare fastidiosamente specialmente se le ventole raccolgono polvere e le viti di chiusura non sono ben serrate.

Il cabinet rappresenta anche la schermatura dei circuiti del PC da interferenze di immissione e irraggiamento elettromagnetico, quindi il PC dovrebbe sempre lavorare chiuso.
A case chiuso non viene alterato il flusso d'aria calcolato per le ventole e ci si protegge dalla polvere oltre che ovviamente ad essere schermati.

I case dedicati ai server sono sempre piuttosto ingombrati e pesanti a causa della loro destinazione a lavorare in maniera gravosa ovvero sempre accesi per molti anni.
Esistono però dei server “leggeri” più adatti alle destinazioni domestiche, per quelle applicazioni home in cui abbia senso istallare una rete a dominio. In un server la prima cosa che salta all’occhio è sempre la predisposizione dell’istallazione dei dischi in RAID e la possibilità di eseguire degli HotSwap, oppero delle estrazioni o reinserimento a caldo.
La manovra pero’ non va presa alla leggera ed e’ spiegata nel libro “corso di istallatore manutentore e amministratore di reti LAN”

Diciamo comunque che un server propriamente detto, supera facilmente i 35 chilogrammi di peso e a causa dell’elevato numero di ventole che contiene è normalmente troppo rumoroso per poter operare in casa o anche in un ufficio.
I server sono di solito relegati alla sala server e l’amministratore di rete può accedervi in amministrazione remota dalla postazione in ufficio tecnico. Raramente dovrà andare in sala server per manutenzioni non ordinarie.
Vediamo alcuni case specifici per sala server.

CASE PER SERVER ATX MONTAGGIO SENZA VITI. 3 BAY 5"1/4. 7 BAY 3"1/2. 5 ALLOGGI PER VENTOLE OPZIONALI. PORTE USB. FIREWIRE. MIC+LINE LATERALI ADD-ON SERVER-BRAVO	
Prezzo medio su web	€ 74,10 iva compresa
Produttore	ADD-ON
Partnumber	SERVER-BRAVO
Larghezza	70cm
Dim Profondità	25 m
Altezza	55 cm
Peso	1 K

Alcuni case sono progettati per il montaggio su Rack, ovvero un grosso armadio, alto circa due metri che può alloggiare più server e più dispositivi dedicati alle infrastrutture di rete come i grandi HUB/Switch per la costruzione di una rete aziendale.

Le misure di questi armadi elettrici sono standardizzate e note con il nome “Europa 2”.

CASE PER SERVER 4 UNITA. MONTAGGIO SU RACK 3 BAY 5"1/4. 2 BAY 3"1/2. VENTOLA 12 CM CON FILTRO INCLUSA FRONTALE NERO CON CHIAVE ADD-ON RACK-4U-NERO (RACK4UNERO)	
Prezzo medio su web	€ 149,30 iva compresa
Produttore	ADD-ON
Partnumber	RACK-4U-NERO
Larghezza	70 cm
Dim Profondità	25 cm
Altezza	55 cm
Peso	16 Kg

Tipico server aziendale

L'armadio contiene sempre dei montanti di fissaggio che consentono l'ancoraggio di dispositivi industriali a misura standard "Europa 2" ovvero 19 pollici, che corrispondono a circa 48 cm.

I Fianchi di questi armadi sono smontabili per permettere l'affiancamento con altri armadi.

Le porte possono essere vincolate su uno oppure l'atro lato.

Il fondo dell'armadio e' aperto per consentire il transito delle alimentazioni e del grande numero di cavi.

Nella parte superiore si nota un ampio numero di slot per le unita' scasi connesse in RAID.

Le CPU sono dislocate nella parte centrale assieme agli hub switch.

Vediamo l'aspetto di un classico armadio per contenere computer server su più ante.

Solitamente i server sono accompagnati da unità piuttosto pesanti ed ingombranti che servono al controllo delle alimentazioni in caso di mancanza della rete.

Tali unità dette UPS producono anche dei segnali di intervento che impongono ai server di salvare le sessioni in corso, di fare il back up di alcuni dati o cartelle importanti e di lanciare verso i terminali dei client l'avviso che il sistema darà entro un tempo di pochi minuti, prefissati dall'amministratore di rete, uno shutdown generale per la salvaguardia del sistema stessa ora alimentato con le batterie interne dell'UPS.

Alcuni computer pur non essendo server dovranno essere montati su Rack normalizzato perché' orientati all'uso industriale. Vengono quindi prodotti questi Chassy in alluminio con misure standard "Europa 2".

MANHATTAN

• Robusto châssis industriale in alluminio per applicazioni a rack 19”. incontra tutte le specifiche richieste dalla normativa CE. • Comodo design per le applicazioni rack 19”. ogni periferica installata è facilmente raggiungibile • Sbarra di rinforzo interno per proteggere da vibrazioni le schede interne • Alloggi frontali: 1 x slot 5.25”. 1 x slot 3.5”. interno 4 x 3.5”. • **Include PCI raiser card** - • Alimentatore: ATX 300 Watt Pentium 4. • Dimensioni: 482 x 610 x 88 mm (2UN) - • Dimensioni: 482 x 508 x 88 mm. • Indicatori Led frontali per lo status HDD/accensione • Include **5 ventole** 80x80 cm (2 interne e 3 esterne) - - - MANHATTAN ICC IO-PCI-R (ICCIOPCIR)	
Prezzo medio su web*	€ 19,40 iva compresa
Produttore	MANHATTAN
Partnumber	ICC IO-PCI-R
Larghezza	70 cm
Dim Profondità	25 cm
Altezza	55 cm
Peso	16 Kg

La scheda madre

Le schede madri, anche dette piastre madri oppure anche in Italia si usa il nome inglese motherboard, sono il componente principale non solo nei computer ma di tutti i device elettronici in cui esista un processore. Ovviamente le schede madri possono essere molto diverse tra loro: differiscono anzitutto per il formato.

Il formato esiste in varie versioni ovvero dimensioni anche per l'uso standard dentro ai computer, si immagini quindi in altri dispositivi di alto consumo, ad esempio i telefoni.

Tutte le schede madri vengono prodotte con a bordo dei controller per le periferiche di massa ovvero per gli Hard Disk.

Lo standard attuale di questi controller è il SATA che permette l'istallazione di un unico dispositivo per ogni connettore, ma questo spesso convive con il vecchi standard IDE/ATA che a volte può risultare il solo presente.

Spesso, soprattutto in macchine destinate ad uso come server, esistono dei controller aggiuntivi istallati su slot PCI che permettono di eseguire istallazioni dei dischi di tipo RAID anche se il bios e l'hardware della scheda madre non lo prevede.

Consideriamo per il momento i controller IDE/ATA, essi si presentano con due connettori a 40 pin maschi, spesso uno è di colore nero e l'altro di colore BLU per distinguere il canale IDE standard da quello ultra DMA o "PATA" da usarsi come primario.

Ogni canale dei due disponibili supporta due dispositivi per un totale di 4 dischi o meglio di 4 unità di memorizzazione infatti qui potremmo collegare anche il CD-Rom o il DVD drive.

In ogni canale sarà comunque presente un dispositivo (device o drive) master e uno slave che andranno configurati usando i jumper presente sul loro connettore. Usualmente i costruttori dei driver mettono una tabellina su una etichetta applicata sul corpo del dispositivo stesso che ci spiega come chiudere i jumper.

Esiste sempre anche una terza possibilità che è quella di lasciare determinare al cavo di collegamento quale dei device è master o slave, ovvero ponendo il jumper in posizione "CS" (cable select). Ovviamente, in questo caso, è obbligo che entrambi i dispositivi si trovino selezionati in posizione CS del jumper per non creare conflitti.

Se la tabellina di impostazione del jumper fosse stata rimossa o resa illeggibile dal tempo è sempre possibile visitare gli archivi elettronici presenti nei siti web dei costruttori.

Nel concreto avremmo il canale nel connettore blu abbinato al canale primario e il connettore nero al canale secondario.

Si ottiene quindi:

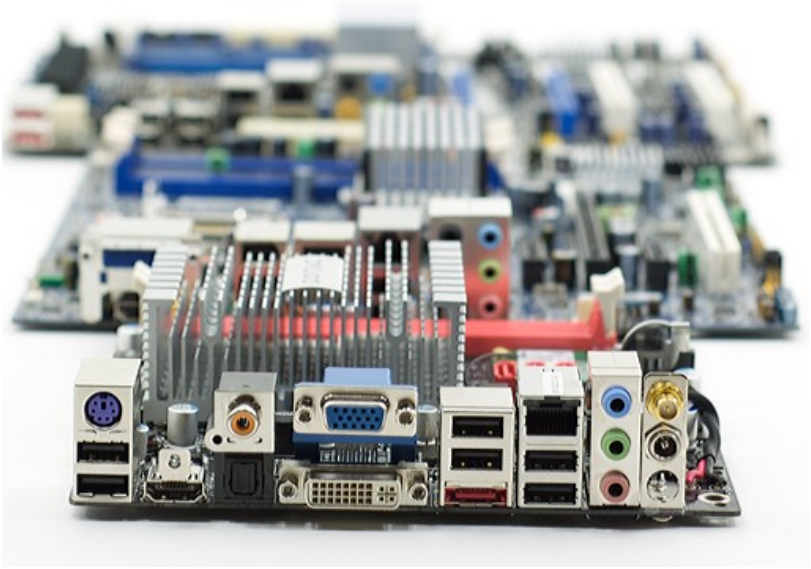
- primary-master
- primary-slave
- secondary-master
- secondary-slave

La prima partizione, detta usualmente primaria, si trova sempre nel disco primario master che di solito I sistemi windows indicano con C:

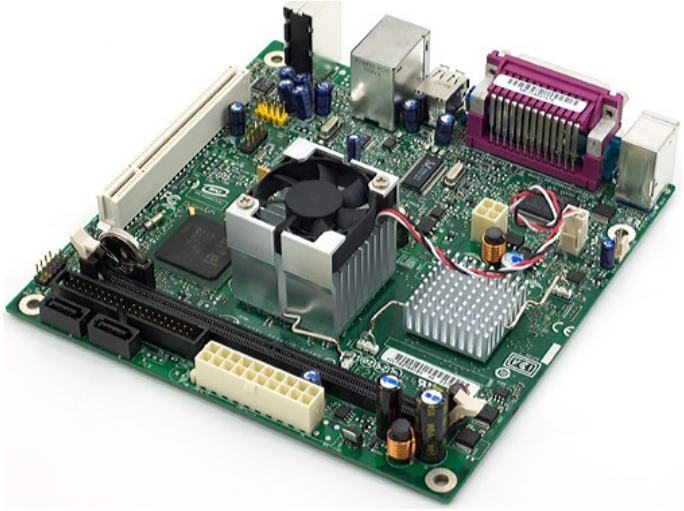
Versioni miniaturizzate di schede madri

l formato mini-ITX è il più piccolo ed è nato da un'idea di VIA Technologies, ma poi è stato ampiamente adottato anche per soluzioni con CPU AMD e Intel. Si tratta di un formato con dimensioni di 17 x 17 centimetri e spesso viene usato in PC raffreddati passivamente, ovvero senza ventole. Le schede madri mini-ITX vanno spesso bene per creare mini-PC oppure per sistemi home theather.

Il micro-ATX è stato introdotto nel 1997 e ha dimensioni maggiori rispetto al mini-ITX. Le dimensioni di una scheda madre micro-ATX sono 244 x 244 mm. Negli anni questo formato ha visto vari e interessanti sviluppi. Sono numerosi i produttori che realizzano schede madri compatte per giocatori, quindi provviste di diversi slot PCI Express per ospitare più schede video, ma anche un insieme di porte molto ampio. Inoltre, sono pensate per gli overclocker, ovvero chi desidera aumentare la frequenza dei vari componenti per ottenere maggiori prestazioni.



Intel è stata in realtà la prima a commercializzare una scheda madre basata su Atom mini-ITX: la D945GCLF. Basata su un Atom 230 single-core a 1,6 GHz, la D945GCLF è una scheda madre relativamente semplice.
Il processore ATOM ora è in disuso ma per circa 5 anni, attorno al 2010, è stato impiegato per la costruzione dei primi portatili molto compatti e poco costosi.
Le caratteristiche tecniche non erano molto performanti ma a basso consumo e facili da impiegare con i driver del sistema operativo di allora, Windows XP ma anche il successivo Windows 7. In primo piano Mini-ITX, dietro vediamo la micro-ATX, dietro ATX



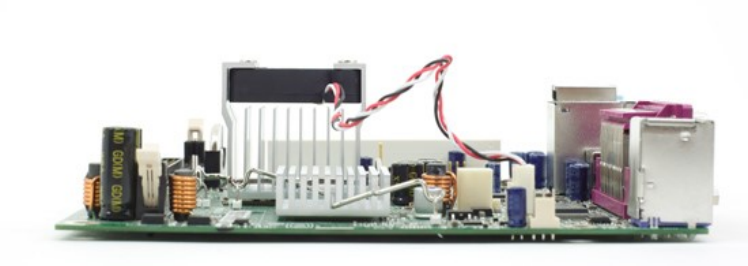
A partire dal lato inferiore è visibile il connettore bianco ATX per collegare l'alimentatore, sempre sul lato inferiore i connettori SATA e un IDE per i vecchi modelli di hard disk. Ora questo connettore è stato eliminato e gli hard disk con quella interfaccia non sono più fabbricati.

Il connettore lungo, è l'alloggio della DDR RAM.

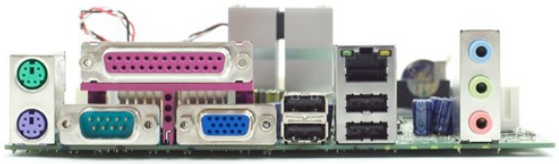
Con l'aletta di raffreddamento ventilata è visibile ma CPU mentre l'aletta passiva è il circuito integrato del chipset.

A sinistra della CPU c'è il south bridge, un circuito integrato che consente l'interfacciamento

Dispone di un singolo slot DIMM DDR2 che supporta fino a 2 GB di memoria (solo DDR2-400 o 533). Non c'è supporto per una GPU moderna, c'è solo un vecchio slot PCI a 32 bit sulla scheda madre. Hai due connettori SATA e un connettore PATA a bordo, quattro USB, nessuna uscita DVI / HDMI (solo VGA) e un set standard di tre porte audio analogiche.



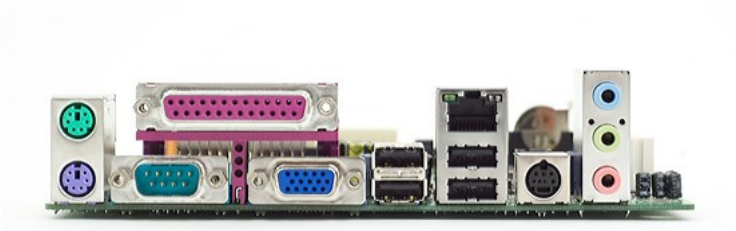
Il dissipatore di calore alto nella parte posteriore è per il chipset, quello piccolo è tutto ciò di cui hai bisogno per la CPU



Come suggerisce il nome, il D945GCLF utilizza il chipset 945G di Intel e una grafica corrispondentemente lenta. Tuttavia, se non hai bisogno di un sacco di prestazioni, la soluzione Atom desktop di Intel è piuttosto interessante in quanto viene venduta per meno di $ 70 - CPU inclusa (è saldata sulla scheda).

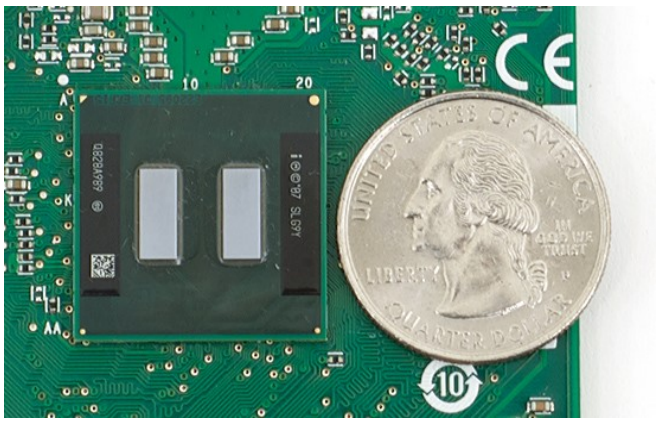
Con la silenziosa introduzione del dual-core Atom, Intel ha rilasciato il D945GCLF2. Quasi identico al suo predecessore single core, il D945GCLF2 utilizza un dissipatore di calore più grande sulla CPU e uno più piccolo sulla GMCH (sebbene sia ancora raffreddato da una ventola). La scheda utilizza anche un connettore di alimentazione ATX a 24 pin invece di un connettore a 20 pin, aggiunge un'uscita S-Video e una porta Gigabit Ethernet (il D945GCLF ha solo una porta 10/100).

Il dual core D945GCLF2 aggiunge un secondo core alla piattaforma Atom.



Bland ports on the D945GCLF2

La più grande differenza è ovviamente l'inclusione del processore Atom 330, che è semplicemente composto da due Atom 230 in un unico pacchetto, entrambi funzionanti a 1,6 GHz:



A dual core Atom 330

La CPU più veloce ha il costo più alto fino a $ 80, ma rimane ancora molto conveniente rispetto ai processori Pentium.

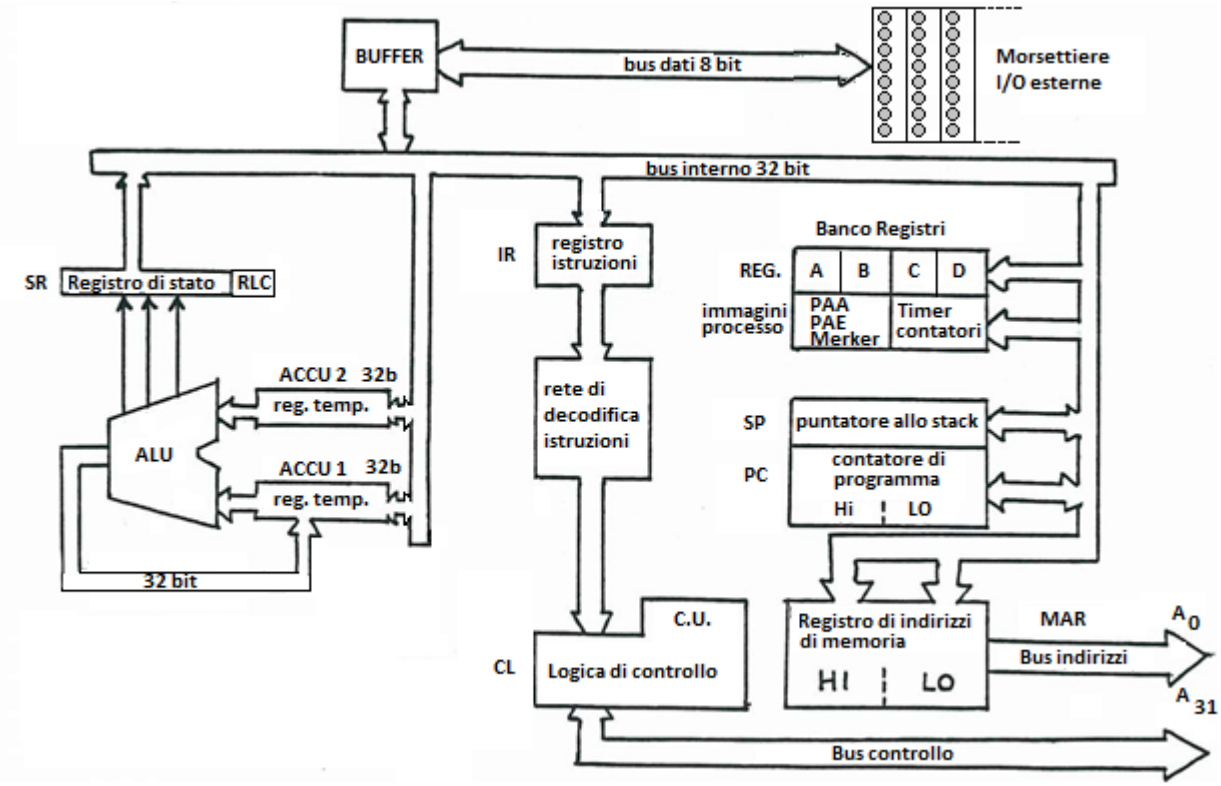
Il processore Atom da 45 nm funziona a temperature notevolmente inferiori rispetto al 945 GMCH sulla scheda madre è chip centrale.

Le schede Intel Atom per desktop funzionano entrambe bene ma sono un po 'noiose. Ricordano le vecchie schede madri di Intel, prima che prendesse sul serio la concorrenza delle schede madri taiwanesi modificabili.

Zotac è il primo produttore a produrre una scheda madre mini-ITX basata sulla piattaforma Ion di NVIDIA. Prendi il processore Atom di Intel, abbinalo al chipset Ion di NVIDIA (che è fondamentalmente un chipset GeForce 9300) e hai la piattaforma Ion.

L'unità centrale di elaborazione

L'unità centrale di processo, la CPU, progettata nelle sue prime moderne versioni dall'italiano Federico Faggin, è caratterizzato da una architettura interna, che mostra parti specifiche che tendono a rimanere sostanti anche durante le migliorie delle versioni che si producono con gli anni.
L'architettura interna del microprocessore viene riportata nello schema a blocchi mostrato qui sotto valido per un generico microprocessore.



Descrizione

Bus: Con il termine bus si identificano le linee fisiche su cui sono trasferiti i Bit di informazione.

I bus contenuti in un microprocessore sono:

1. Un **bus interno** che collega tra loro i singoli elementi: Registri, A.L.U., Accumulatore, Unità di controllo C.U., ecc. Tutti i trasferimenti dei dati nella C.P.U. avvengono su questo BUS.
2. **Bus dati**, necessario allo scambio di informazioni con l'esterno. Il bus dati è collegato in maniera bidirezionale alla memoria e ad i dispositivi di I/O.
3. **Bus indirizzi**, necessario per selezionare le righe e le colonne della matrice di memoria.
4. **Bus di controllo**, necessario per trasmettere o ricevere segnali di controllo col mondo esterno.

La A.L.U.

Aritmetic Logic Unit, contiene la rete logica combinatoria utilizzata dal microprocessore per l'elaborazione dei dati.

La A.L.U. è un elemento privo di memoria, per cui deve essere assistita in ingresso da due registri temporanei (nel caso dei P.L.C. Siemens si chiamano **ACCU1** e **ACCU2,** nei modelli S7-200 e 400 sono 4 accessibili direttamente per i 200 o vincolatamente a ACCU 1 nei 300).

L'accumulatore sarà considerato la destinazione canonica di tutti i movimenti di dati da e per la memoria, ed anche il luogo nel quale vengono effettuate le operazioni. I dati che vanno alla A.L.U. transitano e sono memorizzati nei registri accumulatori. La A.L.U. deve essere in grado di eseguire almeno le seguenti operazioni minime: Addizione, Incremento, And, Ex-or, Shift Destro e Sinistro, Clear dell'accumulatore. Sempre sotto controllo da parte dell'A.L.U. è il registro di STATO, il quale contiene i Flag. (bit che indicano lo stato di funzionamento, quello più importante per i PLC Siemens è RLC).

Registri

Questi dispositivi memorizzano in modo temporaneo i dati da elaborare o alcuni Bit di significato speciale (flag).

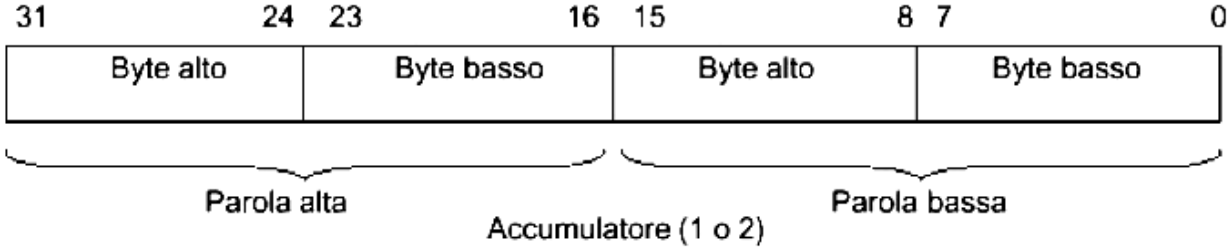
Le informazioni riposte in essi ci permangono (Latch) fino al momento di una sovrascrittura.

Registri temporanei

Si trovano all'ingresso dell'ALU e memorizzano gli operandi da elaborare.

Accumulatore 1.

Costituisce il registro principale di ogni microprocessore, sarà la parte più in uso durante lo stato di RUN di un programma. Nella serie Step7 hanno estensione 32 bit.



Ogni operazione che interessa l'Accumulatore ne cancella il contenuto precedente a causa della sovrascrittura, tuttavia per ragioni di progettazione i P.L.C. Siemens salvano il contenuto di Accu1 nel gemello Accu2 rendendo così possibile effettuare le operazioni con due operandi.

L'accumulatore può operare direttamente con la memoria o i dispositivi di I/O; Esistono infatti operazioni di trasferimento del tipo:

- ACCU -> MEM
- ACCU-> I/O
- I/O-> ACCU
- MEM-> ACCU.

Registri di indirizzo

Nei PLC S7 sono presenti due registri indirizzo AR1 e AR2, usati come puntatori per indirizzare in modo indiretto i dati. Questi funzionano in stretta collaborazione con i registri accumulatori. Se ad esempio viene definito il puntatore P#200.0 l'istruzione **L EB[AR1,P#200.0],** che carica in ACCU1 il byte d'ingresso il cui indirizzo si ottiene sommando 200 byte al contenuto di AR1.

Registri dei blocchi dati.

Quando si utilizzano operandi contenuti in un blocco di dati, il numero della DB a cui si accede viene memorizzato nel registro interno DB1. Quando si vuole accedere a più dati all'interno di una stessa DB, si può specificare solo l'indirizzo interno al blocco utilizzando il comando awl AUF che carica in DB1 il numero della DB da aprire. Esiste anche un secondo registro DB2 in cui viene memorizzato il numero della DB di istanza alla quale si sta accedendo.

Registri di uso generale A,B,C,D.

Vengono utilizzati come se fossero delle celle di memoria R.A.M., e quindi sono indirizzabili (accessibili) anche un byte alla volta. Nei moderni processori a 32 o a 64 bit si accede a Word o Double Word.
Questi registri sono molto utili per trattenere i risultati parziali di un'addizione (o operazioni in genere) contestualmente al contenuto dell'Accumulatore.

Status register. (SR),
In questo registro sono contenuti i Flags i quali hanno lo scopo di memorizzare i risultati di alcuni test eseguiti dalla A.L.U..
I flags (bandierine di segnalazione) normalmente presenti in un microprocessore sono:

C (carry), segnala se durante un'operazione algebrica si è verificato un riporto. Normalmente una segnalazione di carry corrisponde ad un errore.
Z (Zero), segnala quando il risultato di un'operazione è nullo.
N (Negative), quando il microprocessore sta lavorando in complemento a due N risulta alto.
P (Parity), questo flag è in grado di segnalare se durante un trasferimento dati (sia interno che esterno) si è verificato un errore di trasmissione.
RLC (risultato logico combinatorio) è il flag più importante per i computer di controllo industriale detti PLC (controllori a logica programmabile) perché viene testato da quasi ogni istruzione Step7 (il linguaggio di programmazione Siemens), il suo stato è imposto dal risultato booleano dell'istruzione in corso e passato a quella successiva. Lo stato di un'uscita digitale è impostato dalla copia di questo flag con il comando "=", ad esempio =A124.0 copia lo stato di RLC all'indirizzo di morsettiera A124.0 determinando se sarà attiva o spenta.
Nello specifico caso della CPU di un PLC della serie S7 lo status register pur essendo implementato in una word occupa solo 9 bit i cui nomi e posizioni sono mostrate nell'immagine sotto.

2^{15}... ...2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	BIE	A1	A0	OV	OS	OR	STA	RLC	/ER

Dato che, come di consueto, i bit sono numerati da zero, a partire dal meno significativo che si trova a destra, il flag RLC è il numero 1 dello status register. Lo stato di RLC è sia manipolato automaticamente dal PLC che impostabile direttamente dal programmatore. Ovviamente non tutti i flag hanno questa possibilità ma sono gestiti solo dal processo.
Per comprendere bene il funzionamento di RLC si deve considerare un segmento di codice, ad esempio rappresentato in KOP, (in americano Ladder), che si sviluppa dalla linea di potenza verso la bobina di carico attraversando una successione di risultati logici combinatori. Questa successione in bibliografia è nota come stringa logica. Se ad esempio analizziamo il classico segmento di auto ritenuta potremmo dire che si tratta di un OR posto in AND con lo sgancio, il cui risultato imposta (pone a 1 logico) la bobina di uscita. RLC viene impostato una prima volta sull'OR, il suo contenuto attuale viene posto in AND con lo sgancio, si ottiene un aggiornamento del flag RLC e sarà questa attuale situazione ad essere copiata nell'uscita tramite ad esempio il comando = "motore". (uguale al simbolico dell'uscita)
Facciamo una rapida carrellata del significato degli altri bit dello status register:

- bit 0 : **/ER** Indica l'inizio di un nuovo segmento, dato che ogni segmento rappresenta una cascata di condizioni booleane spesso viene definito stringa logica della quale ogni elemento agisce sul bit 1 in funzione che l'operazione booleane transitoria valga 1 oppure 0, fino ad incontrare il comando "=" che imposta un'uscita e conclude il segmento (talvolta detto Rung).
- Bit 1: **RLC**, ovvero risultato logico combinatorio. Nel caso di necessità di impostazione forzata dell'RLC da parte del programmatore si può usare il comando Step7 "S" per portarlo a 1 e "R" per portarlo a 0.
- Bit 2: **STA** - bit di stato, memorizza il valore di un booleano puntato da una istruzione, ad esempio un input digitale, non direttamente in morsettiera ma tramite la sua immagine di processo presente in memoria o un merker. Se il bit puntato è solo acquisito STA è controllato dal campo se invece è impostato, ad esempio come si fa con una uscita allora il controllo di STA passa al programmatore ma è comunque dipendente dall'immagine di processo e non dal morsetto fisico. Il bit di stato non ha significato per le operazioni logico combinatorie che non accedono alla PAA o alla PAE o genericamente in memoria.
- Bit 3: **OR**. Questo bit ha un significato solo nelle stringhe logiche in cui siano presenti operazioni di AND prima di operazioni di OR. In questo caso il bit di OR indica se una funzione AND, eseguita prima di una OR, ha fornito valore 1, anticipando così il risultato dell'operazione di OR.
- Bit 4, **OS**. è il bit di overflow con memoria. Lo si imposta a 1 contemporaneamente a quello del bit successivo OV, ma in più mantiene la memoria dell'evento anche quando questo non è più presente. Il comando AWL *SPS* (salta se OS=1) e i comandi di richiamo blocco e di fine blocco resettano a 0 il valore di OS.
- Bit 5: **OV**. Benché il nome indichi uno sforamento rispetto al numero di bit disponibili a seguito di una operazione matematica, ad esempio una somma, viene impostato anche dai confronti in virgola mobile.
- Bit 6 e 7: si chiamano **bit di visualizzazione A0 e A1**. Forniscono informazioni sull'esito di alcune operazioni. Per esempio, in base al valore di questi due bit si può conoscere se il risultato di un'operazione matematica è minore, maggiore oppure uguale a zero, oppure si può risalire al valore del bit traslato in un'operazione di shift.
- Bit 8: BIE, ripristina il valore di RLC dando la possibilità di un ritorno forzato da un blocco prima della sua fine. Utile nelle operazioni annidate.

Program Counter (PC)

Il PC contiene l'indirizzo dell'istruzione che si sta eseguendo e predispone il microprocessore all'esecuzione della prossima tramite il processo dell'auto incremento.

Considerando la memoria come una matrice, ad ogni sua riga corrisponde un valore esadecimale " indirizzo ".

Per estrarre un'istruzione o dato dalla memoria è necessario "puntare" a tale cella caricando il suo indirizzo nel PC.

La lunghezza di tale registro determina la massa di dati indirizzabili in memoria cioè la "locazione" di memoria più lontana dall'origine (estensione). Durante l'esecuzione di un programma si possono verificare due casi:

1)Il programma è una lista sequenziale di istruzioni da eseguire una dopo l'altra, è questo il caso dell'incremento automatico e progressivo del PC.
N.B. Il PC viene incrementato di un valore che dipende da come è organizzata la memoria.
Esempio: Una memoria organizzata in Byte impone al PC un' incremento di +1, una memoria organizzata in Word di 16 bit impone un' incremento del PC pari a +2.

2)Il PC viene caricato direttamente con istruzioni Software che impongono il salto ad indirizzi noti: vedi salti a subroutine o indirizzamento di Merker.

Registro indirizzi (M.A.R.)
E' questo il registro che collegato direttamente alla memoria ne consente l'accesso "indirizzamento".
Normalmente viene caricato dal contatore di programma P.C., ma è accessibile con particolari istruzioni anche dagli altri registri. Questo registro è ricopiato in due particolari registri a 32 bit delle CPU Step7, chiamati ARI1 e ARI2. Nei programmi degli utenti vengono usati come puntatori alle aree di memoria.

Blocco C.L. (CONTROL LOGIC).
E' la circuiteria di tipo logico combinatorio che coordina tutti gli eventi del microprocessore.
Questa rete logica abilita e disabilita le varie parti del microprocessore rispondendo alle esigenze del software. I bus interni sono direttamente governati dal C.L. I servizi inoltre svolti sono :

- Gestione della fase di Bootstrap attraverso un impulso di reset.
- Gestione dei dispositivi di I/O.
- Gestione degli eventi che devono essere serviti con priorità "interrupt".

Stack pointer (SP)
Questo è il registro che consente la tecnica di programmazione " a blocchi", cioè consente la chiamata a sottoprogrammi.
Lo stack è una struttura di tipo L.I.F.O. (Last In First Out) situata in una particolare area di memoria di ampiezza e posizione dipendente dal tipo di microprocessore.
Lo Stack di tipo LIFO è organizzato in modo che l'ultimo dato ad essere inserito sia necessariamente il primo ad essere letto " estratto".
Un'idea mentale della struttura ce la possiamo fare pensando ad una pila di piatti che dopo essere stati lavati sono in attesa di venire asciugati.
Durante la chiamata ad un sottoprogramma è necessario ricordarsi a che punto il programma è stato interrotto per poterlo riesumare alla fine della subroutine. Il processo di salvataggio si chiama "Commutazione del contesto ". Un blocco subroutine appare come un insieme a se stante di istruzioni allocate "memorizzate" in un punto noto della memoria e richiamabili da un'altra lista di istruzioni detta programma principale.
La subroutine "Blocco di programma " dovrà terminare con un'altra istruzione di ritorno " BE ", che nei nuovi modelli di PLC è sottointesa e quindi non sarà necessario inserirla a meno che non si tratti di una fine condizionata di un blocco ovvero una forzatura all'uscita anticipata, tale comando è il BEB, la quale dirà al microprocessore che è il momento di riesumare il vecchio contesto dalla pila di "Stack".

Durante l'operazione di recupero del contesto viene caricato nel PC l'indirizzo successivo a quello in cui si era abbandonato il programma principale per entrare nel sotto blocco.
I blocchi di programma S7 sono di tipo organizzativo detti OB, di comando/istruzione detti blocchi funzione FC, blocchi funzioni con memoria detti funzionali, e blocchi di dati che non contengono istruzioni Step7 ma solo un elenco di variabili, ad esempio di visibilità globale.
Per quanto riguarda i blocchi organizzativi oltre all'OB1 ci sono altri blocchi che vengono chiamati dal sistema operativo su eventi speciali come un guasto nel rack di montaggio (OB86), il ravviamento del controllore (OB101) oppure il superamento del tempo ciclo massimo (OB80) che funge da watch dog.
L'OB100, invece, viene richiamato solo al primo ciclo di scansione.
Altri OB, come l'OB10, vengono richiamati periodicamente ad intervalli prefissati di tempo impostabili dall'utente (*OB a tempo*).

La memoria centrale

La memoria centrale del computer è l'unità di memoria in cui sono registrati i dati durante l'elaborazione. La memoria è detta "centrale" in quanto sono presenti anche altre tipologie periferiche di memoria (ossia non centrali) in un computer la memoria centrale è un componente dell'unità di elaborazione centrale (CPU).

Nella memoria centrale del computer le informazioni sono registrate in forma binaria in gruppi di bit, detti parole, in appositi indirizzi di memoria. La lunghezza del gruppo di bit (parola) varia a seconda dell'architettura del computer (es. 8 bit,16 bit, 32 bit, 64 bit, ecc.).

La memoria centrale di un computer è composta da tre tipologie di memoria:

- Memoria RAM. La memoria RAM (Random Access Memory) è una memoria ad accesso diretto di tipo volatile che al momento dello spegnimento del computer la memoria si azzera e i dati sono perduti.
- Memoria ROM. La memoria ROM (Read Only Memory) è una memoria di sola lettura. Nella memoria ROM sono memorizzati i programmi che consentono al computer di configurare il sistema durante le operazioni di accensione.
- Memoria Cache. La memoria Cache è una memoria temporanea ad elevata velocità di accesso dei dati, molto più veloce rispetto alla memoria RAM. La memoria Cache è collegata direttamente alla CPU per svolgere più velocemente le operazioni di calcolo.

Alla domanda si può rispondere anche così, La memoria centrale, chiamata anche memoria primaria o memoria principale, è un particolare tipo di memoria informatica in grado di offrire una grande velocità di accesso ad un costo unitario però molto elevato. Il più delle volte, quando si parla di computer, la memoria centrale non indica altro che la memoria RAM dello stesso, ossia una parte fondamentale di tutto il computer in quanto, grazie alla sua capacità di immagazzinamento espressa in termini di GB, dipende la quantità massima di dati che possono essere poi prelevati ed elaborati da parte del processore, rappresentando quindi a tutti gli effetti un parametro prestazionale dell'intero computer. La caratteristica principale della memoria RAM è che si tratta di una memoria volatile, ovvero, una volta spento il computer, tutti i dati contenuti in essa vengono subito persi.

Bisogna comunque tener presente che oltre alla memoria RAM esistono anche altre tipologie di memorie primarie fra le quali quelle più importanti sono la cache (si pronuncia cash), la ROM, l'EPROM e la MRAM.

Come si misura la memoria

La memoria di un computer è sempre multiplo dell'unità fondamentale detto bit.

IL bit si ottiene come stato logico di una variabile booleana ovvero TRUE o FALSE, ma dal punto di vista elettrico si tratta di un circuito che deriva dalla commutazione di un transistor in tecnologia MOS.

Vengono costruiti semplici circuiti digitali, chiamati FlipFlop che sono in grado con un comando di set e uno di reset, di memorizzare il singolo bit.

Mettendo in serie più FLIP Flop si ottengono i Registri di memoria, con varie tecniche per leggerli e scriverli, ad esempio PIPO = parallel input parallel output, o SIPO serial input parallel output. Ecc.

I bit sono organizzati in gruppi di 8 detti Byte.

Un byte è quindi una locazione di memoria RAM o ROM, come anche FLASH nel caso dei drive USB.

In sostanza la memoria è una grande matrice con otto colonne, i byte, il cui bit meno significativo, b0, si trova a destra LSB (less significant bit), mentre quello più significativo, MSB (most significant bit) detto anche più pesante, si trova a sinistra, in sostanza ogni byte si rappresenta così: $b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0$.

Per rappresentare i dati si usano i multipli del byte sempre usando le potenze di 2 che è la base numerica in cui lavora il computer.

$2^0=1$, $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$, $2^6=64$, $2^7=128$, ecc ecc

Usando questa tecnica si capisce che l'espressione 1kB non può rappresentare 1000 byte perché 1000 non è potenza di 2, (non esiste un numero a cui elevare 2 per ottenere 1000).

Di conseguenza **1kB** è il valore più vicino ovvero **1024Byte**. -> $2^{10} = 1024$

Poi si passa alla moltiplicazione, passando alla potenza 20, per ottenere la successiva grandezza fondamentale, ovvero il megabyte.

1 Mbyte = 2^{20} = 1048576 Byte

Le successive grandezze utilizzabili in informatica sono il GigaByte (2^{30}=1073741824 Byte), il Tera Byte pari a 2^{40} = 1099511627776.

Tra poco potremmo vedere i computer in grado di mappare memoria di estensioni del Peta Byte, pari al 2^{50} byte.

La memoria di massa

La memoria di massa, chiamata anche memoria secondaria o memoria ausiliaria, è un particolare tipo di memoria informatica che non possiede una grande velocità di accesso ma che però ha un costo unitario decisamente più basso. Viene detta di massa poiché, rispetto alla memoria primaria, è in grado di raccogliere enormi quantità di dati in maniera del tutto permanente, o perlomeno fino a quando lo stabilisce l'utente.

Di solito, quando si parla di memoria di massa, quasi sempre ci si riferisce al disco fisso del computer, che ha la capacità di poter archiviare migliaia se non decine di migliaia di GB, ma in generale una memoria di massa può pure avere la forma di floppy disk, CD, DVD, Blu-ray, nastri magnetici o anche quella di memorie flash tipo le chiavette USB oppure tipo le moderne unità a stato solido conosciute più formalmente come SSD.

Le periferiche: input, output

I dispositivi atti all'inserimento e al prelievo di dati dal computer, indicati con l'acronimo I/O, sono abbastanza standardizzati e sostanzialmente sono:

- La tastiera
- Il mouse
- Il touch pad
- Lo scanner
- La penna ottica
- Joystick
- Barcode scanner
- Fotocamera o webcam
- Microfono

Specifiche per l'output:

- Il monitor
- La stampante
- Il plotter
- Le casse acustiche

Periferiche bidirezionali:

- Gli access point WiFi
- Il materizzatore
- Le membrane tattili di touchscreen
- Le memorie di massa quali i NAS, i dischi USB, le penne USB ecc

Sistema operativo

Il sistema operativo è un uno strumento software che gestisce, lavorando in background rispetto alle applicazioni, le risorse hardware e software del computer.

Il sistema operativo e lanciato in esecuzione da un sistema firmware, il BIOS (basic input output system), che inizializza il sistema, assegnando le aree di memoria e i livelli di interrupt ai driver dei vari dispositivi e periferiche.

È compito del sistema operativo raccogliere e mettere in esecuzione i driver, quei programmi che descrivono le periferiche ottimizzandone il funzionamento.

Le cose più importanti che il il sistema operativo gestisce sono le unità di memorizzazione di massa ,

Riconoscere e utilizzare le funzioni di base di un Sistema operativo
Le funzioni di base del sistema operativo solo la supervisione e il monitoraggio delle applicazioni in esecuzione facendole interagire con l'hardware.

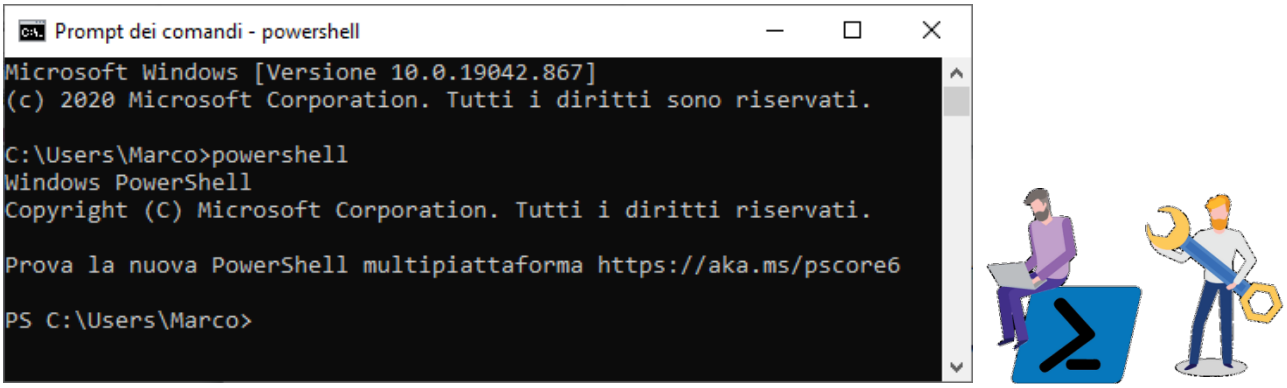
Elencando le funzioni di base sono:

1. Controllo dell'esecuzione dei programmi applicativi con assegnazione delle risorse hardware. Permettere la condivisione delle risorse, regolandone l'accesso da parte di utenti/programmi diversi.
2. Gestione delle periferiche e dei canali di interrupt oltre che aree di memoria RAM in cui mettere in esecuzione i programmi applicativi.
3. Assegna ordine e priorità alle applicazioni.
4. Gestisce il set di istruzioni firmware implementate nel processore, ad esempio le specifiche direct X o MMX che erano in uso per la grafica prima dell'avvento dei processori GPU assegnati a queste funzioni nelle moderne schede grafiche.
5. Crea un'interfaccia grafica agevole per l'utente che aiuta nello spostamento, creazione cancellazione dei file. Mostra i punti di accesso per attivare le applicazioni, mostra pannelli di controllo utili per aggiungere o rimuovere applicazione dal registro di sistema.

Interagire con gli elementi del Sistema Operativo

L'iterazione con il sistema operativo avviene su due fronti, il livello utente e il livello macchina e con due modalità, la più usata l'interfaccia grafica che risulta essere molto user friendly, oppure da console testuale, o modalità esperta, orientata più agli amministratori di rete e ai sistemisti.

Per la modalità console, oggi è disponibile un set di comandi potenziato che offre una shell accessibile tramite il comando "powershell", impartito dal prompt dei comandi.

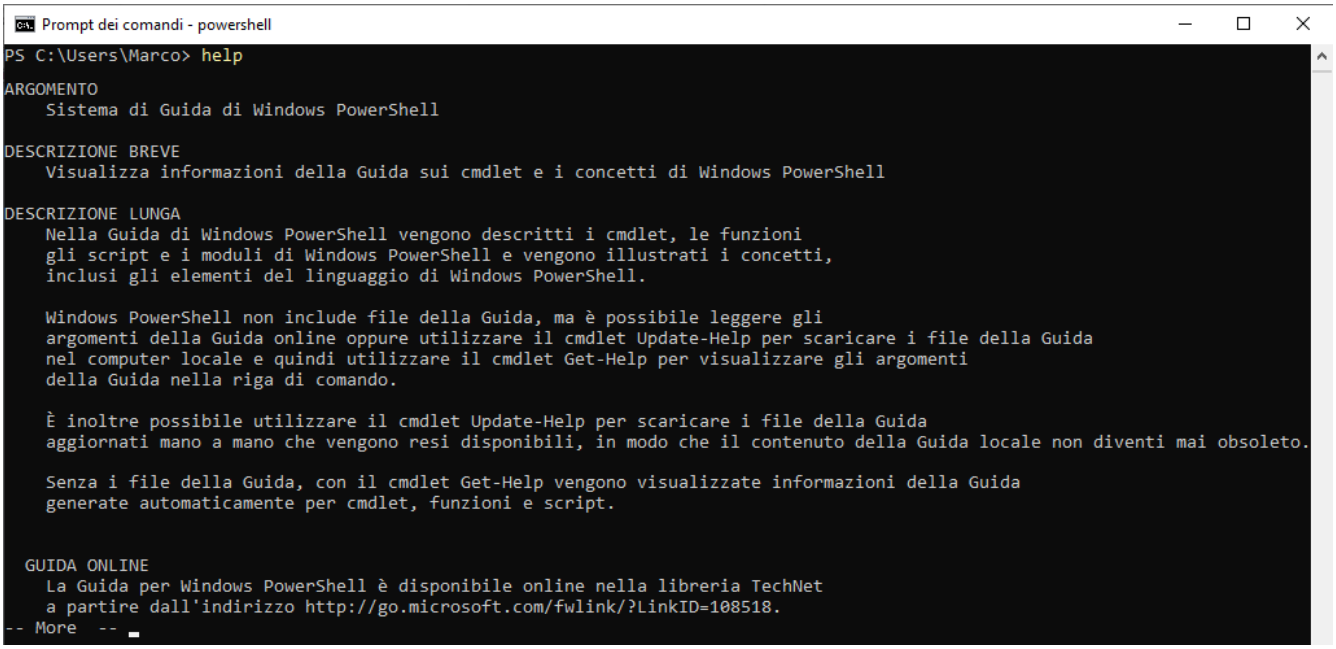


Il prompt dei comandi, cambia da C: a PS C:, indicando che ci troviamo all'interno di powershell

Powershell è una console di comando testuale, basata su tecnologia .net, richiamabile fin da windows 7, da console di comando "cmd" o "prompt dei comandi" digitando:

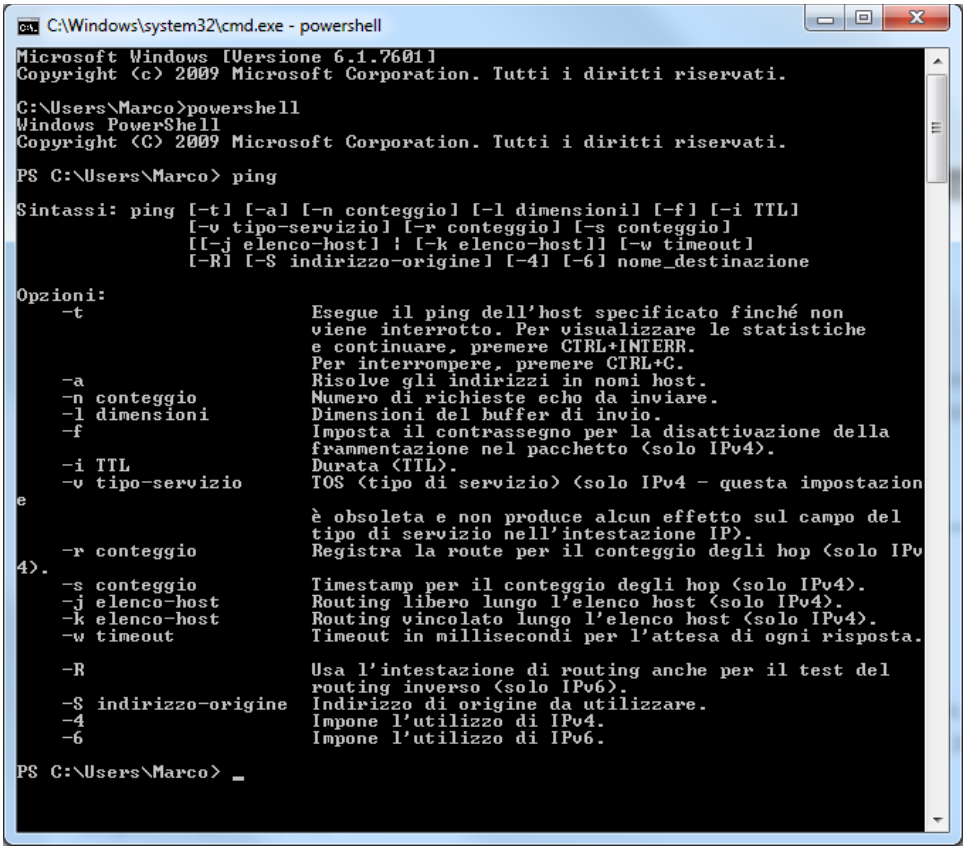
l'uscita dalla console avviene digitando "**exit**".

I comandi di powershell sono detti "**cmdlet**", e sono orientati all'amministrazione del sistema oppure alla creazione di script da eseguire sul sistema host.



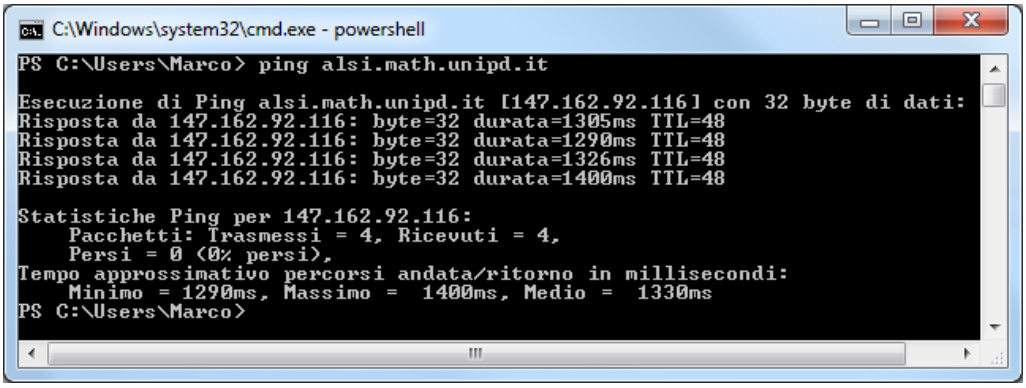
I classici comandi DOS sono operativi, ma ovviamente se ne aggiungono molti altri con finalità amministrative e di scripting.

A titolo di esempio per i comandi classici digitiamo "**ping**" senza un'IP di destinazione, la console ci restituisce un help che ci informa sull'utilizzo e la parametrizzazione del comando:

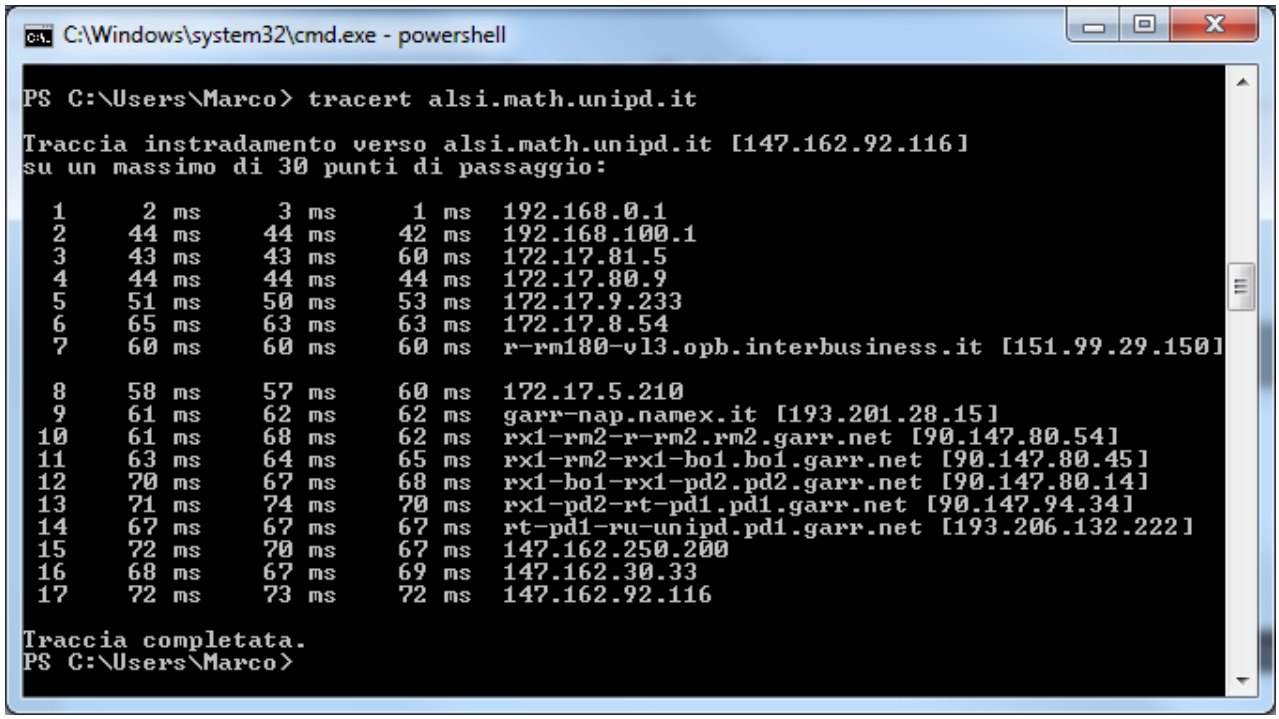


Digitiamo "clear" per pulire lo schermo, e successivamente il comando di verifica della funzionalità: ping alsi.math.unipd.it seguito da invio

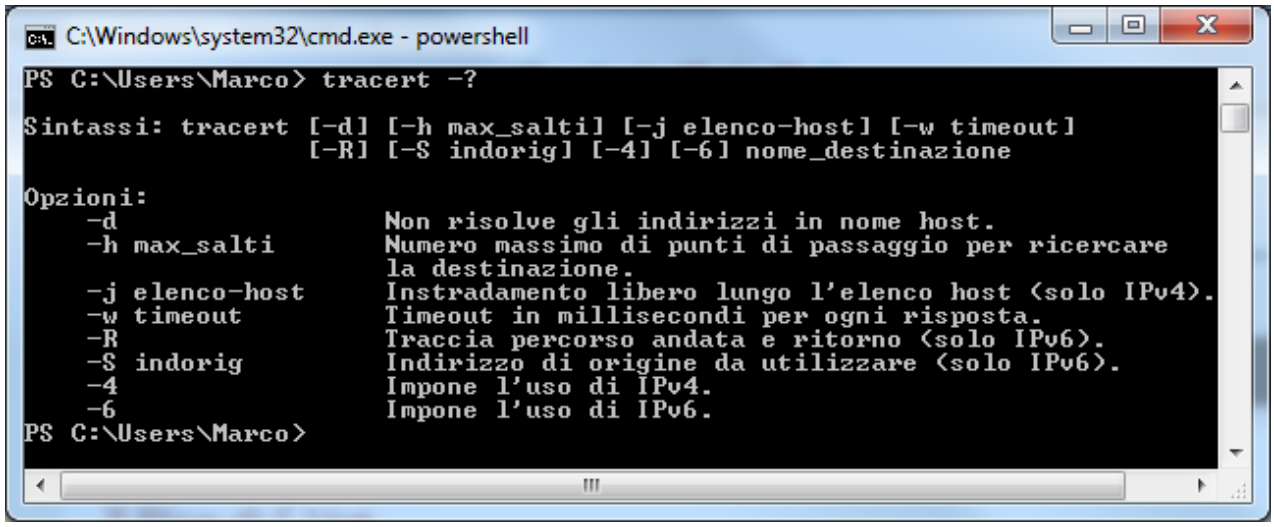
Vedremo eseguire l'invio dei quattro classici pacchetti verso l'IP della destinazione il quale ci risponde identificandosi numericamente.



Visto che il comando “ping” risulta funzionante tentiamo la tracciatura di una rotta verso una destinazione tramite il comando “tracert”, che ci mostrerà non solo l’instradamento dei pacchetti dalla posizione in cui è avvenuto il lancio, ma anche il passaggio attraverso gli eventuali router presente nel percorso. Nell’immagine seguente l’effetto del comando:

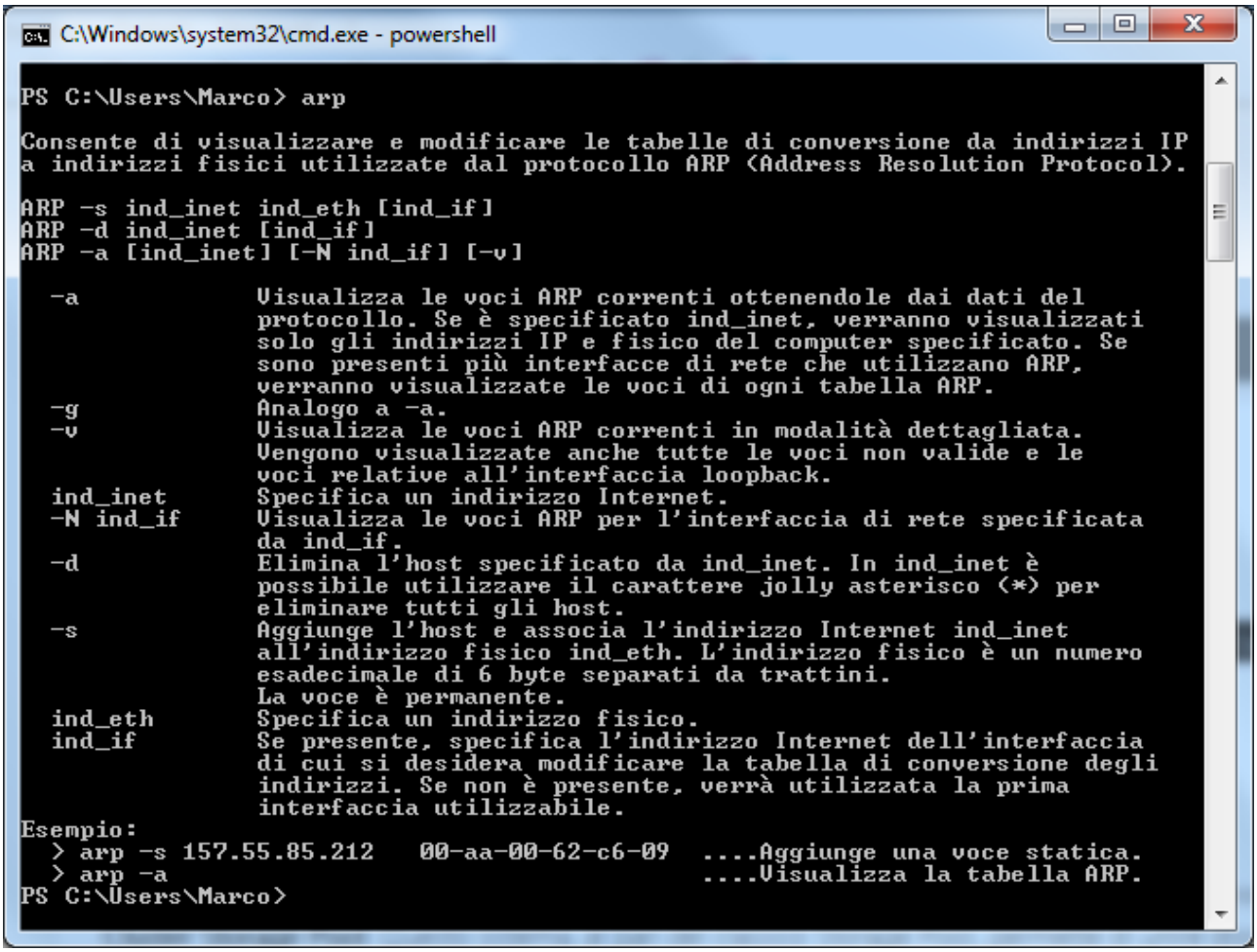


Tutte le informazioni sull’uso del comando si ricavano digitando lo switch “-?”

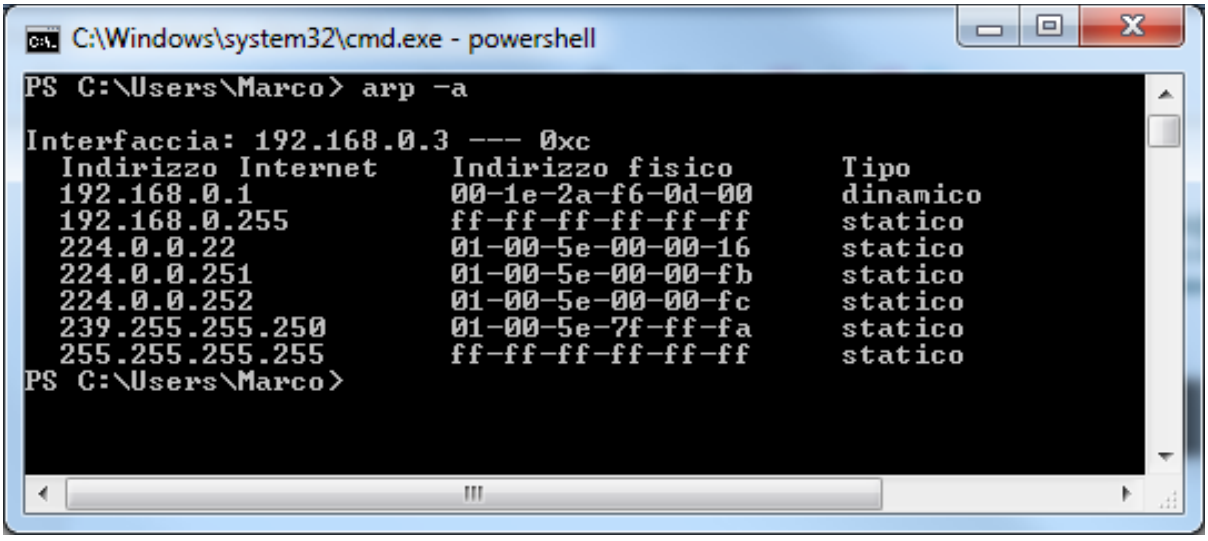


Vediamo cos’è il comando “**arp**” e come utilizzarlo.

Digitiamolo al prompt nella console di powershell, e diamo invio.



Leggendo i contenuti ci accorgiamo che lo switch corretto per vedere le voci arp correnti della macchina host bisogna usare lo switc "-a".

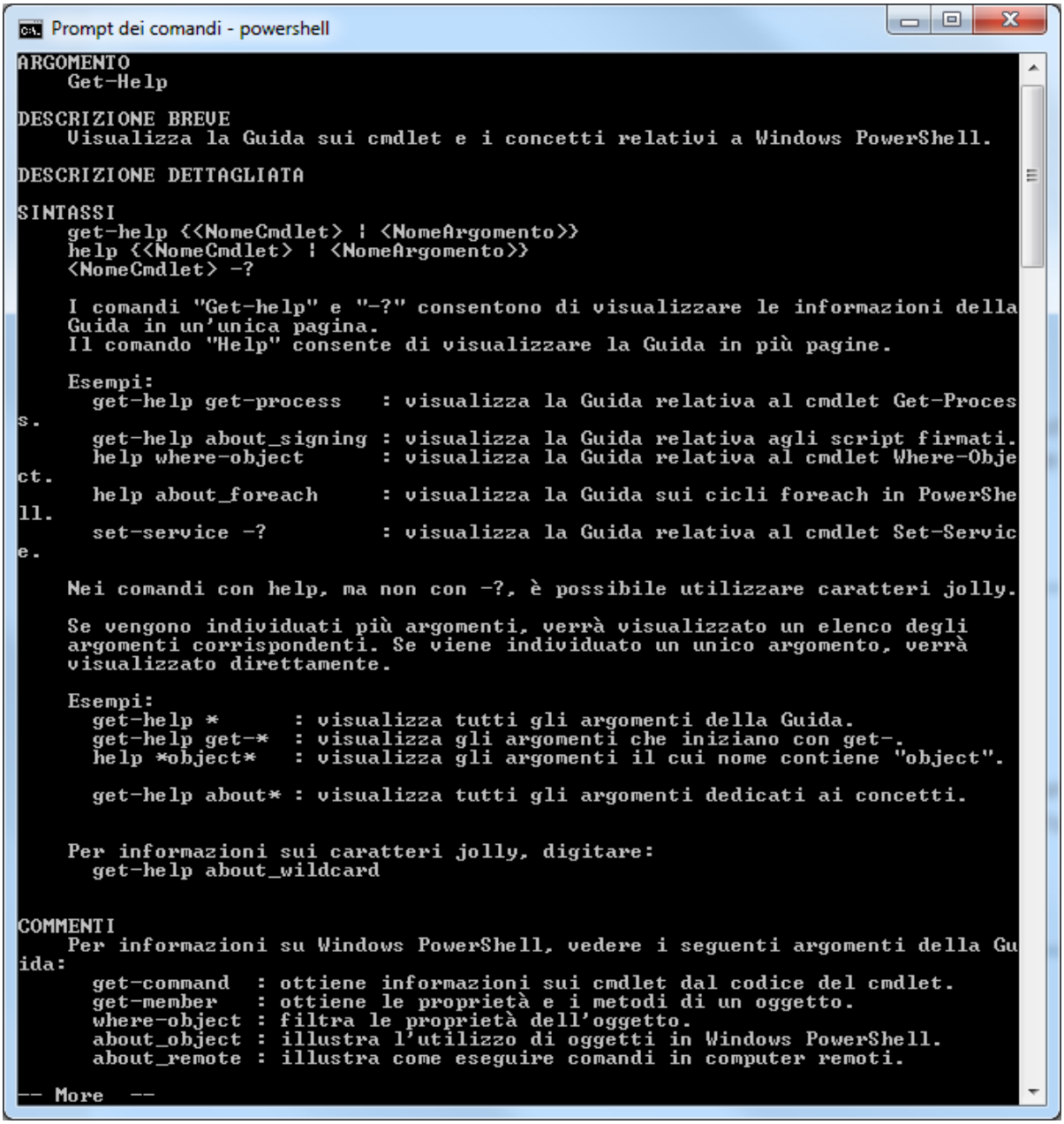


Ecco cosa è visto, come porte attive per la comunicazione dalla mia macchina al momento del lancio. Si noti la presenza degli indirizzi riservati per il broadcast e il networking. Ovviamente questo comando assume

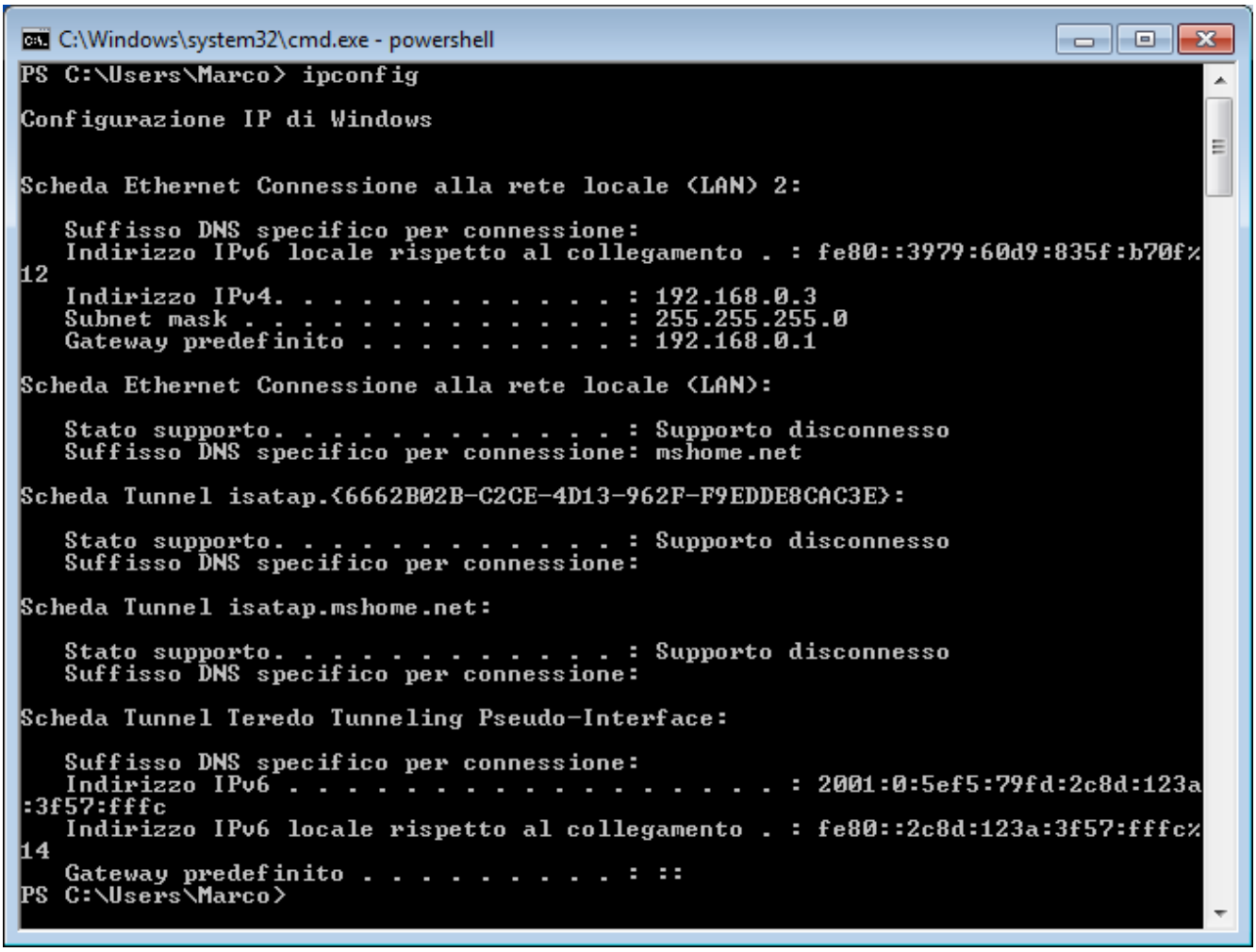
più significato in un server o in un computer con più schede di rete a cui è stato dato il ruolo di proxy o analogo filtro IP.

Digitiamo “help” per vedere una prima nota informativa sull’uso sintattico della console di power shell.

L’interruzione della visualizzazione delle righe successive avviene usando in maniera classica i tasti “Ctrl+C”, e si viene riportati al prompt di power shell.



Per la configurazione attuale dell’IP o degli IP della macchina, digitiamo il cmdlet “ipconfig”, otteniamo la risposta:



Che mostrerà molte informazioni, a parte le canoniche dello stato della macchina corrente. Ma il comando “ipconfig “ è molto più potente di quanto visualizzato sopra. Chiediamo i suoi switch digitando:

ipconfig -?

Otteremo informazioni sulla sintassi, e su come impostare switch complessi ed eventuali pipeline del comando.

Vedremo ad esempio come rilasciare un IP ottenuto in DHCP e fare richiesta di un nuovo IP dinamico senza dovere riavviare la macchina.

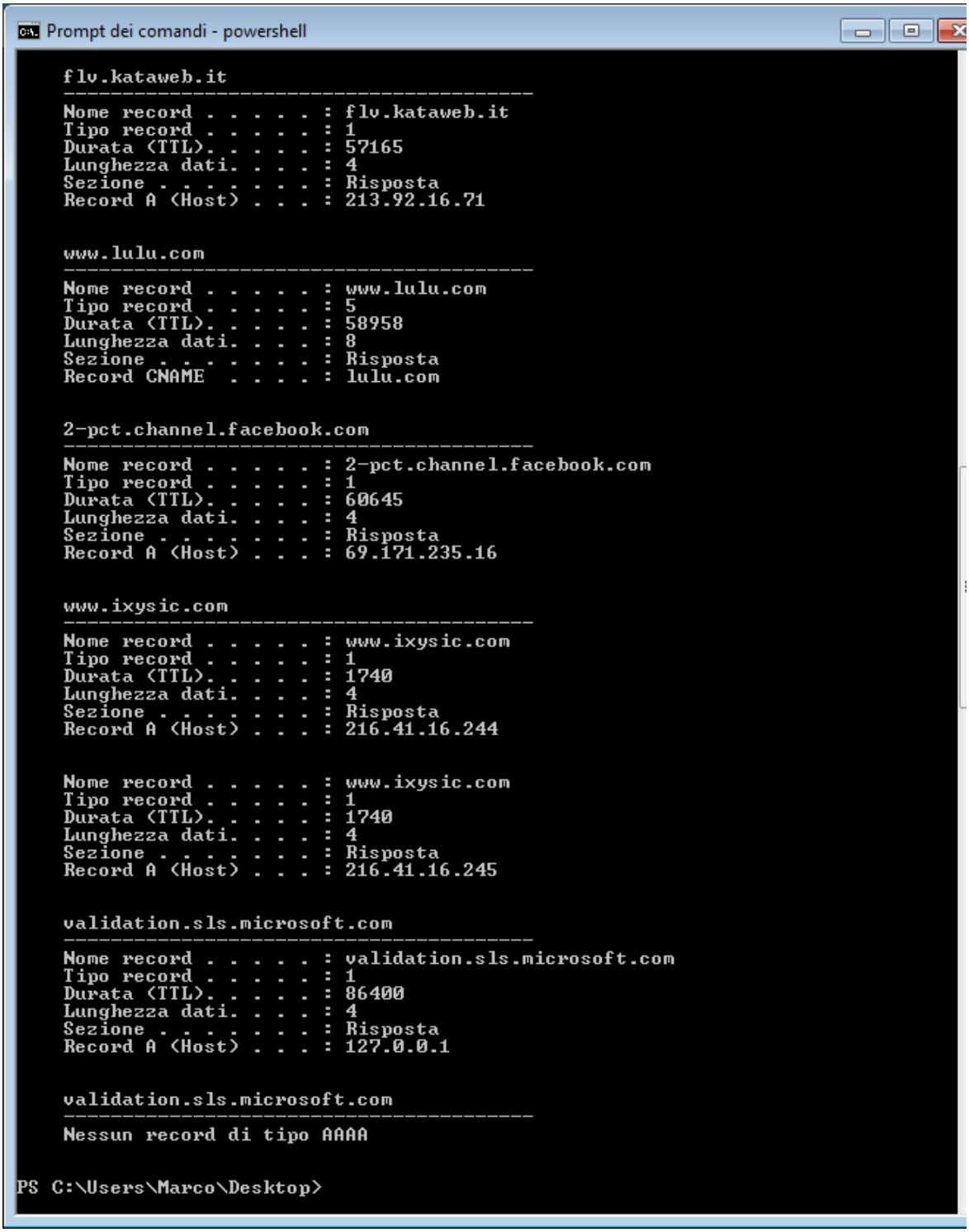
Nella scheda verranno indicate anche le modalità di richiesta di alcuni esempi di utilizzo degli switch di questo cmdlet.

Si nota che il comando ipconfig /release rilascia un eventuale IP v4 assegnato ed attivo in questa macchina, mentre il comando antagonista ipconfig /renew lo riassegna. Esiste la versione per IP V6 come chiaramente visibile nell'immagine.

Eseguiamo il comando ipconfig /displaydns per vedere la tabella di risoluzione dei nomi attiva. Nella maggioranza dei casi avremmo anche l'IP dell'host remoto che fornisce il servizio web.

Questo comando è molto utile alle aziende che lavorano nel ramo del controllo e sicurezza, dato che in un unica ordinata tabella possiamo vedere gli IP e i nomi host ti tutte le telecamere IP visualizzate sull'applicativo aperto. L'esempio dell'immagine successiva è limitato alla visualizzazione di due sezioni di internet explore, aperte in modalità multitab, ovvero con più pagine di navigazione aperte.

Dando il comando “ipconfig /displaydns” vediamo:



Avendo un problema di rete gli strumenti più immediati per la diagnostica e per iniziare la risoluzione sono ipconfig, ping, tracert. Se ad esempio il browser web ci da problemi di connessione e visualizzazione delle pagine, quindi ci sono problemi nel protocollo http grazie a questi tool infatti si può venire a conoscenza del livello a cui il problema è situato e risolverlo agevolmente.

Durante la diagnosi di un problema di rete è utile prima di tutto svuotare la cache del resolver DNS.

Il programma ipconfig offre alcune interessanti funzionalità per configurare e ottenere informazioni sulle schede di rete disponibili. Una funzione particolarmente utile nel caso si verifichino temporanei problemi con la rete è il parametro flushdns. Il comando da digitare sulla riga di comando sarà dunque:

ipconfig –flushdns

Grazie ad esso la cache del resolver DNS, ovvero la lista dei domini (es. www.gtronic.it) di recente trasformati in indirizzi IP (es. 84.78.159.158, ovvero la forma grazie alla quale le richieste sono inoltrate nella rete), sarà cancellata. In ogni caso ciò che conta è che l'utilizzo di questo comando può essere utile nel caso ad esempio sia stata interrotta per un breve lasso di tempo una connessione wireless (senza-fili), in quanto esegue una sorta di reset sulla connessione.

Fatto questo saremo sicuri di non avere in memoria alcuna traccia di possibili problemi precedenti; a questo punto si può dunque tentare di raggiungere un PC nella rete locale LAN (nel caso il problema si verifichi con un collegamento diretto ad Internet eseguire quanto detto di seguito verso un dominio come www.google.com); per fare ciò è sufficiente utilizzare il comando ping seguito dal nome del computer o dal suo IP, ad esempio "ping pc_camera" o "ping 192.168.0.1"; ping tenterà di inviare alcuni pacchetti (ovvero un piccolo quantitativo dati) di prova verso il computer in questione, il quale, nel caso in cui tutto funzioni correttamente, dovrebbe rispedirne una copia identica.

Vediamo ora alcuni possibili risultati che il comando può dare:

Nel caso il risultato sia un messaggio del tipo "Impossibile trovare l'host..." o "Impossibile risolvere il nome dell'host..." significa che la rete non funziona in maniera corretta e che non è possibile effettuare alcuna comunicazione (può darsi dunque che il filo di connessione non sia collegato correttamente, o che la connessione senza fili non abbia segnale sufficiente); in alternativa potrebbe darsi che non sia stato impostato in maniera appropriata l'indirizzo IP del resolver DNS e sarà dunque necessario cambiarlo sul valore che viene in genere fornito dal provider Internet o nel caso il problema si verifichi con una connessione a Internet condivisa da un PC in rete provare ad impostarlo su 192.168.0.1; per cambiare l'indirizzo del resolver DNS andare al menu Start, Impostazioni, Pannello di controllo, Connessioni di rete, fare click destro sulla connessione locale in questione e selezionare Proprietà, nella scheda Generale tra i componenti utilizzati dalla scheda individuare e selezionare Protocollo Internet (TCP/IP), premere su Proprietà, attivare la casella Utilizza i seguenti indirizzi server DNS e quindi impostare il valore desiderato.

se come risultato si ha invece una serie di "Richiesta scaduta" significa che è stato possibile raggiungere il computer ma non è stata ricevuta alcuna risposta, questo potrebbe significare che ci sono dei problemi con la connessione (in particolare se solo parte dei messaggi corrispondono all'errore sopra citato, mentre altri danno un riscontro positivo del tipo Risposta da ###.###.###.###: byte=32 durata=90ms TTL=244) e per questo il pacchetto arriva incompleto; un altro motivo per cui potrebbe non essere possibile ricevere risposta è che il computer di destinazione sia occupato (ad esempio un sito potrebbe essere in sovraccarico oppure bloccato).

Se al contrario tutti i pacchetti vengono ricevuti in maniera corretta significa che non vi sono problemi di comunicazione all'interno della rete locale e dunque il problema è probabilmente ad un livello successivo.

Nell'ultimo caso, si deve provare a eseguire nuovamente ping verso un server su Internet, ad esempio "ping www.google.com" effettuando una diagnosi identica a quella appena fatta.

Se si hanno problemi solo con un particolare server si può allora provare a usare ping verso di esso e vedere i tempi: se sono elevati e magari vi è persino qualche "Richiesta scaduta" (sottointeso che questo non succede con altri siti) allora si può provare un altro comando che permette di individuare il punto in cui si verifica il rallentamento o addirittura la perdita dei pacchetti da noi inviati. Si tratta del comando tracert. Ipotizziamo di avere problemi a raggiungere www.xyz.com, dovremo quindi digitare da linea di comando tracert www.xyz.com; esso fornirà una lista dettagliata di tutti i punti per cui la nostra richiesta passerà, ad esempio se il computer di destinazione si trova in Brasile si potranno vedere indirizzi dapprima contenti il nome del nostro provider di Internet, vari altri punti di passaggio fino al fornitore del servizio Internet del sito brasiliano, quindi il sito stesso; se durante questa operazione si vede che vi sono tempi molto elevati e sproporzionati rispetto agli altri significa che il pacchetto, significa che quando il pacchetto raggiunge quel dato punto viene perso; sarà quindi opportuno farlo notare al gestore Internet.

Se ancora il problema persiste, molto probabilmente il computer di destinazione ha dei problemi o semplicemente non esiste in quanto è stato digitato scorrettamente.

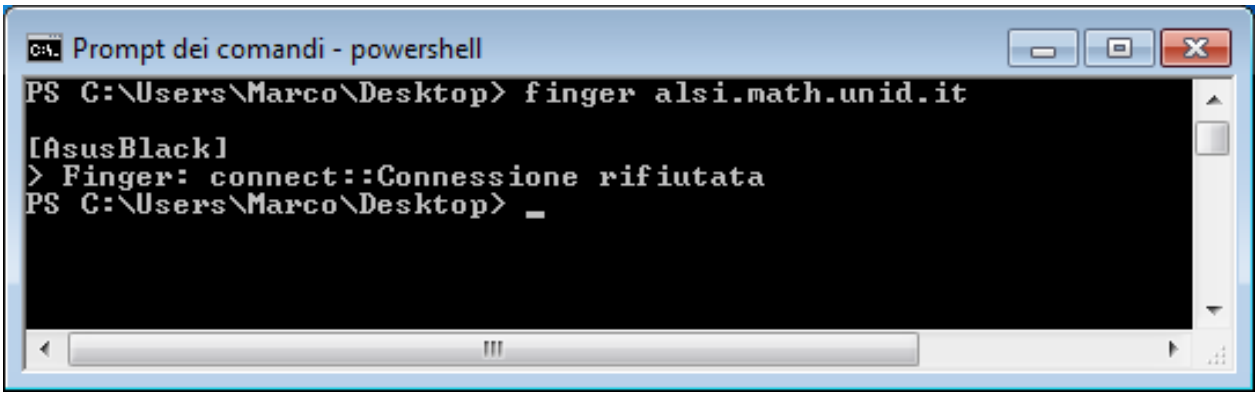
Il comando “finger” mostra, relativamente a un server remoto, quali terminali sono attivi e connessi. Consideriamo il mio server, acceso all’università di Padova, e che gestisce il solo sito web dell’associazione degli studenti lavoratori di ingegneria.

Potrò eseguire il finger indifferentemente sul suo nome host in web, o sul suo IP.

Ad esempio lanciando da powershell il cmdlet

finger alsi.math.unipd.it

Ecco la risposta attuale:

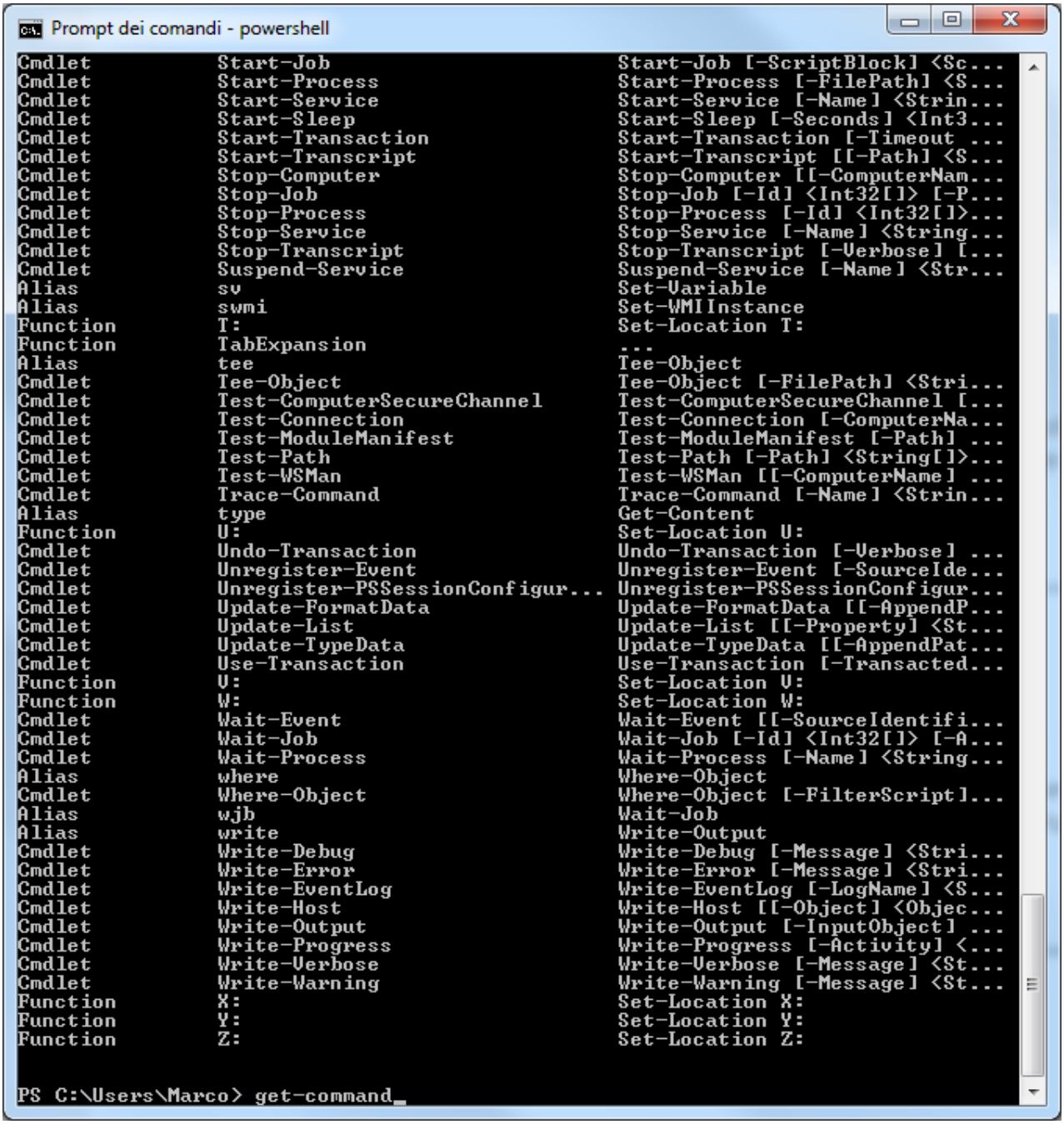


La connessione è stata rifiutata dal sistema operativo perché il servizio telnet è stato disabilitato, infatti:

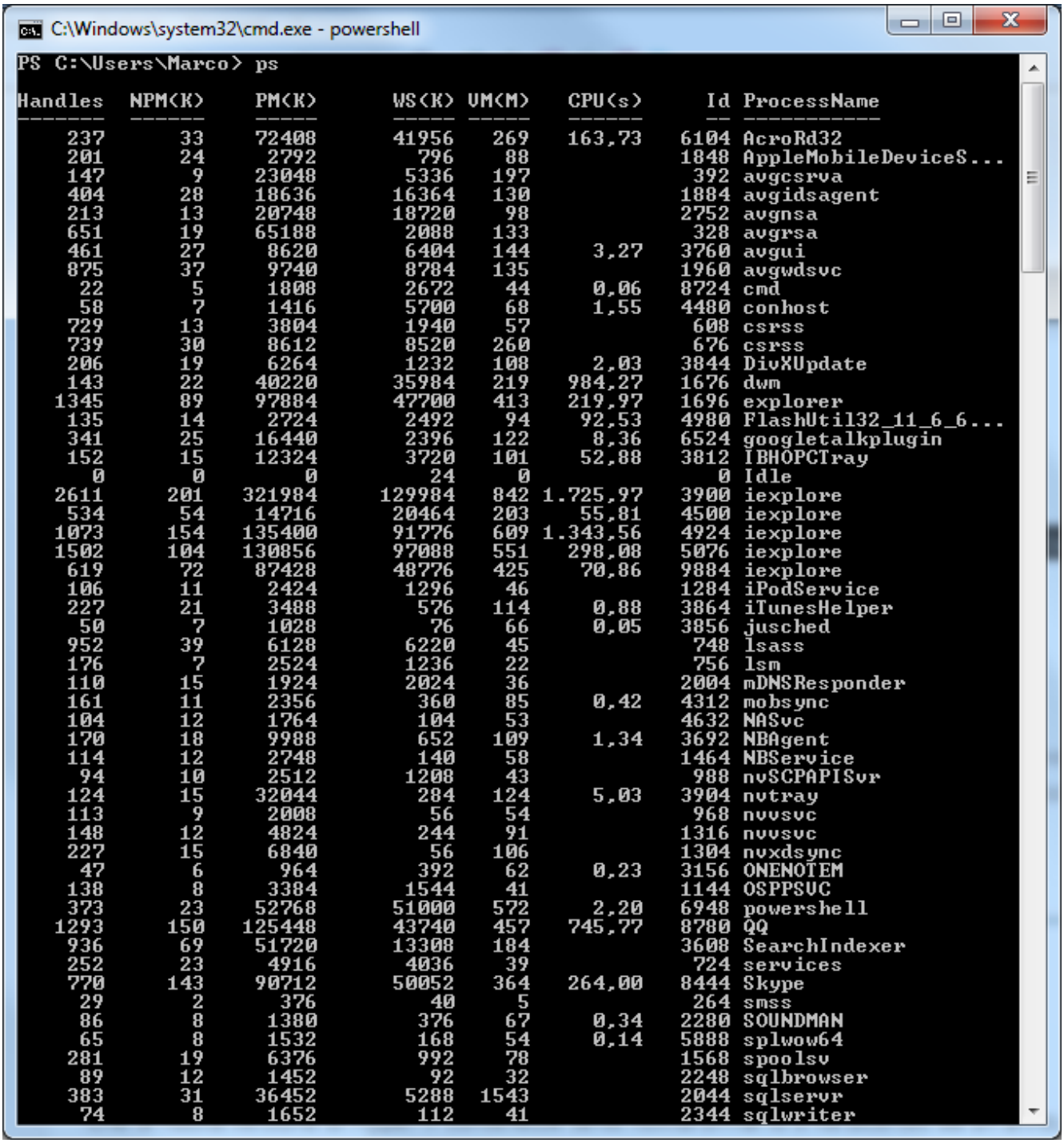
Per ottenere la lista completa dei comandi “cmdlet” digitare

Get-command

Verrà visualizzata una lunga lista di comandi cmdlet, di alias per essi, e di funzioni.

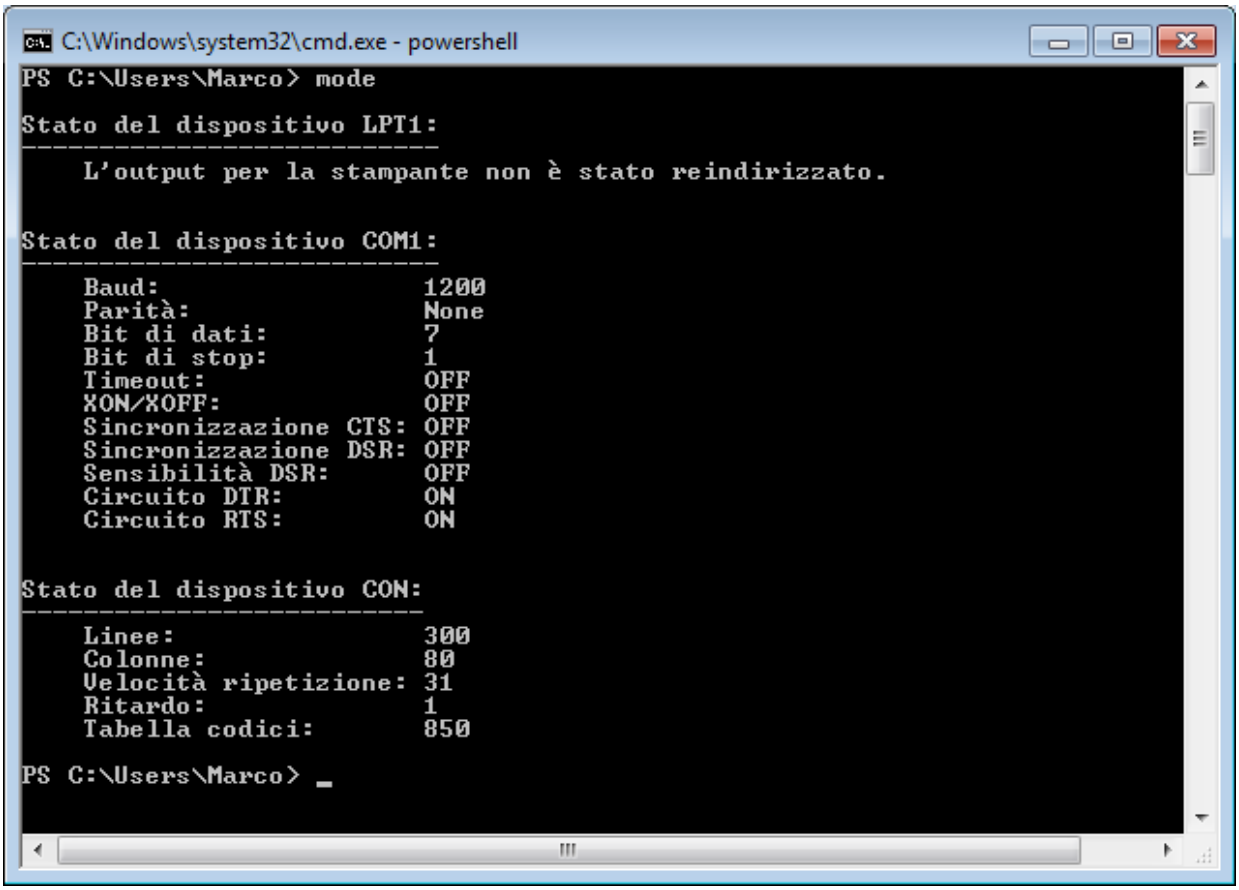


Possiamo ad esempio richiedere la lista dei processi attivi nel sistema digitando il comando “ps” seguito da invio:

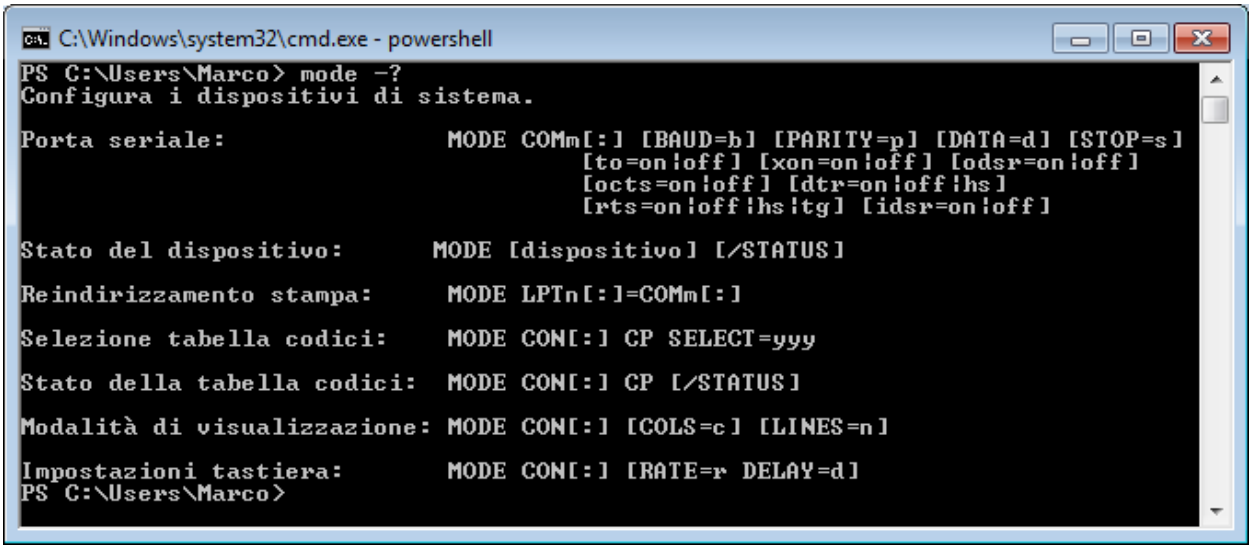


Il comado “kill” seguito dal numero di ID del processo, ovvero l’ultima colonna numerica a destra prima del nome, comporta la chiusura del processo.

Il comando “mode” mostra le porte di comunicazione attive sul PC in uso e la loro configurazione. Ad esempio, in questo PC:

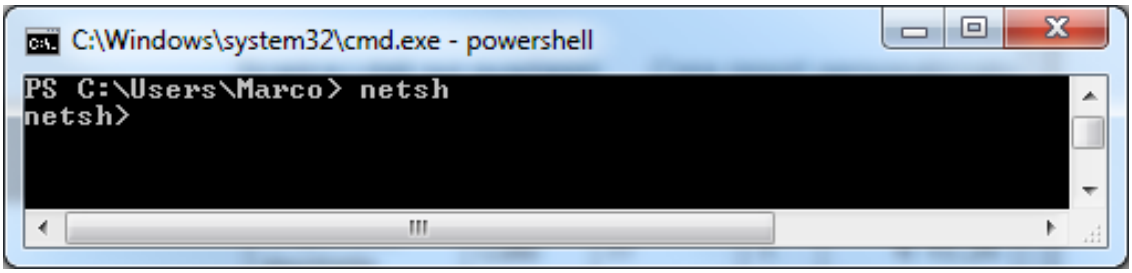


Per avere informazioni su come agire nelle impostazione di queste porte possiamo digitare "mode -?" e ottenere questa risposta:

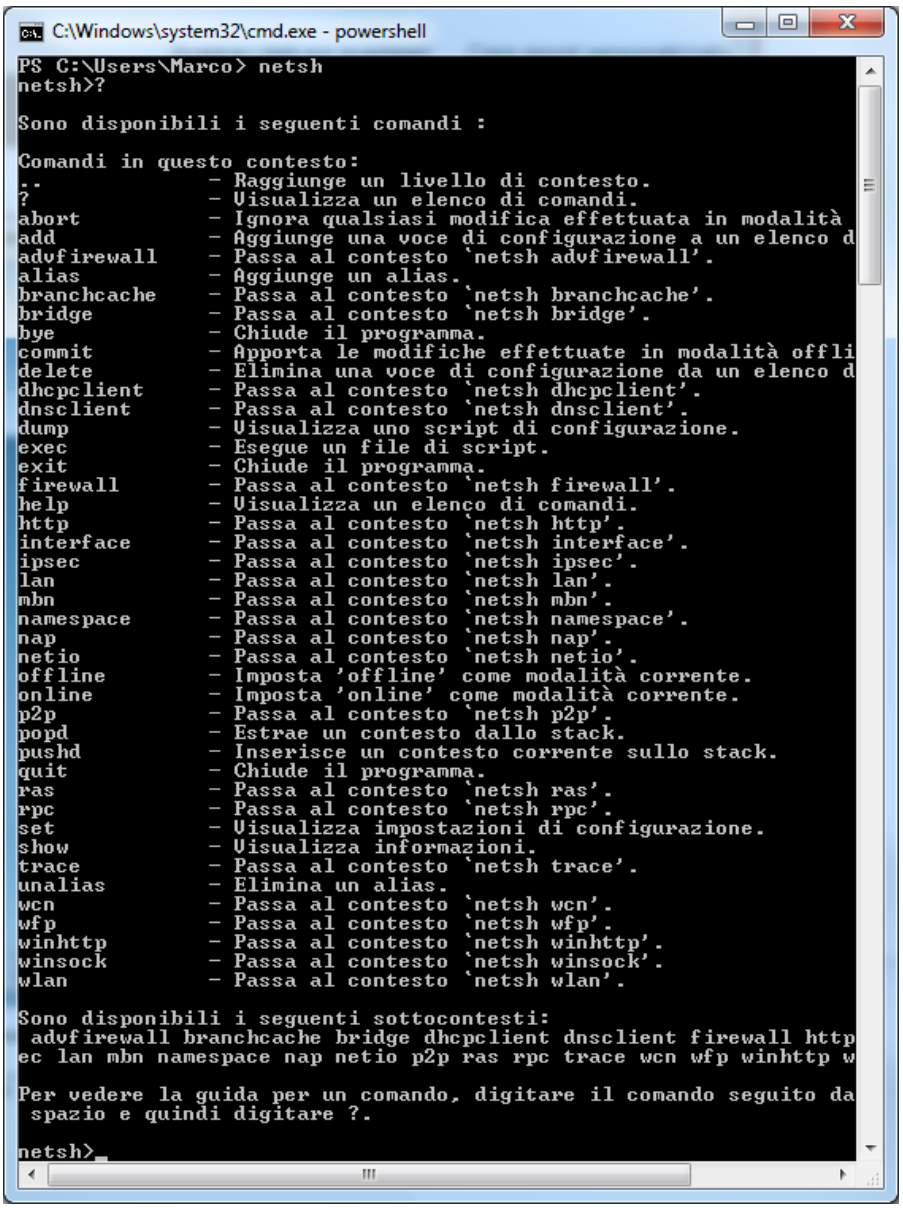


Nel sito di Microsoft si viene informati che nelle versioni future di Windows potranno essere rimosse le funzionali netsh per BranchCache, e quindi consigliano fortemente l'istallazione della powershell.

I comandi netsh di BranchCache sono fortemente usati nella precedente versione di windows server, ovvero il 2008 e nella attuale Windows server 2019.
Vediamo quali sono: Lanciamo il comando, nella consol powersell “Netsh”, vedremo cambiare il prompt come in figura.



Successivamente chiediamo un’estensione help tramite il punto di domanda “?”.



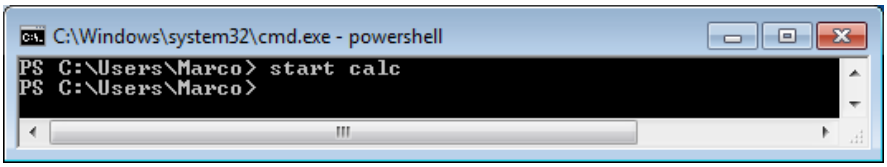
la tecnica detta pipeline

Vediamo alcuni comandi pensati per lanciare delle applicazioni ed eventualmente configurarle ad esempio passandogli un file con **la tecnica detta pipeline**. Quest'ultima consiste in concatenare tra loro più comandi. Un banale esempio è quello di lanciare una applicazione e passargli successivamente una risorsa da usare come una sorta di plugin. Possiamo simulare la cosa semplicemente lanciando prima il paint e successivamente passandogli un file da aprire. In questo momento non consideriamo che windows associa già le applicazioni ai file, quindi lanciando semplicemente il file si aprirebbe il paint, ma non è il nostro scopo.

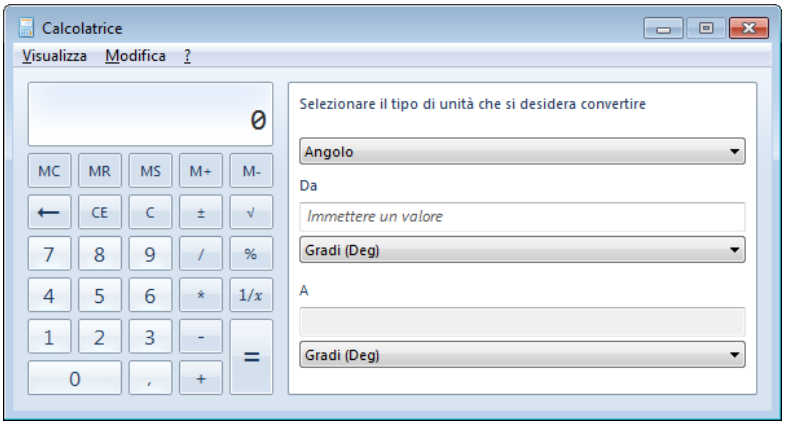
Le applicazioni le lanciamo con il comando "start".

Facciamo una prova preliminare.

start <file eseguibile> lancia un processo/applicazione, proviamo ad esempio con "calc"

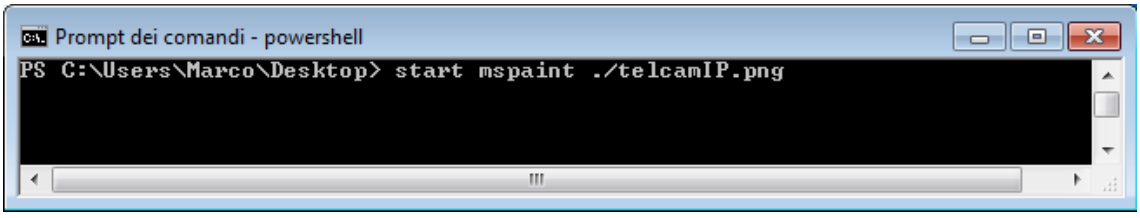


Viene lanciata la calcolatrice:

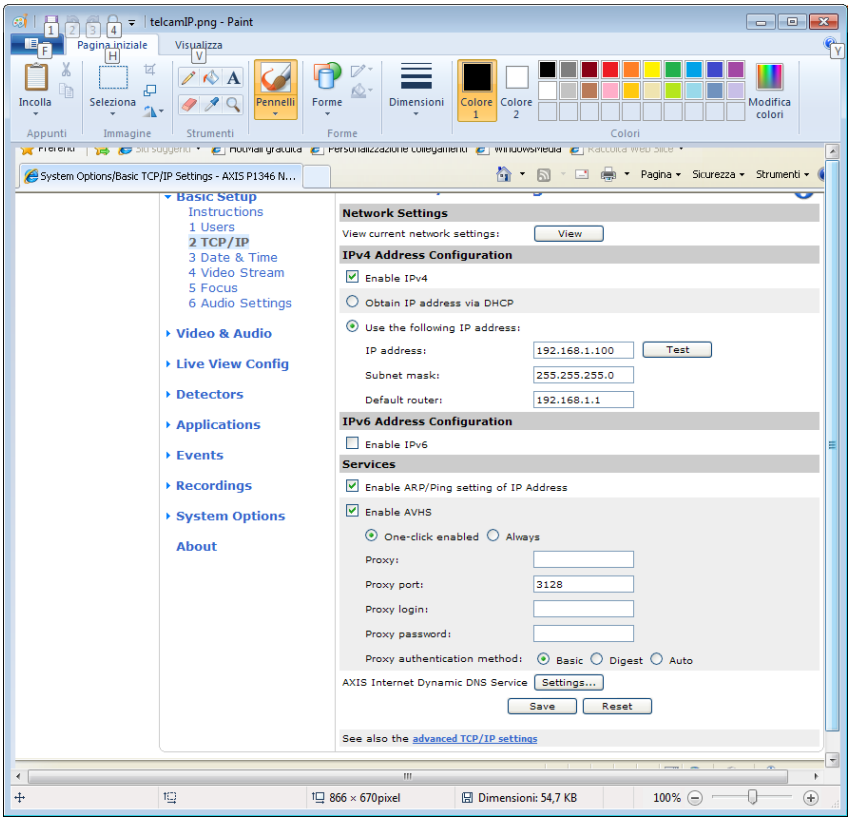


Quindi il comando è operativo, possiamo allora provare il pipeline, aprendo il programma paint e passandogli il file da aprire. Allo scopo predisponiamo un file immagine sul desktop. Apriremo l'immagine telcamIP.png

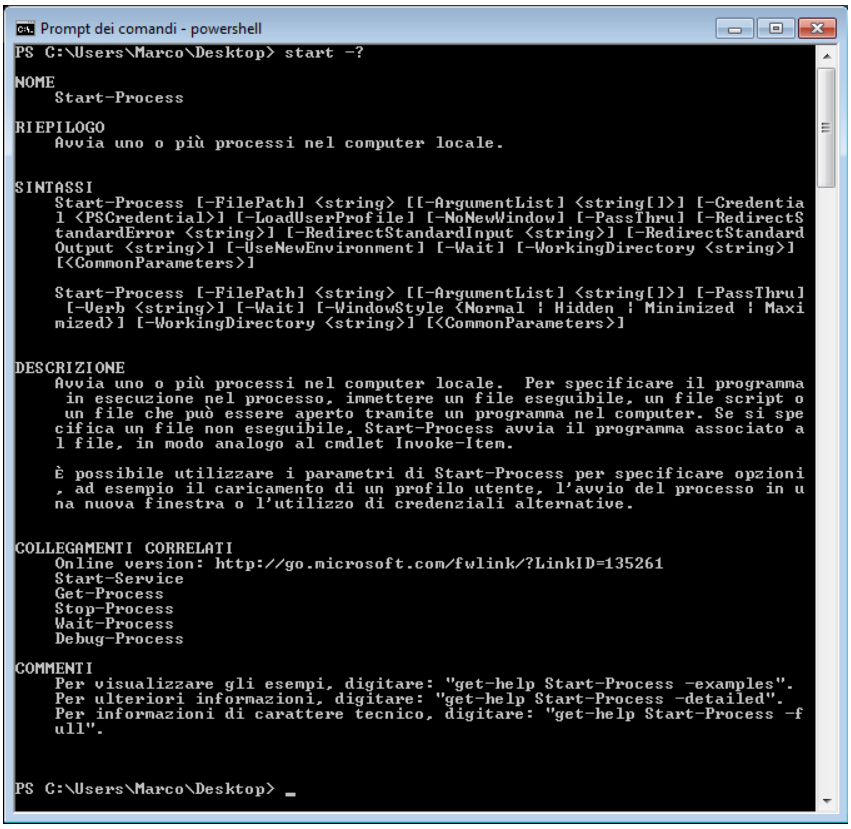
È importante fare attenzione alle path, intese come percorso in cui si trova nell'albero delle cartelle all'interno del file system.



La presenza del puntino prima dello slach "./" indica al sistema di effettuare la ricerca dalla radice fino alla posizione corrente, è così possibile non scrivere tutta la path. All'invio il comando viene eseguito e dentro al paint viene visualizzata l'immagine.



Possiamo chiedere degli info sul comando start facendolo seguire dallo switch "-?"

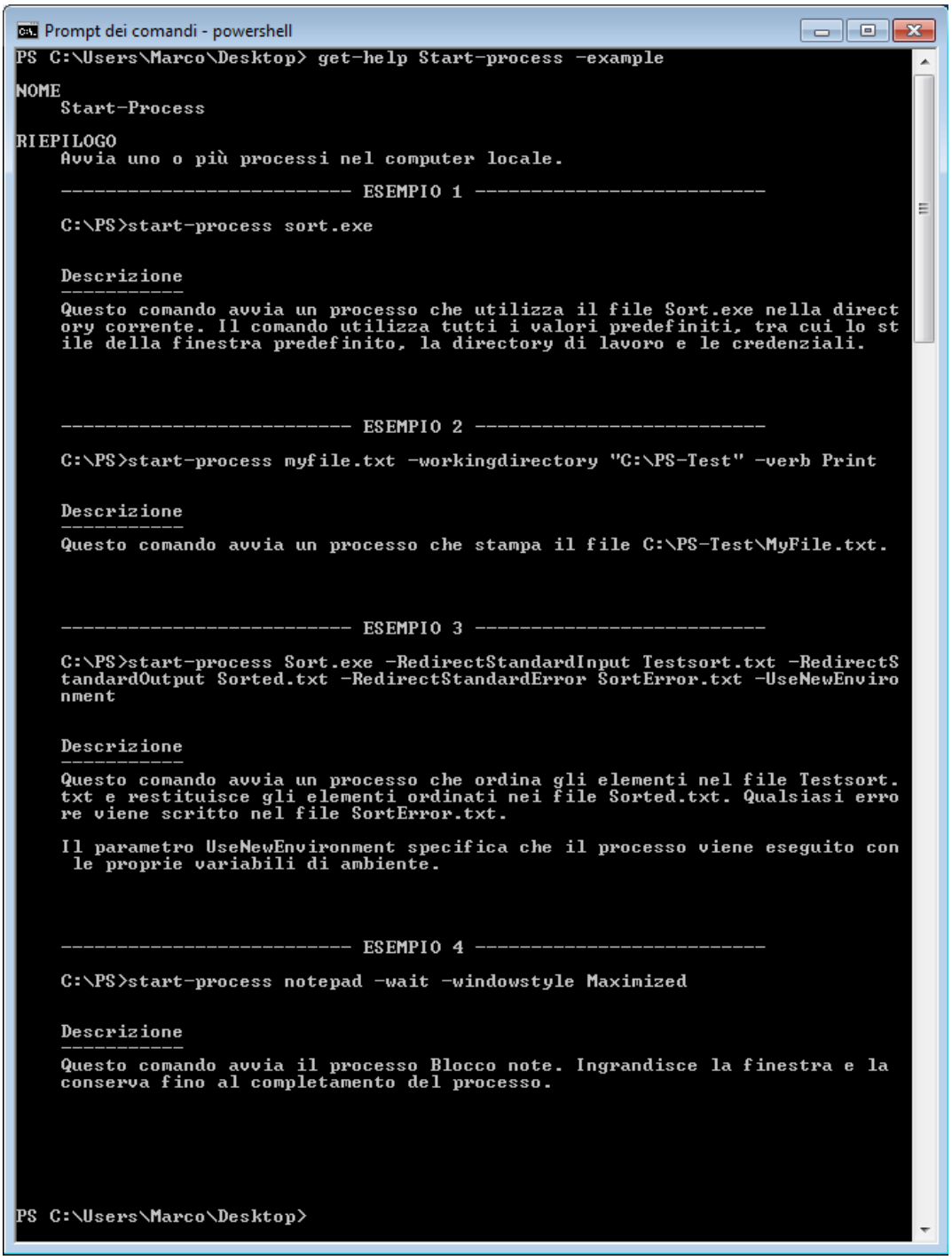


Vediamo come ottenere degli esempi sull'utilizzo del comando.

Digitiamo:

get-help Start-process –example

il sistema risponde come nella prossima immagine:



Vediamo quei comandi basilari che ci permettono di muoversi agevolmente tra i direttori in modo da poter successivamente essere in grado di amministrare agevolmente il server tramite i comandi di powershell e di netsh.

Ormai dovrebbe essere chiaro che il server potrebbe essere stato installato senza interfaccia grafica, oppure che è possibile eseguire un'amministrazione remota in cui la stessa non sarebbe altro che d'intralcio.

Consideriamo il comando “cd” che sta per change directory.

È importante capire che si cerca di convergere verso la filosofia amministrativa di un server Unix o Linux, quindi i comandi hanno specifiche simili, ad esempio il completamento automatico che si esegue con il tasto “TAB” presente nella prima colonna a sinistra della tastiera standard QUERTY.

Questo tasto può essere usato anche per un ritorno rapido alla radice delle path, che in linux è indicato con “./”

Così non è necessario indicare sempre la path completa, specialmente in fase di installazione, quando si lancia un file di setup.

Se dalla cartella dell’utente, ad esempio Marco, a cui si arriva tornando indietro di un livello dalla cartella Desktop.

Da qui eseguiamo il comando:

CD MU -> “TAB”

Il sistema completa con CD ./Music

Dando invio si porta alla path completa.

PS C:\Users\Marco\Music>

Ora eseguiamo un “dir” per vedere il contenuto.

Se invece di “dir” facciamo “ls” otterremo una indicazione Unix like, in cui vengono spiegati anche i permessi di lettura, scrittura, attraversamento dei file e delle cartelle.

Domanda

Come è possibile avere la lista dei servizi in esecuzione da linea di comando?

Risposta

Dalla console di powershell è possibile eseguire facilmente molte delle operazioni che in genere si eseguono manualmente tramite interfaccia grafica, in particolare con l'avanzare delle versioni di Windows.

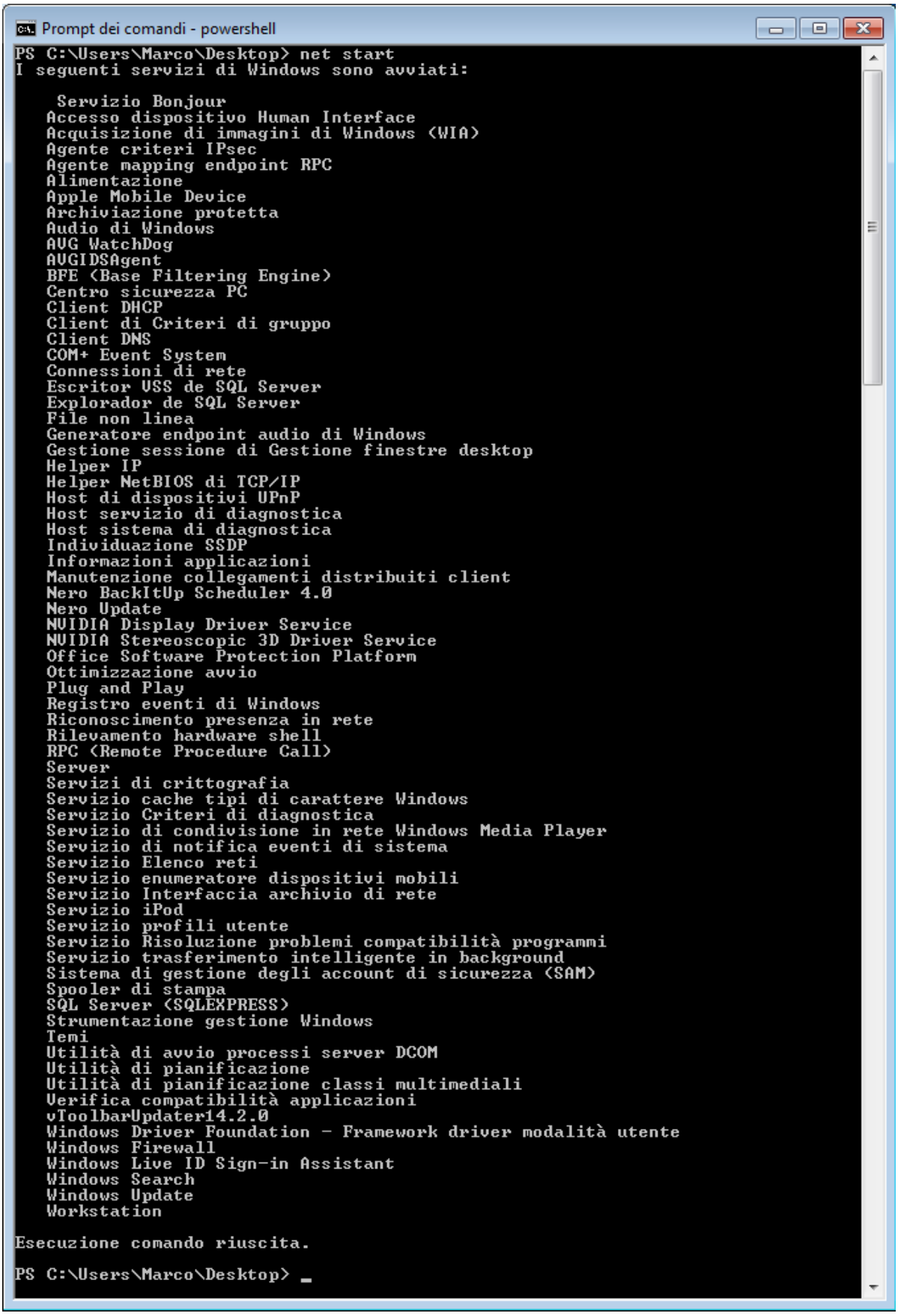
Non fa eccezione la gestione dei servizi. Per prima cosa è importante ricordare cos'è un servizio: un servizio è un programma che viene eseguito in background, senza interfacci grafica, che può svolgere i compiti più svariati, dalla gestione delle stampanti a quella del desktop, dal server web all'antivirus e tante altre funzionalità integrate nel sistema operativo. Sono in sostanza dei programmi sempre in esecuzione che possono essere autonomi o vivere in più d'uno in un solo processo (tipicamente ma non per forza, chiamato **svchost.exe**).

Vediamo per prima cosa come è possibile ottenere la lista di servizi in esecuzione senza ricorrere a **services.msc**. Dal prompt dei comandi semplicemente digitare

net start

Questo comando mostrerà una serie di nomi estesi dei servizi correntemente in esecuzione. Se si desidera terminarne uno, ad esempio il servizio "Audio di Windows" basterà digitare
net stop "Audio di Windows"
E quindi per riavviarlo
net start "Audio di Windows"
Se per qualche problema che si è verificato con il servizio si vuole arrestarlo e riavviarlo questa coppia di comandi può essere utile e spesso è la soluzione più semplice a molti problemi.
Un metodo alternativo per recuperare la lista di servizi in esecuzione è dare il seguente comando:
tasklist /svc
Il comando mostrerà la lista di tutti i processi in esecuzione, con nome, PID e nell'ultima colonna il nome (in versione breve) di eventuali servizi contenuti all'interno del processo. Questi nomi brevi possono essere utilizzati alternativa a quelli estesi, ad esempio se si vuole terminare il servizio "Audio di Windows" è anche possibile utilizzare il seguente comando:
net stop Audiosvc

Per vedere la lista dei servizi avviati nel computer in uso digitiamo il comando “net start”, l’effetto è il seguente:


Prompt dei comandi - powershell
PS C:\Users\Marco\Desktop> net start
I seguenti servizi di Windows sono avviati:
Servizio Bonjour
Accesso dispositivo Human Interface
Acquisizione di immagini di Windows (WIA)
Agente criteri IPsec
Agente mapping endpoint RPC
Alimentazione
Apple Mobile Device
Archiviazione protetta
Audio di Windows
AVG WatchDog
AVGIDSAgent
BFE (Base Filtering Engine)
Centro sicurezza PC
Client DHCP
Client di Criteri di gruppo
Client DNS
COM+ Event System
Connessioni di rete
Escritor VSS de SQL Server
Explorador de SQL Server
File non linea
Generatore endpoint audio di Windows
Gestione sessione di Gestione finestre desktop
Helper IP
Helper NetBIOS di TCP/IP
Host di dispositivi UPnP
Host servizio di diagnostica
Host sistema di diagnostica
Individuazione SSDP
Informazioni applicazioni
Manutenzione collegamenti distribuiti client
Nero BackItUp Scheduler 4.0
Nero Update
NVIDIA Display Driver Service
NVIDIA Stereoscopic 3D Driver Service
Office Software Protection Platform
Ottimizzazione avvio
Plug and Play
Registro eventi di Windows
Riconoscimento presenza in rete
Rilevamento hardware shell
RPC (Remote Procedure Call)
Server
Servizi di crittografia
Servizio cache tipi di carattere Windows
Servizio Criteri di diagnostica
Servizio di condivisione in rete Windows Media Player
Servizio di notifica eventi di sistema
Servizio Elenco reti
Servizio enumeratore dispositivi mobili
Servizio Interfaccia archivio di rete
Servizio iPod
Servizio profili utente
Servizio Risoluzione problemi compatibilità programmi
Servizio trasferimento intelligente in background
Sistema di gestione degli account di sicurezza (SAM)
Spooler di stampa
SQL Server (SQLEXPRESS)
Strumentazione gestione Windows
Temi
Utilità di avvio processi server DCOM
Utilità di pianificazione
Utilità di pianificazione classi multimediali
Verifica compatibilità applicazioni
vToolbarUpdater14.2.0
Windows Driver Foundation - Framework driver modalità utente
Windows Firewall
Windows Live ID Sign-in Assistant
Windows Search
Windows Update
Workstation
Esecuzione comando riuscita.
PS C:\Users\Marco\Desktop>

Domanda

Come si può terminare un servizio da linea di comando se il comando net stop non funziona?

Risposta

Se con il comando **net stop**, si dovesse ricevere un errore che rende impossibile l'arresto del servizio si può provare a terminare il processo che contiene il servizio. Per fare questo occorre prima di tutto individuare il PID (ovvero l'identificatore univoco) del processo tramite il comando **tasklist /svc**, come visto poco fa, cercando nell'ultima colonna mostrata un riferimento al servizio. In secondo luogo bisogna verificare quali altri servizi sono in esecuzione nello stesso processo, è importante accertarsi che tra questi non ve ne siano di critici, o meglio ancora che sia l'unico, poiché terminare servizi di sistema critici potrebbe rendere instabile il sistema.

Una volta individuato il PID, sempre dal prompt dare il seguente comando:

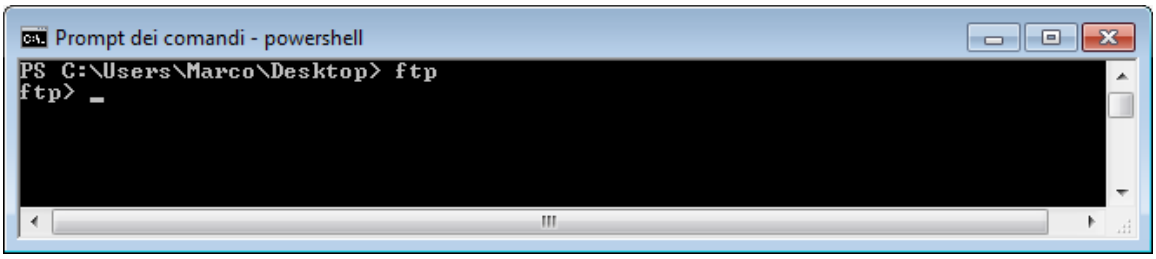
taskkill /PID pidDelServizio /F

Dove il parametro **F** indica che si deve tentare di forzare la terminazione del processo.

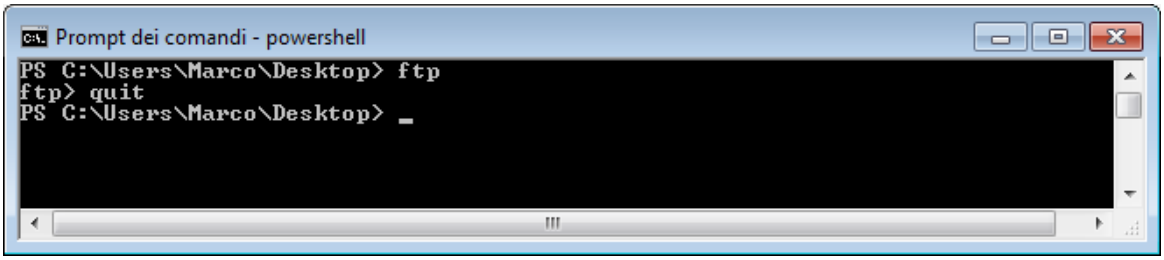

Prompt dei comandi - powershell
PS C:\Users\Marco\Desktop> taskkill -?
TASKKILL [/S sistema [/U nomeutente [/P [password]]]]
{ [/FI filtro] [/PID idprocesso | /IM nomeimmagine] } [/T] [/F]
Descrizione:
Questo strumento della riga di comando è utilizzato per terminare
attività in base all'ID del processo (PID) e al nome immagine.
Elenco parametri:
/S sistema Specifica il sistema remoto a cui connettersi.
/U [dominio\]utente Specifica il contesto utente in cui
eseguire il comando.
/P [password] Specifica la password per il contesto utente
indicato. Se omessa, la password viene richiesta.
/FI filtro Applica un filtro per selezionare un insieme
di attività.
Consente l'utilizzo di "*". Ad es.: nomeimmagine
eq memo*
/PID idprocesso Specifica il PID del processo da terminare.
Utilizzare TaskList per ottenere il PID.
/IM nomeimmagine Specifica il nome immagine del processo
terminare. È possibile utilizzare il carattere
jolly '*' per specificare tutti i nomi di
attività o di immagine.
/T Termina il processo specificato e tutti
i processi figlio avviati dallo stesso.
/F Specifica l'interruzione forzata del processo o
dei processi.
/? Visualizza questo messaggio della Guida.
Filtri:
Nome filtro Operatori validi Valore/i valido/i
STATUS eq, ne IN ESECUZIONE |
NOT RESPONDING | UNKNOWN
IMAGENAME eq, ne Nome immagine
PID eq, ne, gt, lt, ge, le Valore PID
SESSION eq, ne, gt, lt, ge, le Numero di sessione
CPUTIME eq, ne, gt, lt, ge, le Tempo di CPU nel formato
hh:mm:ss.
hh - ore,
mm - minuti, ss - secondi
MEMUSAGE eq, ne, gt, lt, ge, le Utilizzo della memoria in KB
USERNAME eq, ne Nome utente in formato
[dominio\]utente
MODULES eq, ne Nome DLL
SERVICES eq, ne Nome servizio
WINDOWTITLE eq, ne Titolo finestra
NOTA
1) Il carattere jolly '*' per l'opzione /IM è accettato soltanto quando
viene applicato un filtro.
2) La terminazione dei processi remoti sarà sempre imposta (/F).
3) I filtri "WINDOWTITLE" e "STATUS" non vengono considerati quando si
specifica un computer remoto.
Esempi:
TASKKILL /IM notepad.exe
TASKKILL /PID 1230 /PID 1241 /PID 1253 /T
TASKKILL /F /IM cmd.exe /T
TASKKILL /F /FI "PID ge 1000" /FI "WINDOWTITLE ne untitle*"
TASKKILL /F /FI "USERNAME eq NT AUTHORITY\SYSTEM" /IM notepad.exe
TASKKILL /S sistema /U dominio\nomeutente /FI "USERNAME ne NT*" /IM *
TASKKILL /S sistema /U nomeutente /P password /FI "IMAGENAME eq note*"
PS C:\Users\Marco\Desktop>

Comandi FTP da powershell

FTP, ovvero File Transfert Protocol, è usato per la trasmissione massiva di dati da una sorgente a una destinazione. Ottimale per file di grandi dimensione come archivi o film. È possibile lanciare una sessione FTP da powershell usando il cmdlet "ftp" al prompt di powershell.



Per uscire dalla sessione di FTP digitare "quit"



I comandi FTP accettati dalla console FTP sono i seguenti:

ABOR: annulla il trasferimento del file corrente;
APPE: invia un file in modalità accodamento, ovvero scrive i dati alla fine del file;
AUTH: specifica il tipo di autenticazione;
CDUP: cambia la directory corrente alla directory superiore;
CWD: cambia la directory corrente ad una specificata;
DELE: elimina un file;
FEAT: elenca i comandi supportati dal server FTP;
LIST: restituisce informazioni a riguardo della cartella corrente o, se specificato, di un file;
MDTM: restituisce la data di ultima modifica di un file specificato;
MKD: crea una cartella;
MODE: imposta la modalità di trasferimento (stream, block o compressed);
NLST: restituisce la lista dei file di una cartella;
NOOP: non esegue alcuna operazione, comando di IDLE;
PASS: password per l'autenticazione;
PASV: entra in modalità passiva, sarà quindi il server ad aprire le porte per ricevere la connessione per i trasferimenti di dati;
PORT: specifica a quale IP e porta il server deve connettersi per avviare il trasferimento dati;
PWD: restituisce il nome della cartella corrente;
QUIT: disconnette dal server;
REST: specifica da quale byte avviare il trasferimento, utilizzato per continuare trasferimenti interrotti;
RETR: avvia il download di un file;
RMD: elimina un directory;
RNFR: nell'operazione di ridenominazione imposta il file da rinominare;

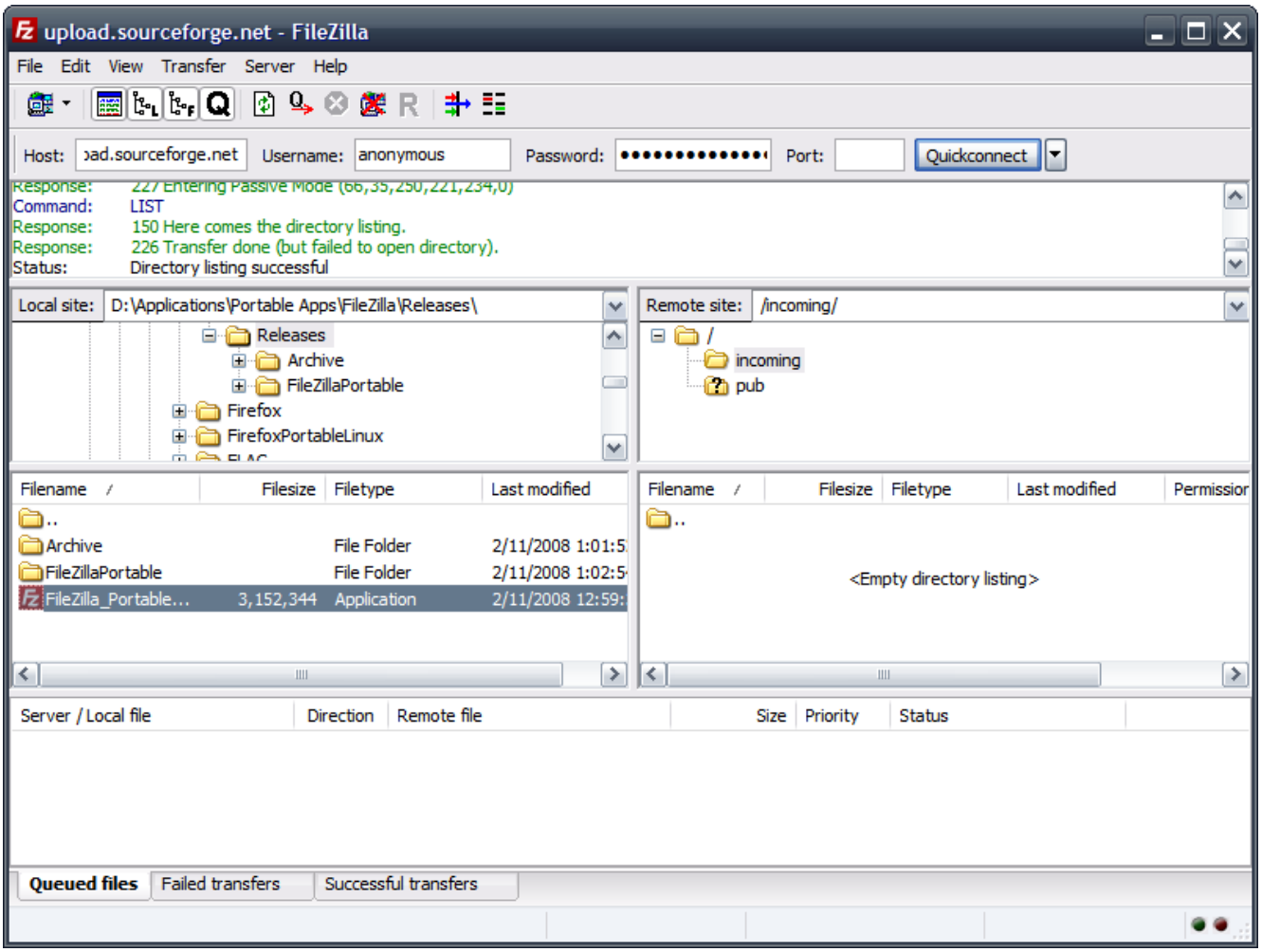
RNTO: rinomina il file precedentemente specificato con RNFR al nome specificato, utilizzato anche per spostare file;
SIZE: restituisce la dimensione di un file:
STOR: inizializza l'upload di un file;
STOU: inizializza l'upload di un file assegnandogli un nome non presente nella cartella di destinazione;
SYST: restituisce informazioni sul server, utile per determinare il tipo di responso del comando LIST (formato UNIX o DOS);
TYPE: imposta la modalità di trasferimento, ASCII o Binary, dove la prima potrebbe avere effetti sulla conversione dei ritorni a capo;
USER: specifica il nome utente per l'autenticazione;

Vediamo come avviene il trasferimento di un file e spieghiamo cos'è la modalità passiva:

Il trasferimento di un file, si tratti di un upload o di un download, avviene sempre su una connessione distinta da quella sulla quale vengono inviati i comandi (connessione di controllo). Esistono due modalità per la creazione di questa connessione, o il client apre una certa porta in locale e attende che il server si connetta (modalità attiva) o viceversa, il server apre una porta ed attende che sia il client a collegarsi (modalità passiva).

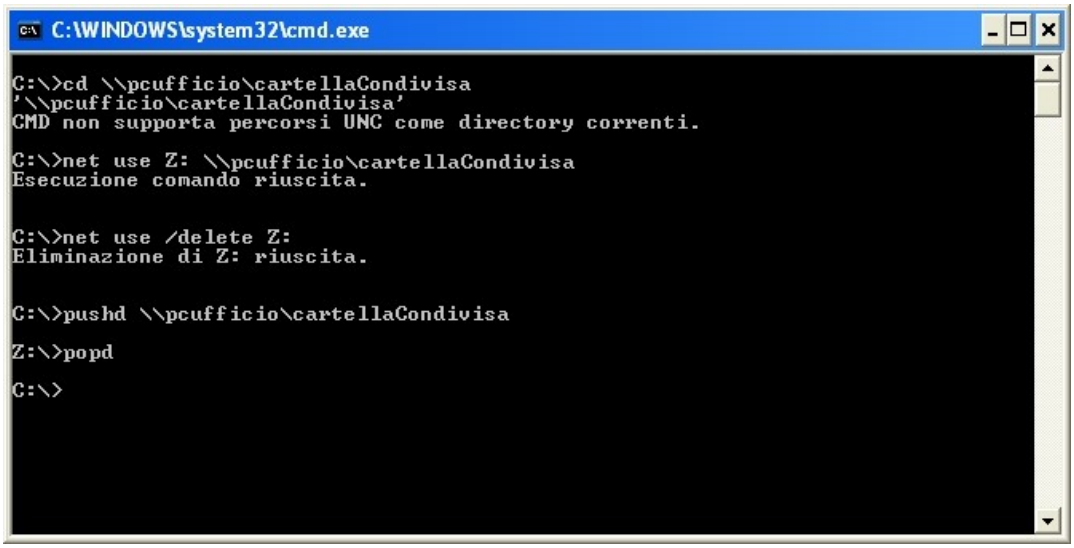
La modalità passiva è utile nel caso ci si trovi dietro ad un firewall che non permette di aprire le porte in ingresso e non ha svantaggi rispetto alla modalità passiva.

L'applicativo "FileZilla" è uno dei software più in voga per eseguire sessioni FTP in maniera visuale.


upload.sourceforge.net - FileZilla
File Edit View Transfer Server Help
Host: oad.sourceforge.net Username: anonymous Password: Port: Quickconnect
Response: 227 Entering Passive Mode (66,35,250,221,234,0)
Command: LIST
Response: 150 Here comes the directory listing.
Response: 226 Transfer done (but failed to open directory).
Status: Directory listing successful
Local site: D:\Applications\Portable Apps\FileZilla\Releases\
Releases
Archive
FileZillaPortable
Firefox
FirefoxPortableLinux
Remote site: /incoming/
/
incoming
pub
Filename Filesize Filetype Last modified
.. Archive File Folder 2/11/2008 1:01:5
FileZillaPortable File Folder 2/11/2008 1:02:5
FileZilla_Portable... 3,152,344 Application 2/11/2008 12:59:
Filename Filesize Filetype Last modified Permissior
..
<Empty directory listing>
Server / Local file Direction Remote file Size Priority Status
Queued files Failed transfers Successful transfers

Percorsi UNC dal prompt dei comandi

Come è possibile accedere ad un percorso di rete (un percorso UNC come "\\pcufficio\cartellaCondivisa") tramite il prompt dei comandi?



Il **prompt dei comandi** (**Start**, **Esegui**, digitare **cmd** e dare conferma) di per sé non supporta l'utilizzo di percorsi di rete, o meglio il comando **ChDir** (o più di frequente CD, comando che serve per spostarsi tra le cartelle) non lo supporta, altri comandi non hanno problemi a lavorare con percorsi UNC. Ad esempio è possibile senza alcun problema re direzionare l'output di un comando o programma ad un file su un percorso di rete:

echo Contenuto del file > \\pcufficio\cartellaCondivisa\nuovo_file.txt

Mentre invece se proviamo a dare un comando come

cd \\pcufficio\cartellaCondivisa

riceviamo il messaggio "CMD non supporta percorsi UNC come directory correnti". Per ovviare a questo problema è necessario creare un'**unità di rete**, ovvero un'unità virtuale (ad esempio Z:\) a cui viene associata la cartella di rete in questione. Per fare ciò vi sono varie vie.

Se si desidera effettuare questa operazione dal prompt è necessario usare il comando **net use**. Per associare alla cartella di rete **\\pcufficio\cartellaCondivisa** l'unità **Z** usare la seguente sintassi:

net use z: \\pcufficio\cartellaCondivisa

Per disattivare l'unità di rete invece:

net use /delete z:

A questo punto semplicemente digitando Z: si avrà accesso alla cartella di rete e sarà ad esempio possibile effettuare un **Dir** (comando per ottenere la lista dei file contenuti nella cartella corrente). Per ulteriori informazioni sul comando **net use** digitare **net help use**: il comando permette anche di impostare

credenziali di accesso e altre opzioni a volte indispensabili. Se questa soluzione dovesse sembrare ostica, è possibile creare un'unità di rete anche per via grafica:

aprire **Risorse del computer**, menu **Strumenti**, **Connetti unità di rete...**, selezionare la lettera e digitare il percorso di rete (oppure individuarlo tramite il pulsante **Sfoglia**). Se si desidera rendere permanente l'unità di rete selezionare la casella **Riconnetti all'avvio**. Dando conferma l'unità di rete verrà creata.

Il terzo e ultimo modo è il più comodo da utilizzare dal prompt dei comandi per agire temporaneamente su di una cartella di rete: si tratta di utilizzare il comando **pushd** e **popd**. Per creare un'unità di rete digitare:

pushd \\pcufficio\cartellaCondivisa

Fatto questo il prompt creerà un'unità di rete identificata dalla prima lettera di unità disponibile a partire dalla Z andando a ritroso (dunque se Z: è già in uso verrà utilizzata Y:, W: e così via). Per eliminare l'unità di rete basterà richiamare il comando **popd** senza parametri.

Si badi che nei due metodi che utilizzano il prompt (ovvero tramite **net use** e **pushd**/**popd**) se la finestra di comando viene chiusa senza dare i rispettivi comandi per disattivare l'unità di rete essa rimarrà attiva fino al prossimo riavvio. Per disattivare un'unità di rete usare la sintassi precedentemente vista di **net use** con il parametro **delete** oppure da **Risorse del computer** fare click destro sull'unità di rete in questione e cliccare su **Disconnetti**.

La lista dei comandi netsh di window server 2008 R2 è la seguente:

exportkey	exportkey	flush
importkey	set cachesize	set key
set localcache	set publicationcache	set publicationcachesize
set latency	show status	set service
show hostedcache	show localcache	smb
show publicationcache	show latency	reset

Nella versione server 2012 i comandi BranchCache di tipo Cmdlets sono eseguibili solo da powershell e la lista è la seguente.

cmdlet	**Description**
Add-BCDataCacheExtension	Increases the amount of cache storage space that is available on a hosted cache server by adding a new cache file.
Clear-BCCache	Deletes all data in all data and hash files.
Disable-BC	Disables the BranchCache service.
Disable-BCDowngrading	Disables downgrading, so that client computers that are running Windows® 8 Consumer Preview do not request Windows® 7-specific versions of content information from content servers.
Disable-BCServeOnBattery	Configures a client to ignore content discovery requests in distributed cache mode when operating on battery power.
Enable-BCDistributed	Enables BranchCache and configures a computer to operate in distributed cache mode.
Enable-BCDowngrading	Instructs a client computer that is running Windows® 8 Consumer Preview to operate in a downgraded Windows® 7 mode.
Enable-BCHostedClient	Configures BranchCache to operate in hosted cache client mode.
Enable-BCHostedServer	Configures BranchCache to operate in hosted cache server mode.
Enable-BCLocal	Enables the BranchCache service in local caching mode.
Enable-BCServeOnBattery	Configures a client to listen for content discovery requests in distributed cache mode when operating on battery power.

Export-BCCachePackage	Exports a cache package.
Export-BCSecretKey	Exports a secret key to a file.
Get-BCClientConfiguration	Retrieves the current BranchCache client computer settings.
Get-BCContentServerConfiguration	Retrieves the current BranchCache content server settings.
Get-BCDataCache	Retrieves the BranchCache data cache.
Get-BCDataCacheExtension	Retrieves the BranchCache data cache extensions from a hosted cache server.
Get-BCHashCache	Retrieves the BranchCache hash cache.
Get-BCHostedCacheServerConfiguration	Retrieves the current BranchCache hosted cache server settings.
Get-BCNetworkConfiguration	Retrieves the current BranchCache network settings.
Get-BCStatus	Retrieves a set of objects that provide BranchCache status and configuration information.
Import-BCCachePackage	Imports a cache package.
Import-BCSecretKey	Imports the cryptographic key that BranchCache uses for the generation of segment secrets.
Publish-BCFileContent	Generates hashes, also called content information, for files in shared folders on a file server that have BranchCache enabled and the BranchCache for Network Files role service installed.
Publish-BCWebContent	Creates hashes for web content when deploying content servers that are running Windows Server® 8 Beta with the Web Services (IIS) server role installed.
Remove-BCDataCacheExtension	Deletes a data cache file.
Reset-BC	Resets BranchCache to the default configuration.
Set-BCAuthentication	Specifies the BranchCache computer authentication mode.
Set-BCCache	Modifies the cache file configuration.
Set-BCDataCacheEntryMaxAge	Modifies the maximum amount of time that data can remain in the cache.
Set-BCMinSMBLatency	Sets the minimum latency that must exist between client and server before transparent caching functions are utilized.
Set-BCSecretKey	Sets the cryptographic key used in the generation of segment secrets.

Per visualizzare tutti I comandi Cmdlet disponibili usiamo il commando Get-Command –Module

BranchCache cmdlet.

Per avere maggiori informazioni sulla sintassi dei comandi cmdlet digitare “Get-Help *<cmdlet>*. Per maggiori dettagli si può usare uno dei tre comandi sottostanti:

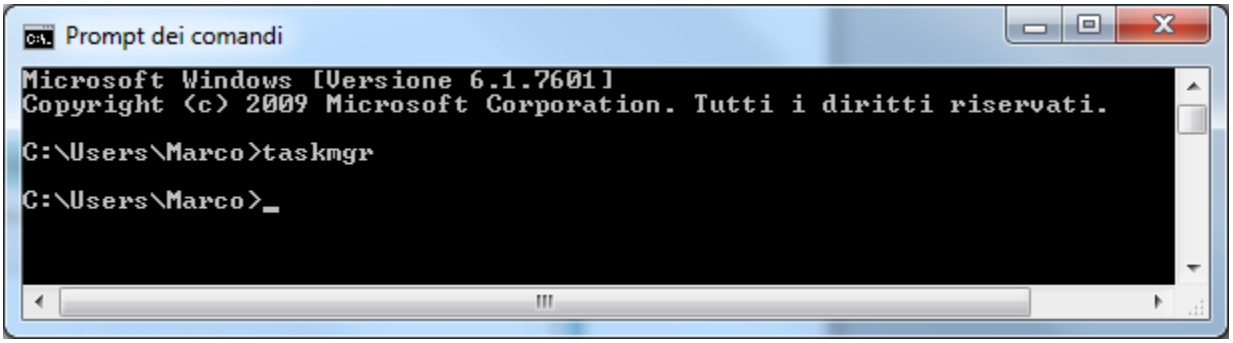
- Get-Help <cmdlet name> -Detailed
- Get-Help <cmdlet name> -Examples
- Get-Help <cmdlet name> -Full

Alcuni comandi di powershell, più specifici per la versione server 2012 sono nella tabella seguente:

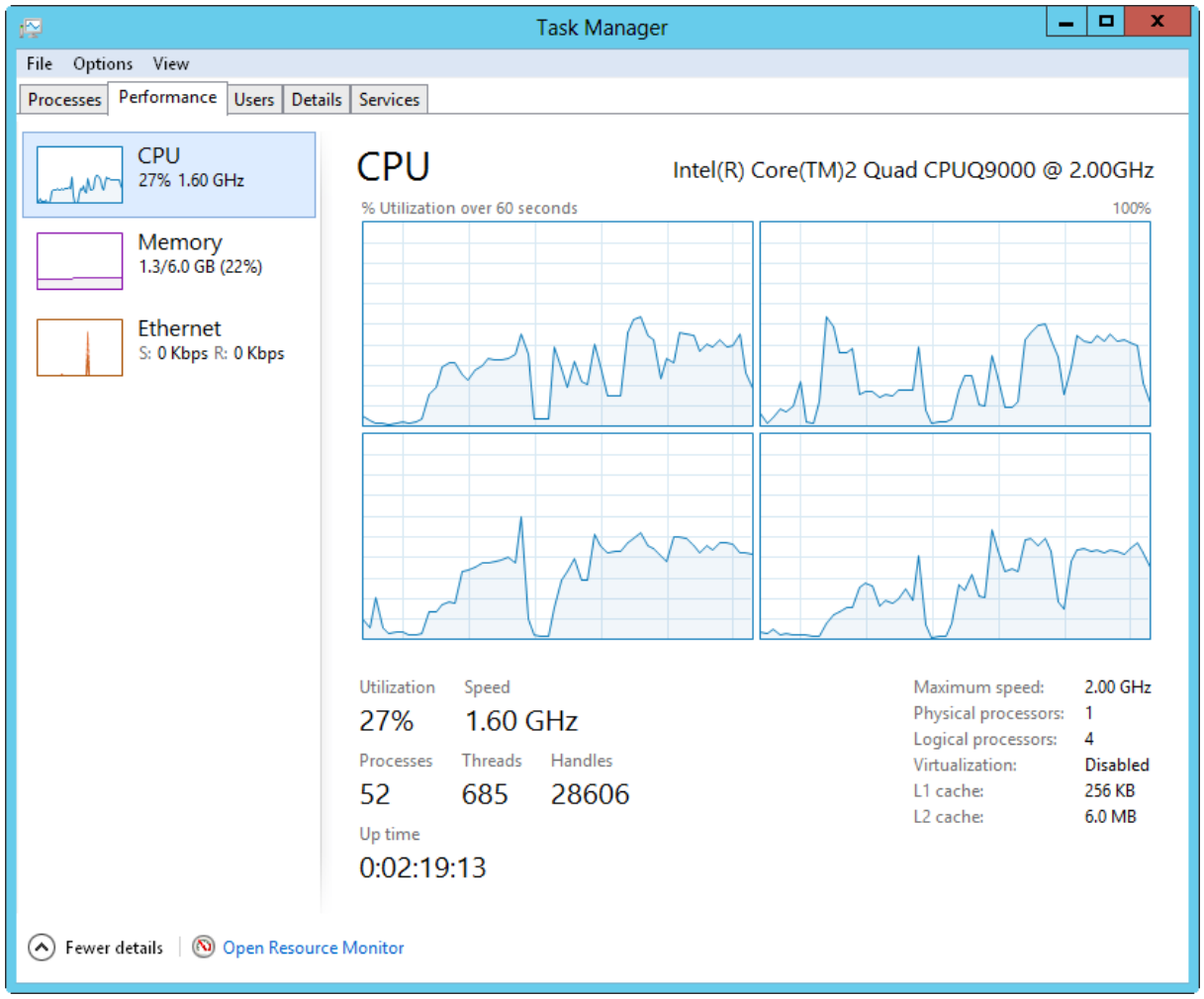
COMMAND	TASK
Cscript Scregedit.wsf	Configure the operating system. Use the */cli* parameter to list available configuration areas.
Diskraid.exe	Configure software RAID.
ipconfig /all	List information about the computer's IP address configuration.
Netdom RenameComputer	Set the server's name.
Netdom Join	Join the server to a domain.
Netsh	Provide multiple contexts for managing the configuration of networking components. Type **netsh interface ipv4** to configure IPv4 settings. Type **netsh interface ipv6** to configure IPv6 settings.
Ocsetup.exe	Add or remove roles, role services, and features.
Pnputil.exe	Install or update hardware device drivers.
Sc query type=driver	List installed device drivers.
Serverweroptin.exe	Configure Windows Error Reporting.
Slmgr –ato	Windows Software Licensing Management tool used to activate the operating system. Runs *Cscript slmgr.vbs –ato*.
Slmgr –ipk	Install or replace the product key. Runs *Cscript slmgr.vbs –ipk*.

COMMAND	TASK
SystemInfo	List the system configuration details.
Wecutil.exe	Create and manage subscriptions to forwarded events.
Wevtutil.exe	View and search event logs.
Winrm quickconfig	Configure the server to accept WS-Management requests from other computers. Runs *Cscript winrm.vbs quickconfig*. Enter without the *quickconfig* parameter to see other options.
Wmic datafile where name="FullFilePath" get version	List a file's version.
Wmic nicconfig index=9 call enabledhcp	Set the computer to use dynamic IP addressing rather than static IP addressing.
Wmic nicconfig index=9 call enablestatic("IPAddress"), ("SubnetMask")	Set a computer's static IP address and network mask.
Wmic nicconfig index=9 call setgateways("GatewayIPAddress")	Set or change the default gateway.
Wmic product get name /value	List installed Microsoft Installer (MSI) applications by name.
Wmic product where name="Name" call uninstall	Uninstall an MSI application.
Wmic qfe list	List installed updates and hotfixes.
Wusa.exe PatchName.msu /quiet	Apply an update or hotfix to the operating system.

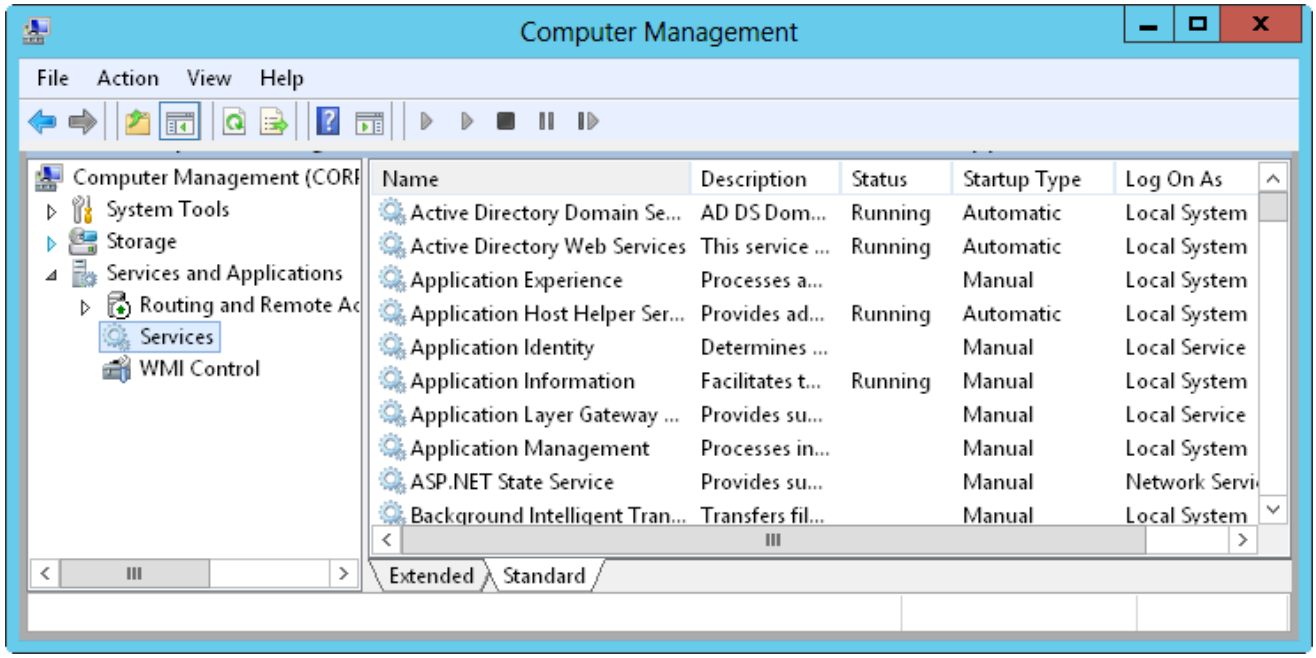
Per lanciare il task manager digitare tskmgr da powershell o da prompt dei comandi.



Una volta lanciato il task manager è possibile visualizzare lo stato e le perfomance dell'hardware.

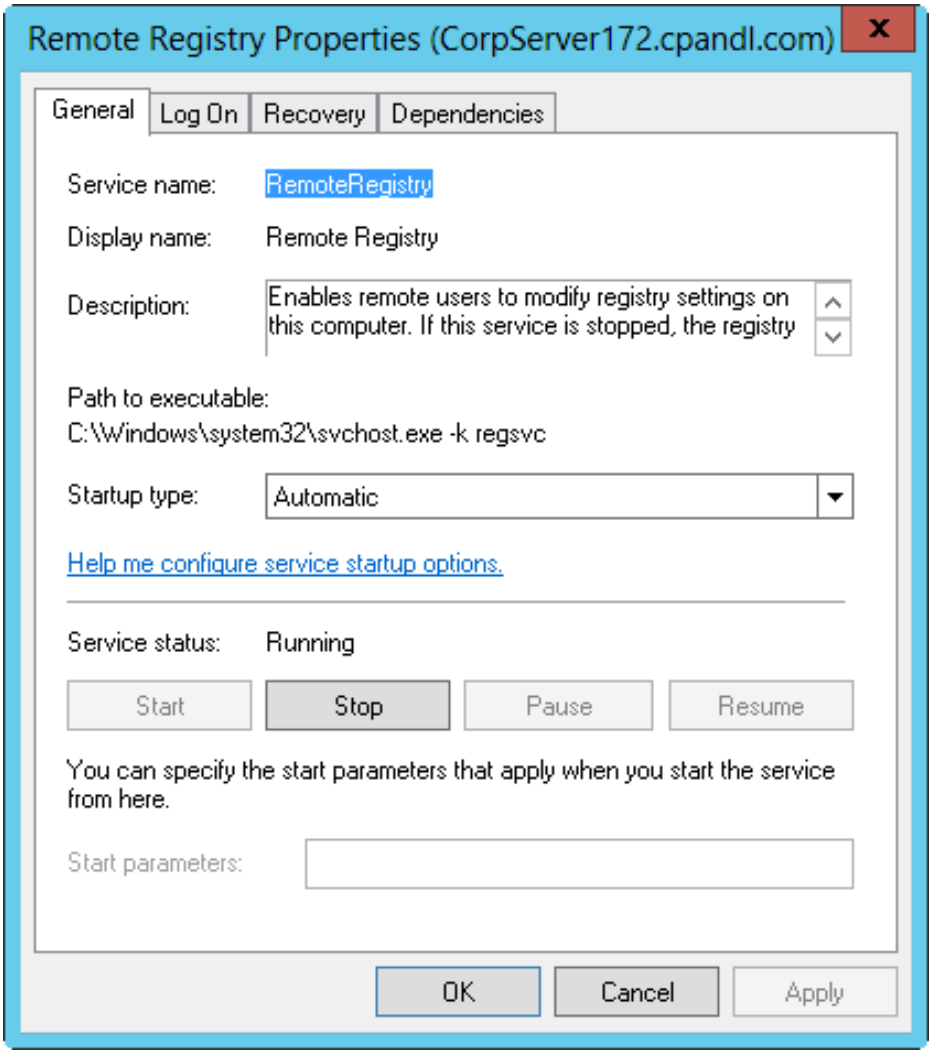


Nella finestra “Computer management, è possibile visualizzare i servizi attivi nel server.



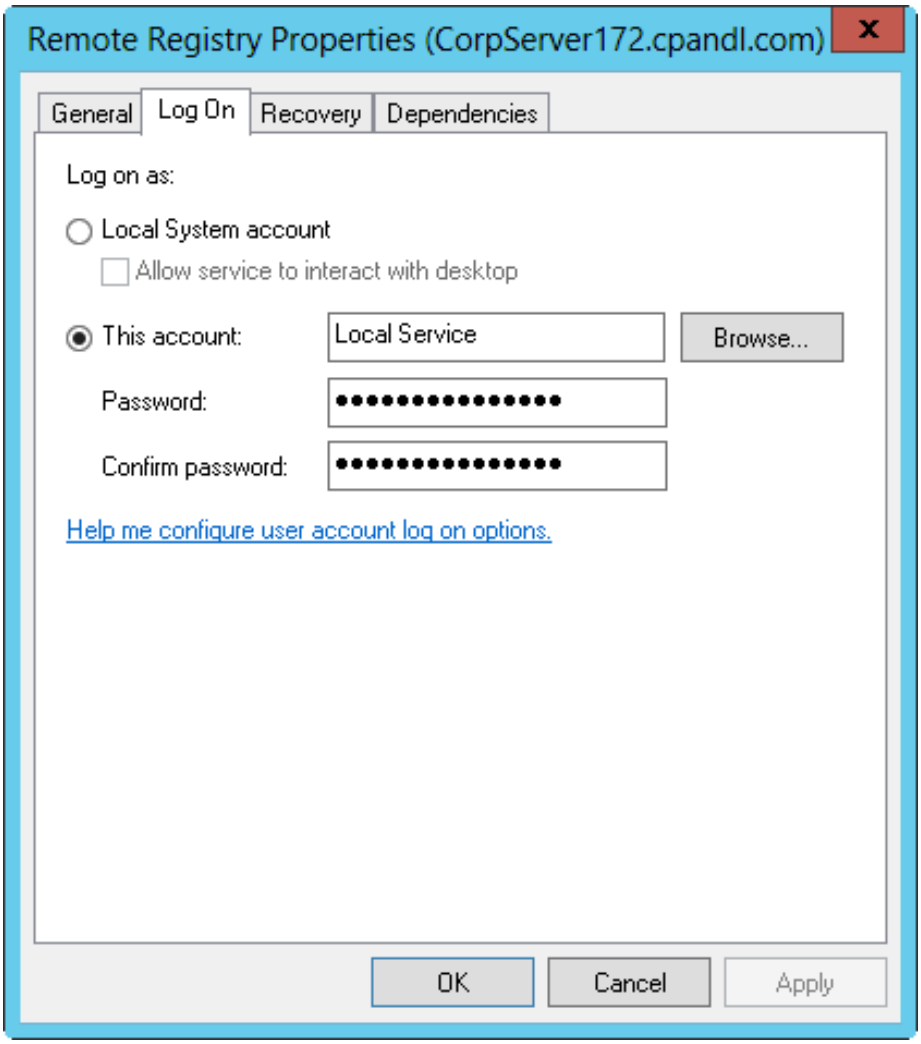
Per fermare, oppure riavviare oppure bloccare permanentemente un servizio, tappiamo o clicchiamo su di esso nella finestra “computer Management” visualizzata nell’immagine precedente. Vedremo comparire la finestra della proprietà del servizio. Nell’esempio è stato scelto il “Remote registry”. Del quale potremmo selezionare le opzioni di startup. Queste sono:

1. Automatico -> sia avvia automaticamente al boot.
2. Automatico ritardato -> Per avviarlo dopo la sequenza di boot.
3. Manuale -> per avviarlo su richiesta dell’amministratore.
4. Disabilitato



Configurare il servizio di log on

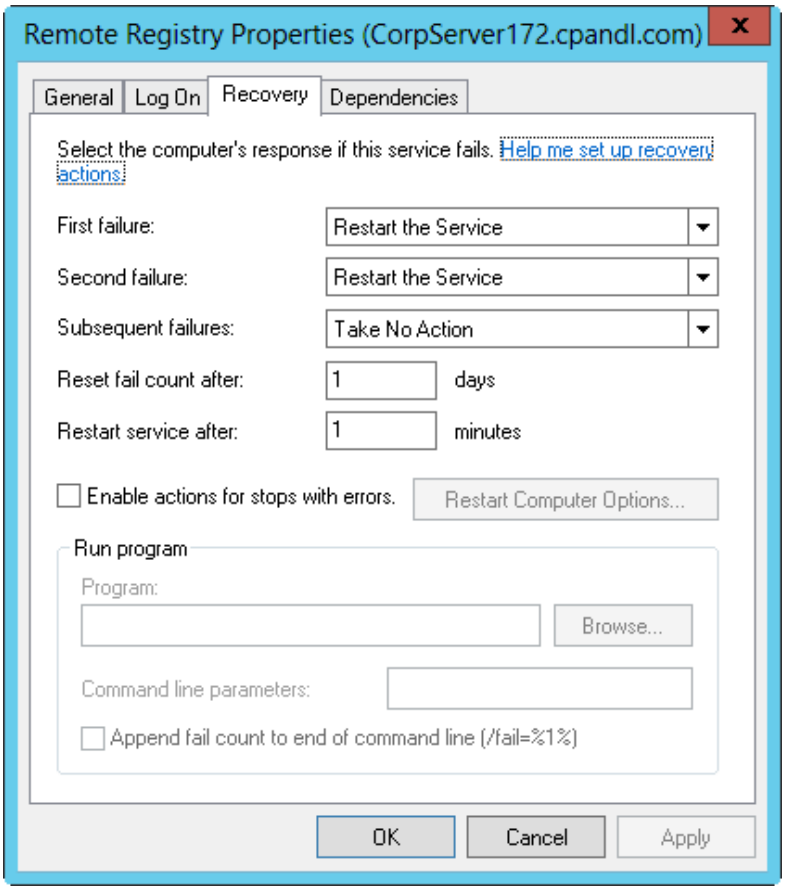
Si può eseguire per logarsi al sistema oppure per un singolo account utente. Dalla finestra di “Computer management” tenere il ditto sul tap, oppure eseguire tasto destro, sul servizio che si vuole configurare e quindi selezionare “proprietà”. Successivamente selezionare il Tab, “Log On”.



Selezionare “Local System account” se si desidera che il servizio di accesso utilizzi l’account di sistema (è l’impostazione predefinita per la maggior parte dei servizi). Se il servizio fornisce un interfaccia utente che può essere manipolata, selezionare “Consenti al servizio di interagire con Desktop” per consentire agli utenti di controllare l'interfaccia del servizio.

Configurazione di recovery dei servizi.
È possibile configurare le azioni che dovranno accadere quando un servizio fallisce l’attivazione o la sua esecuzione. Ad esempio è possibile programmare il suo riavvio o lanciare un’applicazione di recovery.
Vanno eseguiti i seguenti step:

- In “Computer management” premere o cliccare sul servizio che si vuole configurare, e poi scegliere proprietà.
- Tapp-are o cliccare su “recovery”

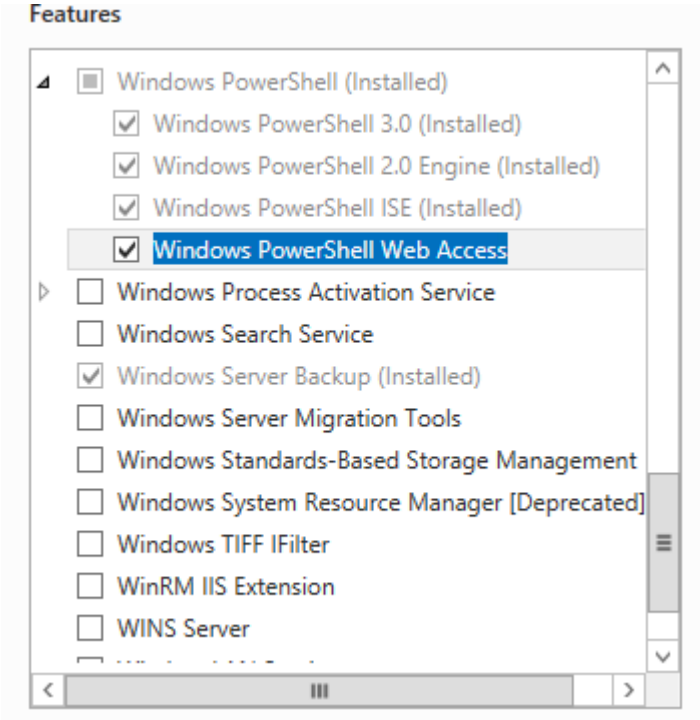


Windows server 2012 Power shell WEb access.
Windows 2012 è amministrabile da remoto tramite riga di commando grazie all'istallazione, che avviene per default, di questa utility, che crea un gateway di accesso, per l'amministrazione del server, tramite i comandi testuali lanciabili dalla console di power shell noti come "cmdlet".

Power shell non è molto usato perché assomiglia un po' a un salto al passato quando si amministravano i sistemi da riga di comando, ma in realtà si avvicina molto al modo di pensare e operare degli amministratori di sistemi basati su linux.

Le configurazione dei ruoli del server risultano abbastanza semplici. Identificata la macchina in cui istallare il servizio comunicheremo con essa tramite la porta 443.

L'attivazione può avvenire agendo su "Server management" come in figura.

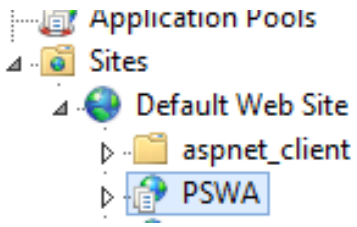


Notiamo tra le features disponibili la presenza del TIFF filter che è uno strumento di ottimizzazione delle ricerche all'interno di testi a alto contrasto visivo quali quelli scritti in nero su sfondo bianco. Attivata la feature è necessario procedere con la creazione dell'applicazione all'interno di IIS e delle relative configurazioni di sicurezza.

Il primo comando da lanciare è il seguente:

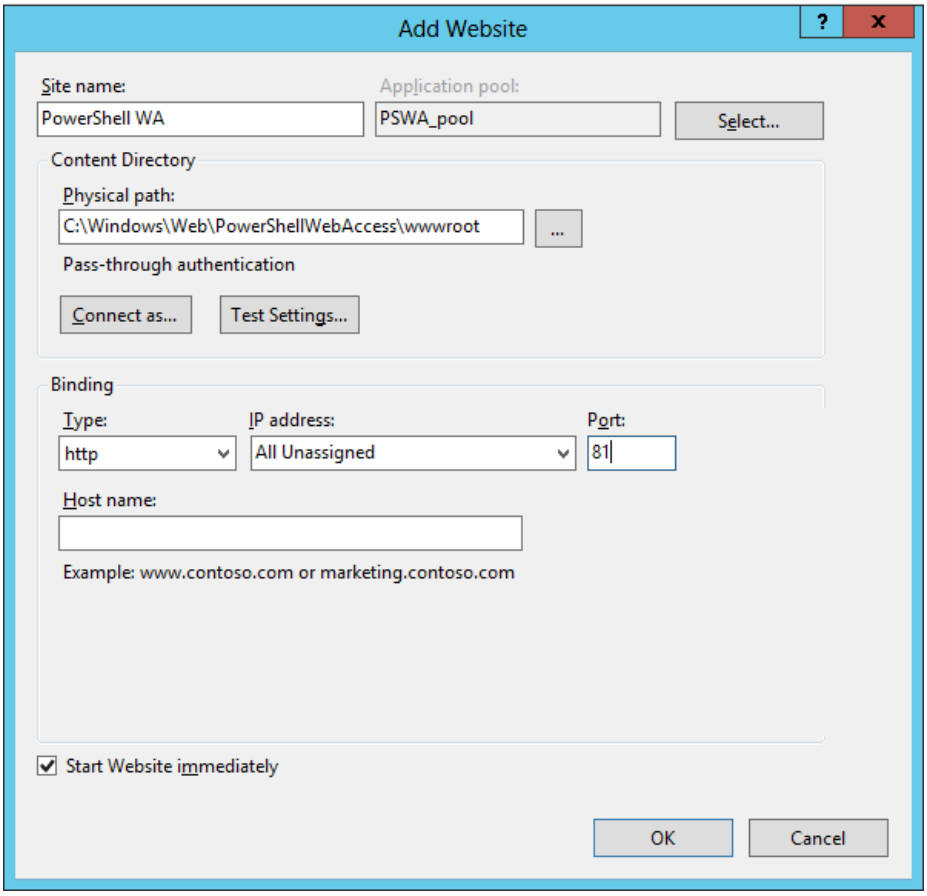
Install-PswaWebApplication –webApplicationName PSWA -useTestCertificate

All'interno di IIS comparirà una nuova applicazione dentro il Default Web Site, come mostra la prossima figura.



La features potrebbe essere anche pronta così, tuttavia è sempre meglio creare un sito web dedicato, anche perché è possibile agganciare il certificato digitale in modo corretto e fare le relative configurazioni.

Il nuovo sito dovrà puntare alla cartella **C:\Windows\Web\PowerShellWebAccess\wwwroot** e dovrà utilizzare lo stesso ApplicationPool creato automaticamente dal comando lanciato in precedenza, come mostra la prossima immagine.



Configurazione Nuovo Sito

Essendo un Gateway che espone dei servizi di un certo tipo, non è possibile utilizzare porte diverse dalla 443, quindi sarà necessario generare un certificato digitale interno per fare in modo che il sito risulti correttamente configurato. Il certificato può essere sia creato da un CA interna che acquistato presso un'azienda specializzata.

Una volta configurato il certificato sul sito, si può procedere con l'attivazione dei permessi per l'accesso alle macchine. Questo è un codice di esempio:

Add-PswaAuthroizationRule –UserGroupName 'centroZIP\IT Support' –ComputerGroupName 'centroZIP\HV Hosts' –ConfigurationName Microsoft.powershell

Come si può notare dal codice, viene fatta l'attivazione per il gruppo IT Support, in modo che possa accedere a tutte le macchine presenti all'interno del gruppo HV Hosts. Ovviamente si può andare a fare l'attivazione anche di un singolo utente e di una singola macchina, con una sintassi simile a questa:

Add-PswaAuthorizationRule -UserName 'centroZIP\prof.gottardo' -ComputerName 'myhost.centroZIP.com' -ConfigurationName Microsoft.powershell

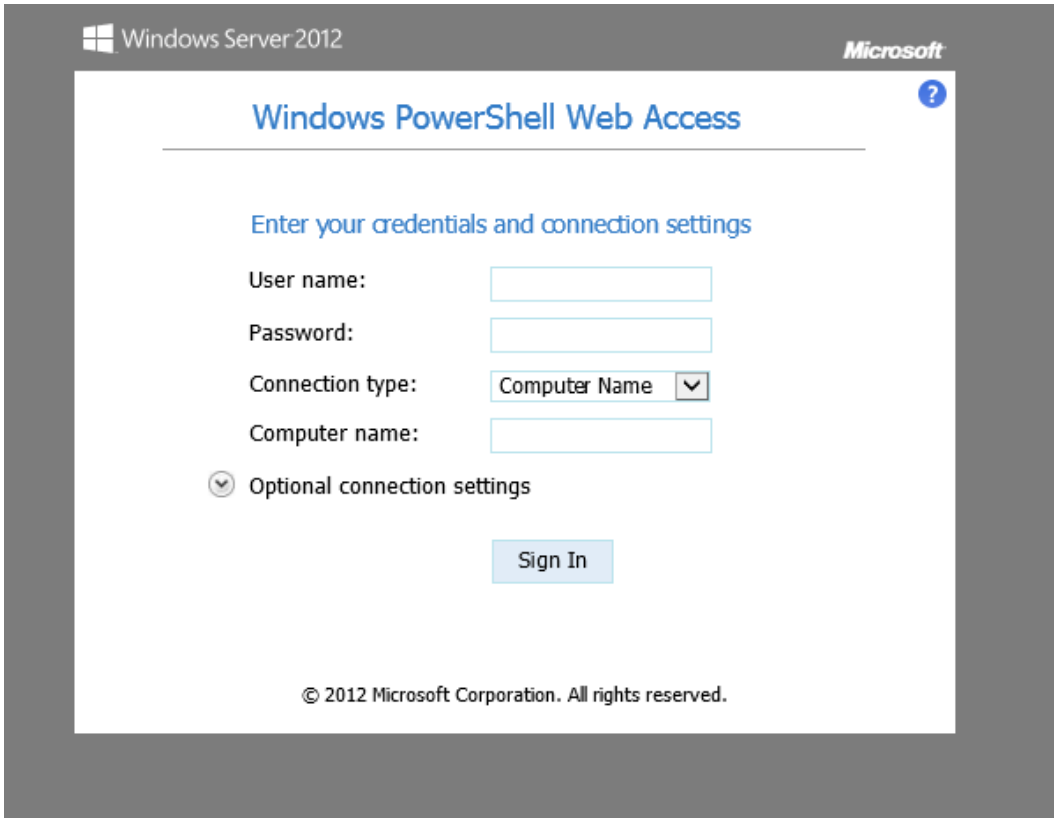
Avviato il comando verrà restituito un messaggio di conferma e da quel momento in avanti si potrà testare la console Web. I Browser compatibili con PowerShell Web Access sono i seguenti:

- Internet Explorer 8.0, 9.0, 10.0
- Mozilla Firefox 10.0.2
- Google Chrome 17.0.963.56m for Windows
- Apple Safari 5.1.2 for Windows
- Apple Safari 5.1.2 for Mac OS

I requisiti che questi browser devono avere sono i seguenti:

- Allow cookies from the PowerShell Web Access
- Be able to open and read HTTPS pages
- Open and run websites that use JavaScript

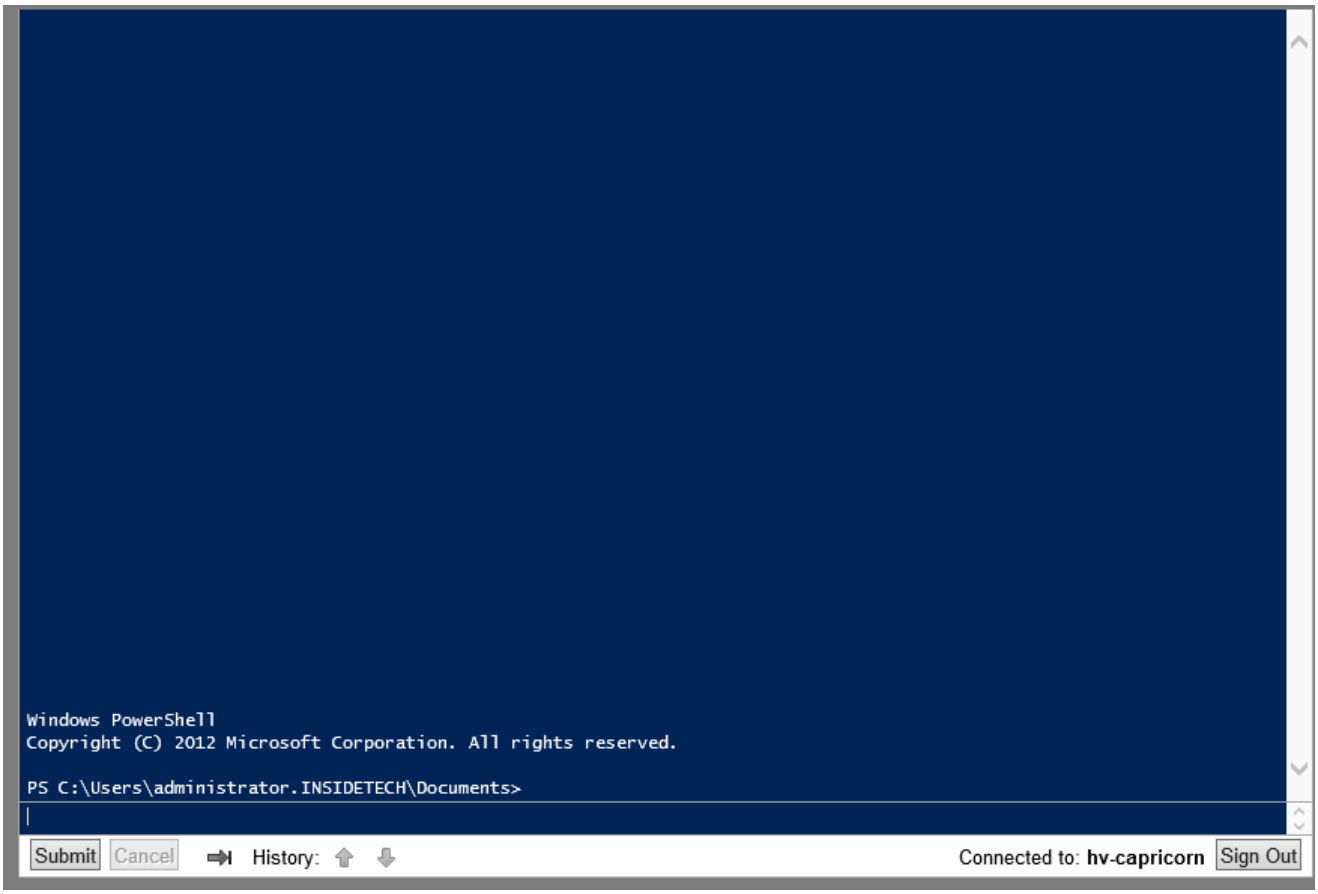
Lanciando il sito web del Web Access comparirà una schermata come quella mostrata nella prossima figura, dove inserire username, password e computer al quale ci si vuole collegare.



esempio di PS Web Access

Come detto in precedenza, è necessario selezionare una macchina ed un utente che sono stati abilitati.

Se le operazioni fatte in precedenza sono state fatte in modo corretto, si avrà una schermata come quella della successiva immagine.



Sicuramente arrivati a questo punto, qualcuno sarà sorpreso di quello che si potrebbe fare con questa console: basta uso dei Remote Desktop, basta con l'uso di giri particolari per avere una console. Uno strumento facile e veloce da implementare per gestire la propria infrastruttura.

- Operare su file e cartelle per strutturare e organizzare l'archivio (viene lasciato al lettore)
- Elementi e caratteristiche dell'interfaccia grafica del Sistema Operativo (viene lasciato al lettore)

L'avvio del computer: il bootstrap

Il bootstrap è la prima fase di inizializzazione del sistema, che avviene in automatico quando si agisce sul pulsante di accensione.

La sequenza di bootstrap è fissa ma usa delle impostazioni che possono essere customizzate dal proprietario del computer.

Dopo la lettura ed esecuzione dell'area ROM si passa al test delle periferiche di base.

In generale, la prima cosa eseguita, dopo l'accensione in modalità minima della scheda video, è l'interrogazione del chip che gestisce il bus della memoria RAM.

La RAM viene gestita da un apposito chipset, detto northbridge, e durante la fase di bootstrap viene interrogata per vedere se tutti i suoi segmenti sono operativi, tramite una procedura automatica di lettura e scrittura.

Dopo il test della RAM avviene l'interrogazione della presenza delle unità di massa e una volta rilevate in queste la presenza di un boot sector, da cui caricare la tabella di allocazione dei files, la FAT, che nel tempo ha subito migliorie tecnologiche adeguandosi agli odierni sistemi operativi.

Vengono accesi i canali di comunicazione dopo che sono stati caricati i driver delle schede ethernet o WiFi in modo che il PC possa accedere alla rete.

Alla fine di questi basilari processi di caricamento e test di operatività viene mostrato il desktop e il computer è pronto per l'utilizzo.

Il desktop e le icone.

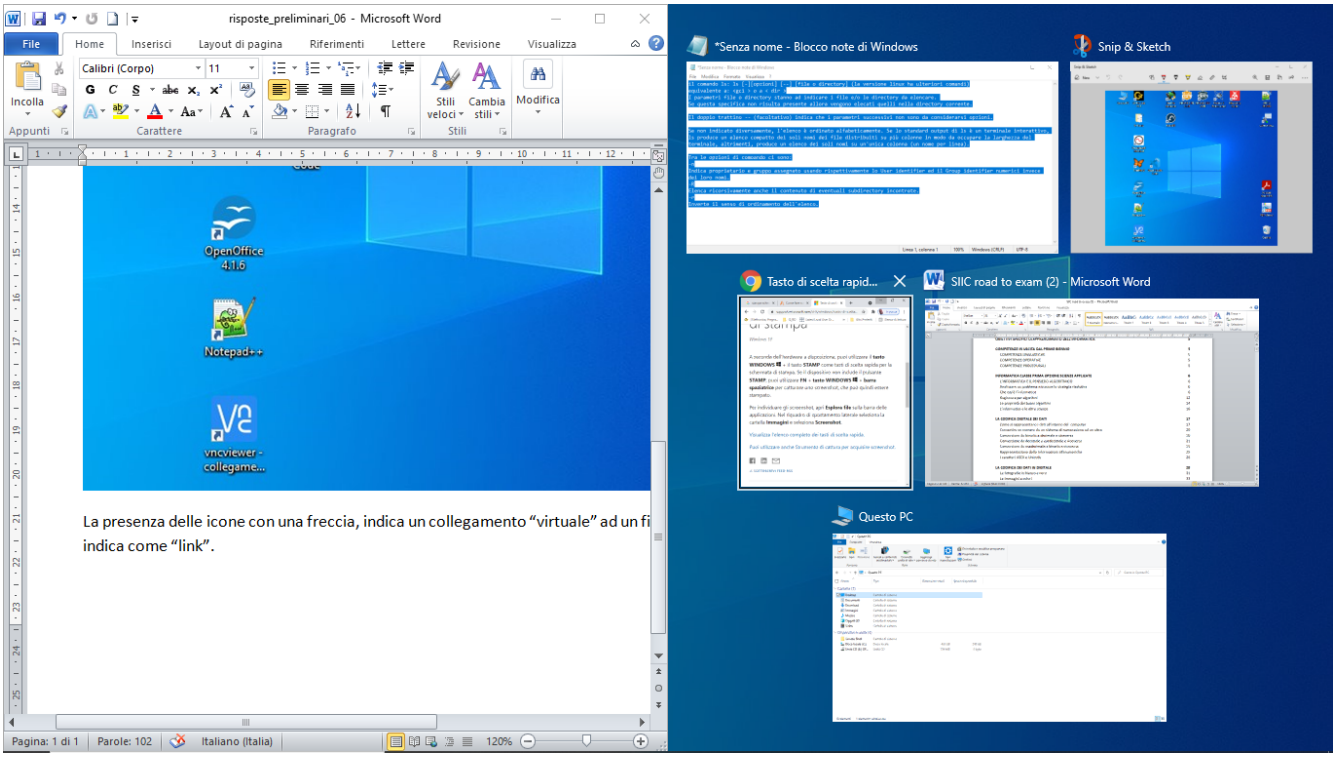
Il termine “desktop” viene utilizzato per indicare il piano virtuale della scrivania del computer, dove ci sono le cartelle e file o i programmi, che si tengono a “portata di mano”, quelli insomma che devono risultare immediatamente accessibili, che con l’avvento del sistema di rappresentazione iconografico introdotto negli anni ’80 da APPLE con il famoso Macintosh ha rivoluzionato la gestione “a riga di comando” che era allora tipica di tutte le altre case di software.

La presenza delle icone con una freccia, indica un collegamento “virtuale” ad un file che in termine tecnico si indica come “link”.

Gli elementi delle finestre

Nel desktop le icone quindi, quando attivate, possono assumere vari stati, che dipendono dalle azioni che si stanno facendo.

Quando un programma viene attivato compare la sua “finestra”, ovviamente, ci possono essere più di un programma che si trova in una certa condizione lavoro, poiché, i moderni computer hanno potenzialità “multitasking”, ovvero, possono operare contemporaneamente su più processi.



In particolare pero, le finestre hanno la possibilita di far assumere al programma in esecuzione degli stati particolari, usufruento solo di un tocco del mouse su dei simboli appositi:

Dove se si clicca sul segno "-" la finestra si "minimizza" riducendosi ad "icona", se invece si clicca sulla "X", il programma si chiude, mentre se si clicca sul simboletto delle "cartelline", il programma riduce la sua finestra, (questo simbolo cambia forma quando la finestra ha la massima visibilita a schermo):

File e cartelle

Solitamente su "desktop" vi dovrebbero essere solo i link a dei programmi, il cestino e il programma di gestione dei file, a volte comunque ci sono anche delle cartelle, magari in occasione di qualche operazione di copiatura o scaricamento di file, od altre operazioni temporanee, ma, di norma, ci andrebbe solo la cartella di lavoro, ad esempio, in alcune impostazioni standard Linux, nel desktop c'è soltanto il cestino e la cartella "Home", che e quella principale.

Le cartelle quindi sono dei contenitori di files, ma possono anche contenere ulteriori cartelle, in una struttura "gerarchica", cosiddetta "ad albero", denominata anche "directory"

Alcune di queste cartelle possono anche trovarsi in una particolare condizione: quella compressa, in questo caso sono paragonabili a dei file (vedasi paragrafo su compressione dei dati), mentre altre potrebbero anche essere "nascoste".

Estensione dei file

Le cartelle quindi vanno viste come dei contenitori di file eterogenei, nel senso che questi file possono essere ben diversi tra loro, dai file di testo, a quelli contenenti immagini o video, o file audio, o file di tantissimi altri tipi comunque sia generati da qualche "file programma", quindi la differenza fondamentale tra le cartelle ed i file e quella per cui le cartelle hanno dei nomi, ma non hanno nessun tipo di estensione, mentre tutti i tipi di file hanno una loro propria caratteristica identificativa che viene scritta dopo il punto posto immediatamente dopo il nome, questa sigla, che si limita a tre o quattro lettere, viene denominata "estensione" o suffisso, in particolare, i "file programmi" che sono direttamente eseguibili sul computer hanno la classica estensione "nome.**exe**".

Le estensioni, o suffissi, solitamente vengono direttamente associate dal sistema operativo al programma che le ha generate, e pertanto, nella visualizzazione iconografica, i file associati ai programmi hanno tutti delle icone personalizzate, mentre se il file non viene riconosciuto dal sistema operativo ha una icona generica e se viene richiamato in esecuzione con un doppio click, il sistema operativo richiede di identificare un file genitore… che sia in grado di poter eseguire delle operazioni sui dati contenuti, solitamente viene chiesto con quale applicazione aprire il file da locale o viene proposta una ricerca dell'applicazione più adeguata da scaricare dalla rete.

Alcuni esempi di suffissi:

.txt per file di testo;

.MP3, .ogg o .WAV per file di tracce sonore (audio);

.jpg, .png, .bmp, .gif o .psd, per alcuni formati di immagini digitali statiche;

.avi, .mpeg, .wmv, .mp4, .3gp, .flv per file di immagini digitali in movimento (video).

.exe, .com, .bat e .cmd per programmi e script eseguibili in Windows;

.htm, .html, .shtml, .shtm, .stm, per pagine web statiche;

.asp, .aspx, .php o .dtw per pagine web dinamiche o script;

.doc .docx per file generati dal programma di videoscrittura Microsoft Word;

.odt per file generati dal programma di videoscrittura LibreOffice Writer;

.ods per file generati dal programma di foglio di calcolo LibreOffice Calc;

.odp per file generati dal programma di presentazioni LibreOffice Impress;

.odb per file generati dal programma di database LibreOffice Base;

.xls e .xlsx per file contenenti fogli di calcolo generati con Microsoft Excel;

.xml per file scritti in linguaggio eXtensible Markup Language;

.pdf per documenti di tipo PDF;

.xps per file scritti in XML Paper Specification;

.zip o .rar per file compressi;

.dwg, dxf per programmi di disegno CAD;

I sistemi di archiviazione

I dati, di qualsiasi natura informatica , siano essi programmi o files, vanno archiviati nelle cosiddette memorie di massa, anche denominate memorie secondarie, in quanto non sono direttamente utilizzate dalla CPU durante l'elaborazione del programma.

Si distinguono dalle memorie che invece vengono utilizzate durante l'elaborazione dal processore come le ROM la RAM, e la cache di sistema, chiamate le memorie primarie.

Vengono considerate come memorie di massa, quei componenti di archiviazione digitale utilizzati dai computer ma anche da altri apparecchi elettronici per registrare, conservare e rileggere i dati anche dopo la riaccensione del device, cosa che ovviamente non avviene per i dati conservati nelle RAM.

La memoria di massa fissa è integrata nel computer o nell'apparecchio elettronico. Non può essere facilmente rimossa dall'utente salvo applicazioni particolari e a computer spento. Un esempio di memoria di massa di questo tipo è il disco fisso (hard disk) dei computer.

Solo in server che dispongo di particolare Array di dischi ridondanti (RAID) è possibile lo swap o la rimozione a caldo, al fine della sostituzione, di hard disk che fanno parte dell'array.

Memoria di massa rimovibile. La memoria di massa rimovibile può essere facilmente rimossa e separata dal computer. Alcuni esempi di memorie di massa rimovibili sono le pennette Usb (memory key), i dvd-rom o i cd-rom, HD con interfaccia USB ecc.

Con il crescere delle applicazioni digitali, la necessità di disporre di memorie di massa di notevoli dimensioni, ha indotto ad introdurre un sistema di archiviazione distribuito denominato NAS (Network Attached Storage) che è un dispositivo di archiviazione intelligente connesso alla rete domestica o professionale.

Generalmente i NAS sono attrezzati per fungere da computer attrezzati per poter comunicare via rete. Pertanto, non sono dei semplici dispositivi di memorizzazione di dati da connettere via RJ-45 a dei computer, ma sono, a tutti gli effetti, degli storage il cui scopo è mettere a disposizione spazio multi-disco in rete in maniera "intelligente" ovvero interfacciabile, gestibile e collegabile ad altre risorse di rete, in particolare i server su cui sono installati le applicazioni e il Domain controller.

Nota: possono essere utilizzati come NAS dei dispositivi di archiviazione gestiti da "piccoli" SBC (dall'inglese: single-board computer) Si tratta di dispositivi dotati solitamente di un sistema operativo basato su Linux (generalmente trasparente all'utente) e di diversi hard disk destinati all'immagazzinamento dei dati.

Esistono anche dei sistemi di archiviazione "pubblici", denominati CLOUD. Il "cloud computing" (in italiano nuvola informatica sta ad indicare, un servizio di erogazione offerto su richiesta da un fornitore a un cliente, attraverso la rete internet, come l'archiviazione, l'elaborazione o la trasmissione dati, a partire da

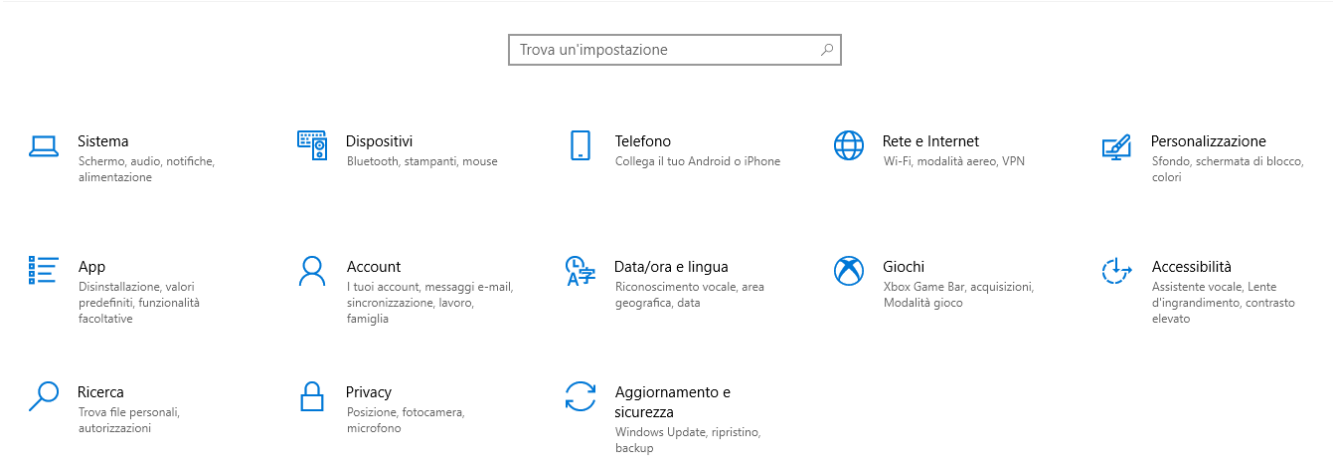
un insieme di risorse preesistenti, configurabili e disponibili in remoto sotto forma di architettura distribuita, tipicamente, per esempio tutti i servizi e-mail, vengono gestiti in questo modo, ma ultimamente sono molti i servizi utilizzati anche per la gestione delle immagini fotografiche.

Per la tipologia della gestione dei sistemi di archiviazione e trasmissione dei dati vedasi anche il paragrafo dedicato alla compressione dei dati.

Il pannello di controllo

Il pannello di controllo è lo strumento che viene utilizzato per avere accesso a tutte le impostazioni del sistema operativo, quindi, ogni sistema operativo avrà un suo particolare pannello di controllo.

Per esempio, in Windows 10 si avrà una schermata di questo tipo:

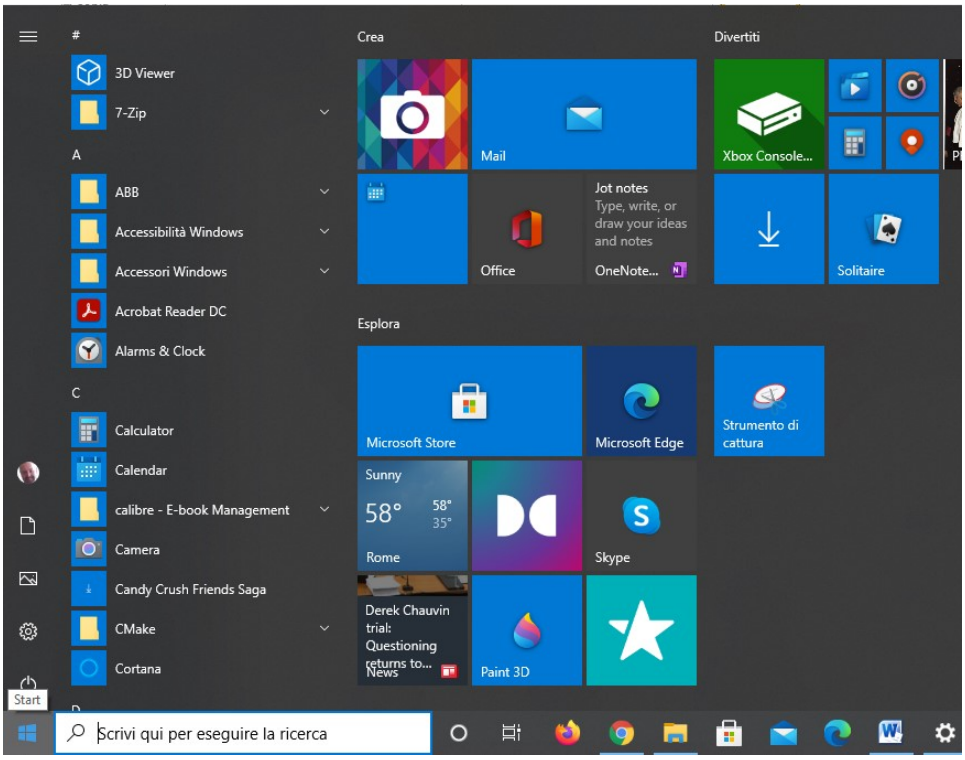


Permette le configurazioni hardware delle parti interagenti che formano il sistema.

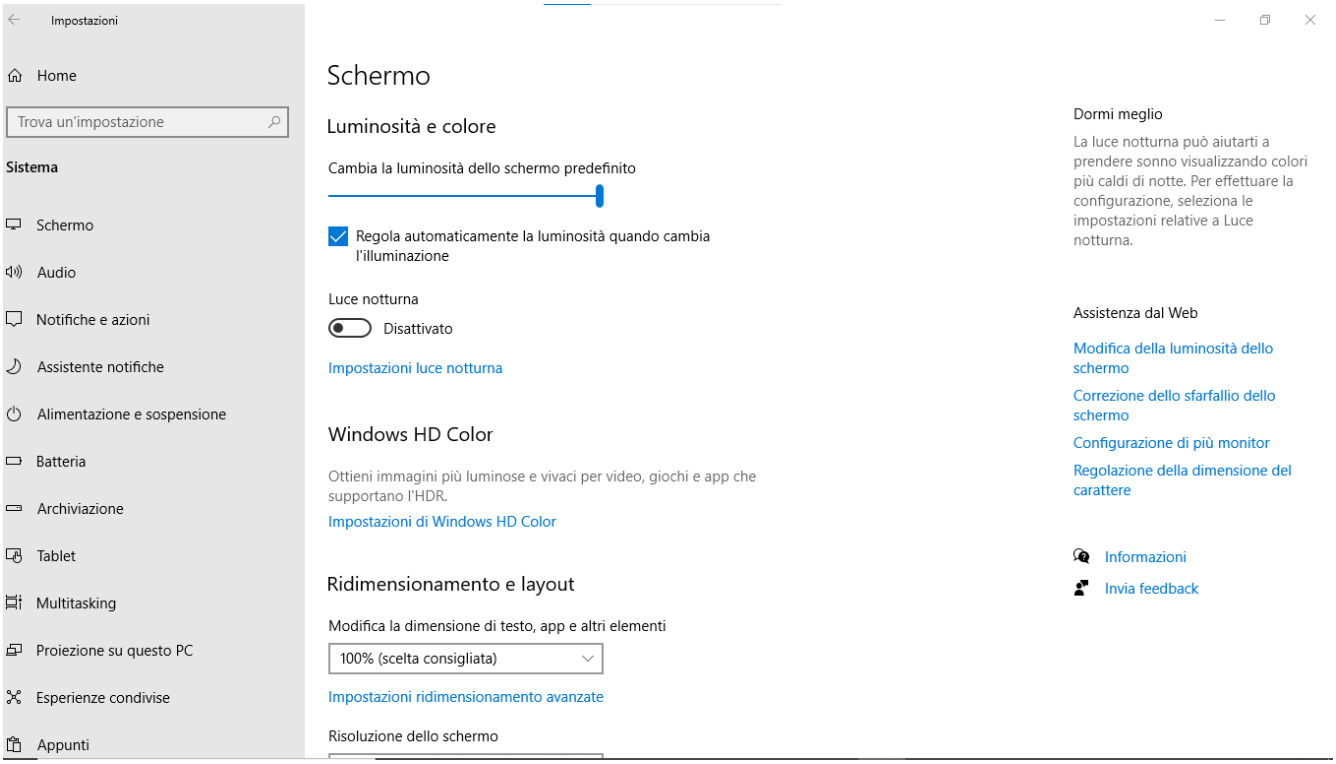
Dal pannello di controllo si possono installare i driver che sono gli strumenti software che permettono il corretto funzionamento del device.

In primo luogo dal pannello di controllo si assegnano ai dispositivi i canali di interrupt e le aree di memoria che i processi andranno ad utilizzare durante l'esecuzione.

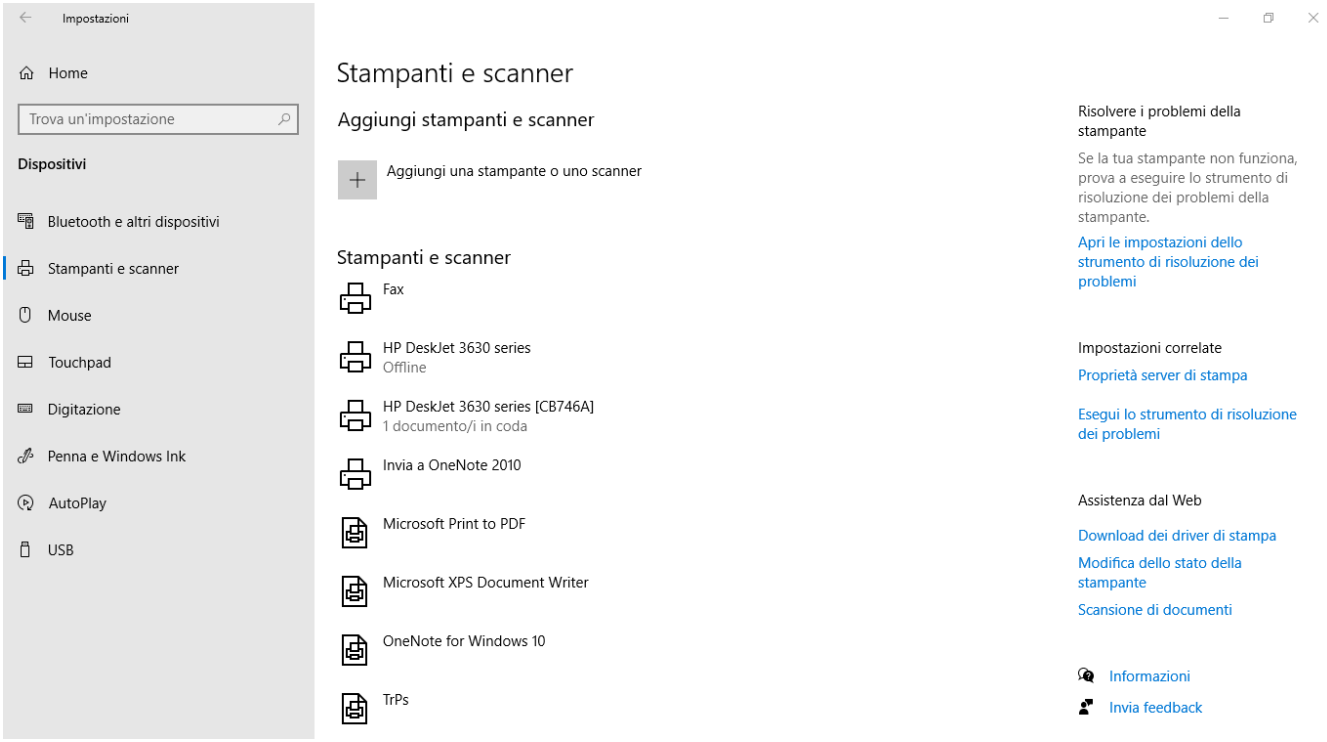
Cliccando sulla icona "ingranaggio" accessibile tramite l'icona di "start":



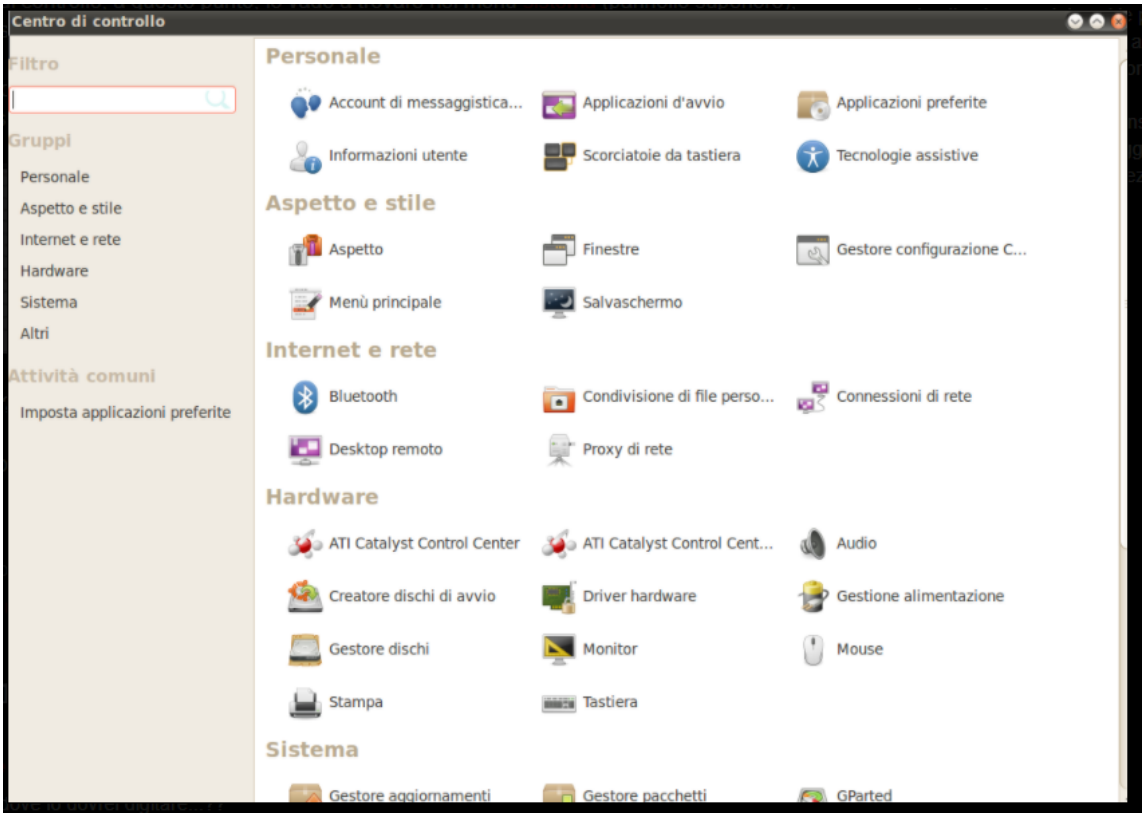
Come si può notare, la finestra del pannello di controllo contiene molti menu, ognuno dei quali apre delle specifiche finestre da dove si può accedere alle varie impostazioni, come nell'esempio illustrato dalla seguente immagine per il menu "sistema":



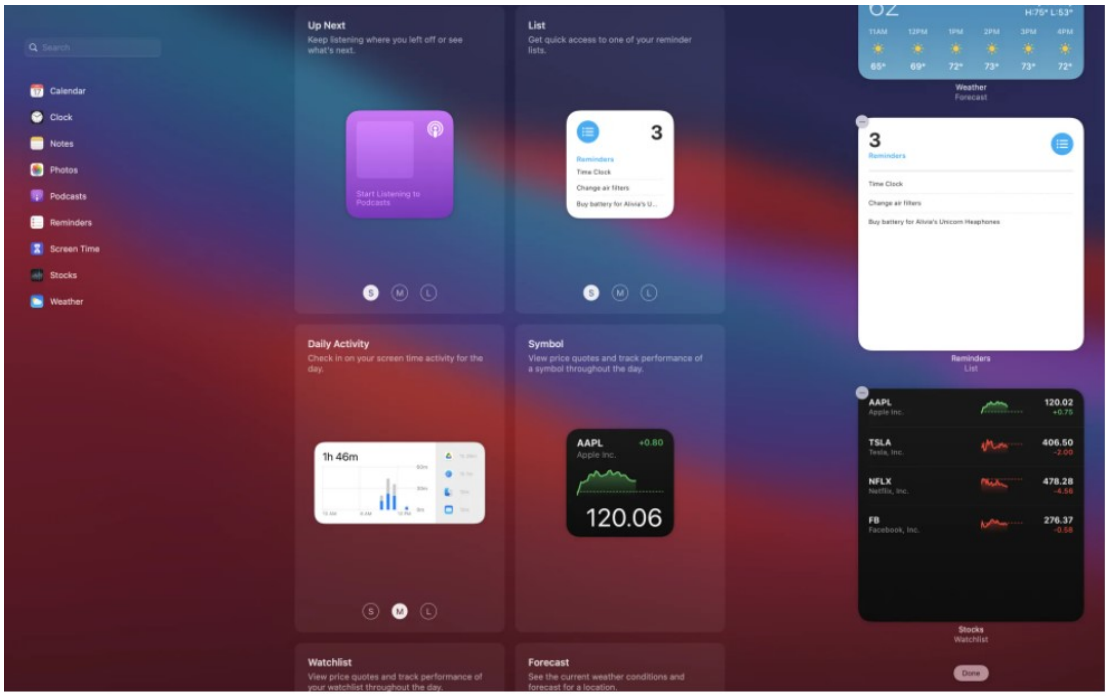
Oppure per il menu "dispositivi":



Altri sistemi operativi ovviamente hanno una interfaccia grafica (GUI) che sarà diversa, ma fondamentalmente avrà le stesse funzioni, come illustrato dalla seguente immagine che si riferisce ad un sistema operativo Linux Ubuntu:



O per il “centro di controllo” macOS”:



I sistemi operativi per i vari dispositivi

In informatica, con il termine “Sistema operativo” si intende un insieme di programmi in grado di gestire le funzionalità di un computer, quindi, un sistema operativo ha necessariamente la caratteristica di essere letteralmente “legato” al suo Hardware originale, ma nel corso del tempo, e con l’evoluzione dei componenti e delle tipologie di software, è possibile “simulare” un sistema operativo in condizioni “virtuali”, quindi, per esempio si può installare un sistema Linux o Windows su una macchina macOSX e viceversa., ovviamente, l’unica condizione è che l’hardware lo consenta.

Di derivazione Linux, in maniera più o meno diretta sono alcuni dei sistemi operativi per gli smartphone, ad esempio Android.

Alcuni dispositivi che stanno a metà tra l’essere un computer o un sistema di controllo, ad esempio il Raspberry, hanno il sistema operativo di derivazione Linux, montato su una scheda SD da cui avviene il bootstrap.

La caratteristica principale del sistema operativo è quella di rimanere sempre caricato, dall’accensione della macchina fino al suo spegnimento (o la sua disattivazione). Il sistema operativo prende in carico tutto il traffico dati all’interno del computer e tra la macchina e le sue periferiche e opera come intermediario tra l’hardware e i vari programmi.

Componenti di base di un sistema operativo :

Un sistema operativo è composto nelle sue funzioni basilari da:

– Un Kernel: che costituisce il motore del sistema operativo, ovverosia un software specifico che fornisce ai moduli che compongono il sistema le istruzioni necessarie per svolgere le proprie funzioni, in alcuni casi mantenendo e gestendo solamente le funzionalità essenziali: in questo caso si parla di **microkernel**.

– Il File system: è l'insieme dei tipi di dati astratti necessari per la memorizzazione (scrittura), l'organizzazione gerarchica, la manipolazione, la navigazione, l'accesso e la lettura dei dati.

Esistono diversi file system, creati per vari sistemi operativi o apparecchiature di memorizzazione, i file system definiscono, ad esempio, le seguenti proprietà:

- convenzioni sui nomi di file
- attributi dei file
- controllo di accesso

I supporti di memoria possono essere formattati nei diversi file system in relazione ai diversi tipi di sistema operativo, i principali formati di memorizzazione dei dati sono:

Nome	Campo di applicazione	Sistema operativo (supporto)	Particolarità
FAT32	Supporti mobili	- Windows - macOS X/macOS - Linux (richiede eventualmente l'installazione di driver)	- Compatibilità elevata - Ampio supporto hardware - Nessuna funzione di crittografia e compressione - Sicurezza dei dati non particolarmente sviluppata - Ideale per partizioni più piccole - Dimensione massima dei file: 4 GB
exFAT	Supporti mobili	- Windows - macOS X/macOS (compatibile dalla versione 10.6.4) - Linux (richiede eventualmente l'installazione di driver)	- Non ancora uno standard generale - Nessuna gestione dei diritti - Nessuna compressione dei dati - Ideale per piccole memorie flash da 32 GB (chiavette USB, schede SD) - Dimensioni e partizioni illimitate (in base

			all’attuale stato dell’arte) - Dimensione massima dei file: 512 terabyte
NTFS	Dischi rigidi interni ed esterni	- Windows - macOS X/macOS (supporto completo solo con strumento aggiuntivo) - Linux (con l’installazione del driver)	- Gestione dei diritti - Miglioramento della sicurezza dei dati: protezione contro la perdita e le modifiche, possibile crittografia dei dati - Compressione dei dati possibile - Prestazioni elevate su supporti dati di grandi dimensioni - Specializzato in file di grandi dimensioni e grandi capacità di archiviazione - Non adatto per unità e partizioni di dimensioni inferiori a 400 MB (gestione troppo complessa) - Dimensione massima dei file: 256 TB
APFS	Unità SSD	- macOS (standard da versione 10.13, High Sierra) - Software aggiuntivo per l’utilizzo con versioni macOS e Windows meno recenti	- Ottimizzato per unità di memoria a stato solido (SSD) e altri dispositivi di archiviazione all-flash - Compatibile anche con unità meccaniche e ibride - Crittografia dei dati possibile - Gestione ottimizzata dello spazio di archiviazione (funzione di condivisione dello spazio) - La funzione di protezione dai crash protegge da eventuali guasti del file system (ad esempio in caso di crash del sistema) - Supporto Fusion Drive da macOS 10.14 Mojave - Dimensione massima dei file: 8 Exbibyte
HFS+	Dischi rigidi interni ed esterni	macOS X/macOS	- File system maturo e collaudato - Particolarmente adatto per unità meccaniche - Non ottimizzato per le moderne tecniche di archiviazione (SSD, Flash) - Migliore compatibilità con le versioni precedenti rispetto a APFS

			- Durata limitata, probabilmente non più supportato da Apple nel lungo periodo - Sempre meno importante a causa della "conversione forzata", parzialmente automatizzata, in APFS - Dimensione massima dei file: 8 Exbibyte
ext4	Linux	- Linux - Windows (solo con strumento aggiuntivo) - macOS X/macOS (solo con strumenti aggiuntivi)	Rispetto alle versioni precedenti di ext: - Miglioramento delle prestazioni - Miglioramento della sicurezza dei dati - Crittografia integrata (da Linux Kernel 4.1) - La nuova funzione Extents offre vantaggi in termini di velocità durante la gestione di file di grandi dimensioni e previene la frammentazione - Gestione dei diritti possibile - Dimensione massima dei file: 16 TB

– Lo Scheduler: fondamentale per sistemi operativi in multi task, si occupa di portare avanti un processo, interrompendone temporaneamente un altro.

– Il Gestore della memoria: che fondamentalmente è un gestore di puntamento che si occupa di distribuire la memoria tra i processi attivi regolandone l'accessso, evitando delle sovrapposizioni nell'uso della memoria, e liberandone quella occupata da processi che hanno subito standby o altre condizioni operative .

– **Lo Swapping**: ovvero, un sistema con il quale è possibile attivare un'area di memoria per un nuovo processo, scaricando quello già in uso. Viene riservata un'area o percentuale di memoria appositamente a questi processi, che si chiama appunto swap e che funziona un po' come la Ram del computer.

– **Il Gestore di memoria virtuale**: si occupa di mappare la memoria virtuale che viene messa a disposizione dei programmi, sulla memoria fisica e sui dischi rigidi del sistema (quello che permette anche la simulazione di un altro sistema operativo).

– Il controllo della memoria: un modulo che si occupa di controllare e proteggere la memoria da accessi indesiderati, è un sistema posto a tutela della conservazione dei dati.

– **L'Interfaccia utente**: ovvero, un programma (GUI) "Graphical User Interface", che consente all'utente di interagire con la macchina. Sono rimaste operative anche le interfacce a linea di comando (utilizzate solitamente da remoto).

– Lo Spooler di stampa: il software che serve a risolvere il problema della gestione dei documenti inviati alla stampante. Lo spooler accumula i dati inviati alla stampa in un'apposita aerea di memoria, facendosi poi carico di gestire interamente il processo, lasciando così liberi gli altri programmi di continuare a funzionare, senza subire rallentamenti.

-Il Task Manager: uno modulo che contiene strumenti di **monitoraggio** GPU dettagliati, dove è possibile visualizzare l'utilizzo della GPU per applicazione e a livello di sistema.

Come abbiamo accennato sopra, vi sono diversi sistemi operativi, in particolare, i più diffusi sonoç

- **Windows di Microsoft**
- **MacOs di Macintosh**
- **Linux**
- **Unix**

I sistemi Unix e Linux sono molto simili tra di loro. Si differenziano solamente dal fatto che Linux gira sui PC, mentre Unix ha bisogno di macchine più potenti. Entrambi degli open source, ovvero non sono il prodotto di un'azienda ma vengono implementati da una comunità di ricercatori. Per questo motivo sono completamente gratuiti.

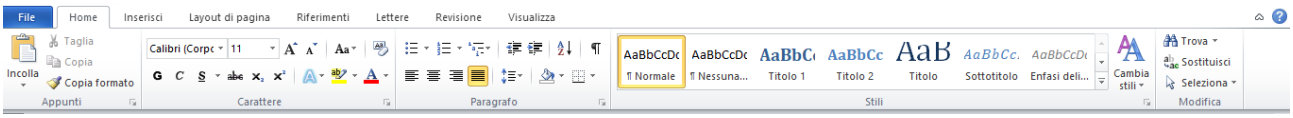
All'acquisto un computer non è in grado di funzionare. Può al massimo accendersi, eseguire il boot e rilevare l'assenza di un sistema operativo. Il sistema operativo è quindi la componente primaria di un pc, risiede nell'hard disk o USB e, al momento dell'accensione della macchina, viene caricato nella memoria RAM.

Elaboratore testi

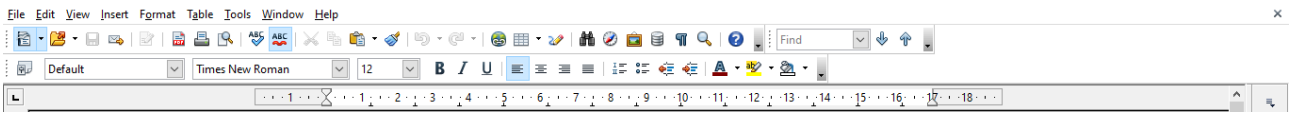
I programmi che assolvono il compito di elaborare dei testi, solitamente appartengono a delle software suites che sono delle collezioni di software, ovvero, programmi orientati dei compiti specifici, solitamente queste collezioni prendono il nome di “office automation”, tutti conoscono la “piattaforma” Microsoft Office, per esempio, forse, alcuni conoscono anche “Libre Office” o “Open Office”, che in qualche modo assolvono al compito di accomunare dei programmi per la gestione dei testi, per la gestione dei calcoli tabellari, per la gestione delle presentazioni, per la gestione di grafi, per la gestione delle mail ecc.

Creare, salvare, aprire, modificare, correggere

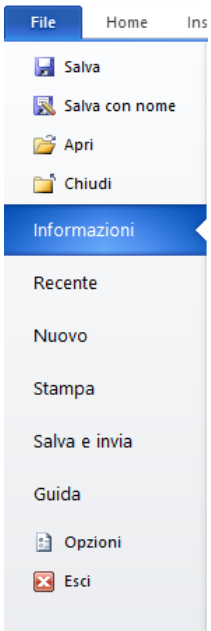
Editare un file significa Crearlo, salvarlo, aprirlo, modificarlo, correggerlo. Quasi tutti i software per la gestione del testo hanno una “barra dei menu” simile tra loro, nella immagine riportata di seguito ad esempio si può osservare quella di Microsoft Word:

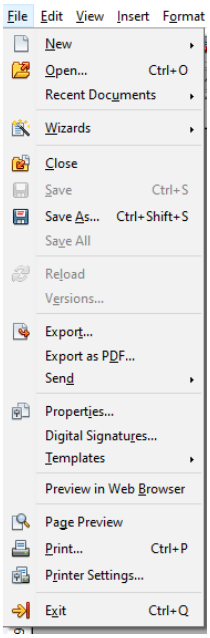


In questa invece viene rappresentata quella di OpenOffice:

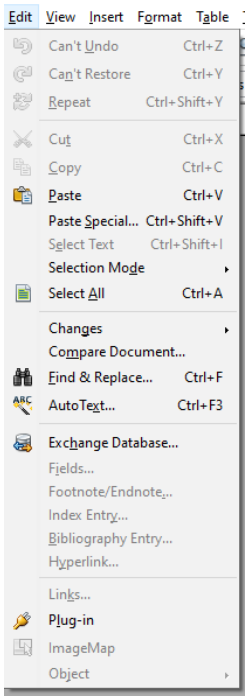


Ovviamente, vi sono delle diversità, ma solitamente nel menu a sinistra vi sono i comandi per poter aprire dei file esistenti o per crearne di nuovi, per salvarli in una directory opportuna, fornendo loro un nome appropriato ed una appropriata estensione identificativa, per eseguire i settaggi di stampa, le anteprime e le stampe, per effettuare eventuali importazioni di formati compatibili, per effettuare anche delle possibili esportazioni su formati diversi da quello standard previsto per il prodotto, ed infine per chiudere e salvare i programmi.





stampare e chiudere un file



Solitamente il secondo menu, a partire dalla sinistra, e dedicato alle operazioni più comuni per la modifica e la formattazione del testo, si intende dire che tutti i programmi più comuni utilizzano al formula “taglia”, “copia”, “incolla” che sono le principali operazioni, per le modifiche, poi, nello stesso menu, sono presenti molte altre potenzialità di lavoro quali ad esempio comandi di utilità, quali ad esempio il comando “seleziona tutto”, il comando “cerca” e “cerca e sostituisci”, il comando “incolla in formato speciale”… ed altre varie alternative di gestione.

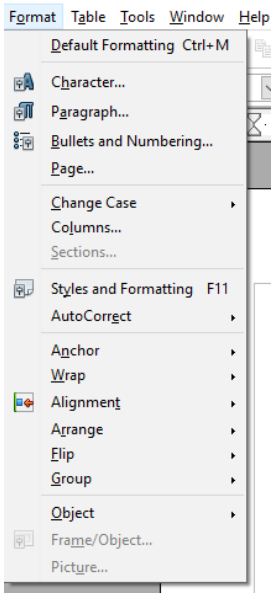
Applicare le procedure operative per la formattazione di base del testo

La formattazione del testo, negli anni, ha subito delle ulteriori potenzialità, non solo per esigenze puramente estetiche, ma in soprattutto, per esigenze di impaginazione, e di indicizzazione, in quanto, come tutti sappiamo, i testi sono, in ultima analisi, “un elenco organizzato di parole ed altre cose…” che si possono stampare, ma anche riassumere, indicizzare… tabellare…

Quindi, l’automazione introdotta dai software moderni, consiste appunto nel rendere il piu facile possibile tutte queste elaborazioni.

Come si vede dall’esempio in immagine, e possibile organizzare il testo in “titoli gerarchici” in relazione alla loro funzionalità all’interno della strutturazione del contesto: titolo principale, titolo di capitolo, sottotitolo, titolo di paragrafo, enfasi di testo ecc.

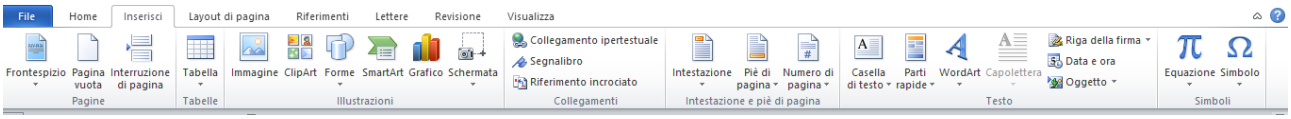
Formattare i documenti con elenchi, bordi e sfondi

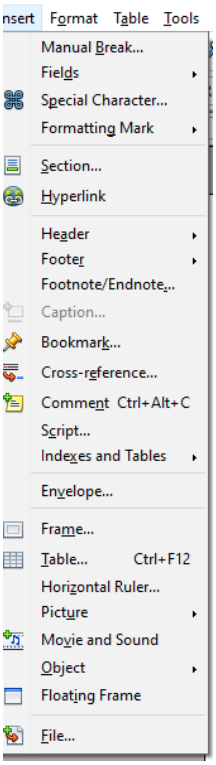


Solitamente, vi e un menu dedicato alla configurazione di stampa delle pagine definito coma “layout di pagina”, all’interno del quale si trovano gli strumenti dedicati alla “formattazione di stampa”, come quelli che definiscono gli spazi da lasciare sia per i bordi, bordo destro e bordo sinistro, sia per l’intestazione che per il pie di pagina, la suddivisione in colonne, la formattazione dei paragrafi, ma anche per la definizione della filigrana dello sfondo, o il colore, la definizione del rientro del testo, la spaziatura tra le interlinee, l’ancoraggio e la posizione di riferimento per le immagini e molte altre funzionalità che sono necessarie per la rappresentazione a schermo e su carta del file prodotto.

Inserire e gestire : oggetti grafici, immagini, forme e caselle di testo.

Contestualmente, solitamente nel terzo menu, ci sono i comandi di gestione del contesto di impaginazione, come si può osservare dalle seguenti immagini:





Con il termine gestione del contesto di impaginazione, vengono raggruppati tutti quei comandi che permettono di "configurare i contenuti della pagina", almeno nelle sue componenti principali:

- **Intestazione**
- **Corpo del testo**
- **Pie di pagina**

Ma, come si può osservare dai menu di esempio, le "utility" fornite sono molte di più, permettono l'inserimento di immagini, tabelle, simboli e caratteri speciali, formule, riferimenti ed indici, sommari, link a file esterni o ad indirizzi on-line, commenti ecc.

Altri elaboratori testuali

Oltre che ai programmi per l'elaborazione del testo solitamente utilizzati nella "office automation", vi sono pero dei programmi, che vengono utilizzati solitamente dai programmatori, per l'elaborazione del testo di programmazione o per operazioni condotte su software da postazioni remote: sono elaboratori di testo essenziali che si possono utilizzare da "terminale" come ad esempio "**Vim**", (Vi IMproved) che è un editor di testo avanzato altamente configurabile. Questo software è una versione migliorata dell'editor "vi" distribuito sui più importanti Sistemi Operativi Linux e Mac. Il programma è utile ai programmatori, a chi necessita di un semplice word processor per la scrittura di documenti, di email o per la modifica dei file di configurazione. Disponibili funzioni per la formattazione e l'allineamento del testo.

Vi sono inoltre altri software particolarmente adatti alle fasi di elaborazione " da linea di comando" come "nano": che e una evoluzione di un software unix che si chiama "Pico" e che viene distribuito con licenza GNU GPL Su sistemi Unix ed Unix-like.

- Le applicazioni per il word processing (la risposta è lasciata al lettore)

Le azioni di base sui file e sul testo

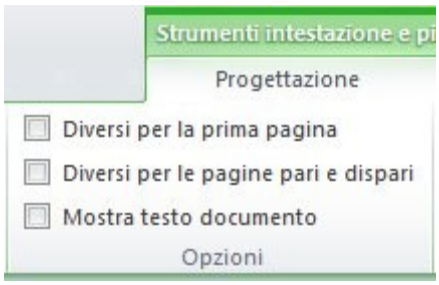
È possibile iniziare la numerazione in una pagina del documento diversa dalla prima.

Inizio della numerazione dalla seconda pagina

Fare doppio clic sul numero di pagina.

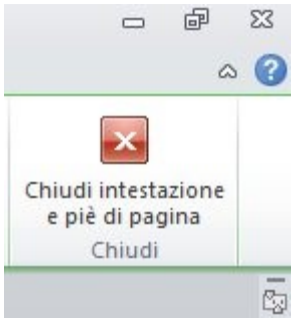
Verrà aperta la scheda **Progettazione** in **Strumenti intestazione e piè di pagina**.

Nel gruppo **Progettazione** della scheda **Strumenti intestazione e piè di pagina** selezionare la casella di controllo **Diversi per la prima pagina**.



Per iniziare la numerazione da 1, fare clic su **Numero di pagina** nel gruppo **Intestazione e piè di pagina**, quindi su **Formato numeri di pagina** e infine su **Comincia da** e immettere **1**.

Per tornare al documento, fare clic su **Chiudi intestazione e piè di pagina** nella scheda **Progettazione** in **Strumenti intestazione e piè di pagina**.



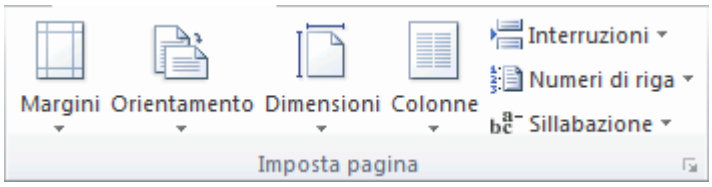
Inizio della numerazione da una pagina diversa

Per iniziare la numerazione da una pagina diversa, anziché dalla prima pagina del documento, è necessario aggiungere un'interruzione di sezione prima della pagina da cui si desidera iniziare la numerazione.

Fare clic all'inizio della pagina da cui si desidera iniziare la numerazione.

Per essere certi di trovarsi all'inizio della pagina, premere HOME.

Nel gruppo **Imposta pagina** della scheda **Layout di pagina** fare clic su **Interruzioni**.



In **Interruzioni di sezione** fare clic su **Pagina successiva**.

Fare doppio clic nell'area dell'intestazione o del piè di pagina, in prossimità del punto superiore o inferiore della pagina.

Verrà aperta la scheda **Strumenti intestazione e piè di pagina**.

Nel gruppo **Spostamento** della scheda **Strumenti intestazione e piè di pagina** fare clic su **Collega a precedente** per disattivare il collegamento.

Seguire le istruzioni per l'aggiunta di un numero di pagina o per l'aggiunta di un'intestazione o un piè di pagina con un numero di pagina.

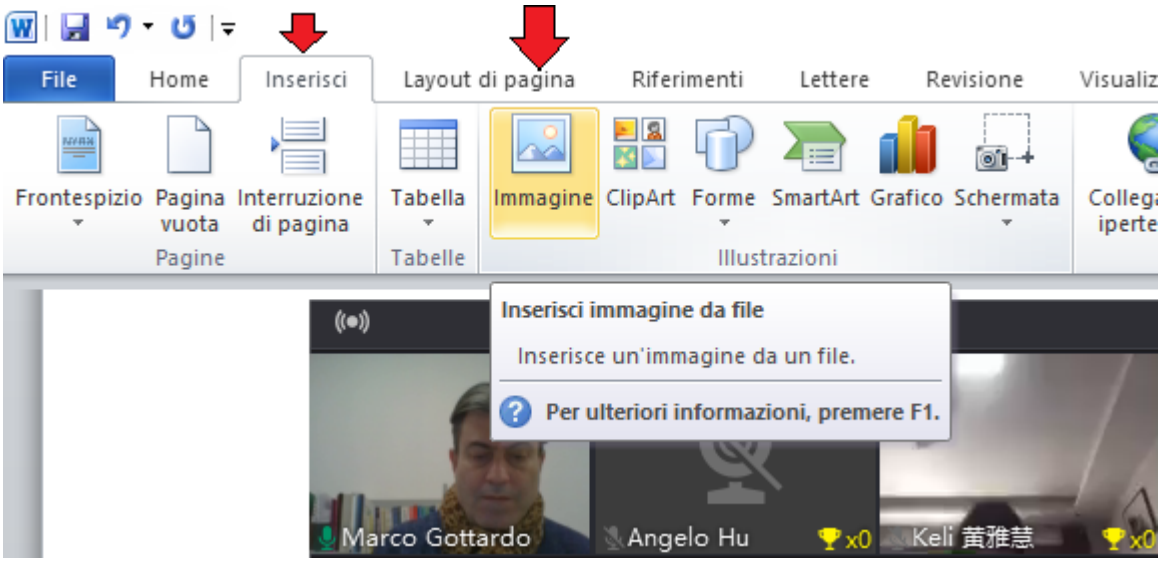
Per iniziare la numerazione da 1, fare clic su **Numero di pagina** nel gruppo **Intestazione e piè di pagina**, quindi su **Formato numeri di pagina** e infine su **Comincia da** e immettere **1**.

Per tornare al documento, fare clic su **Chiudi intestazione e piè di pagina** nella scheda **Progettazione** in **Strumenti intestazione e piè di pagina**.

La gestione delle immagini

I word processor, che nel pacchetto office si chiama word permette di inserire e adattare immagini all'interno di un testo con molte opzioni di elaborazione sia della stessa immagine che del testo che la contorna.

Per inserire un'immagine all'interno del testo è necessario portarsi sul menù inserisci, e poi cliccare su immagini.



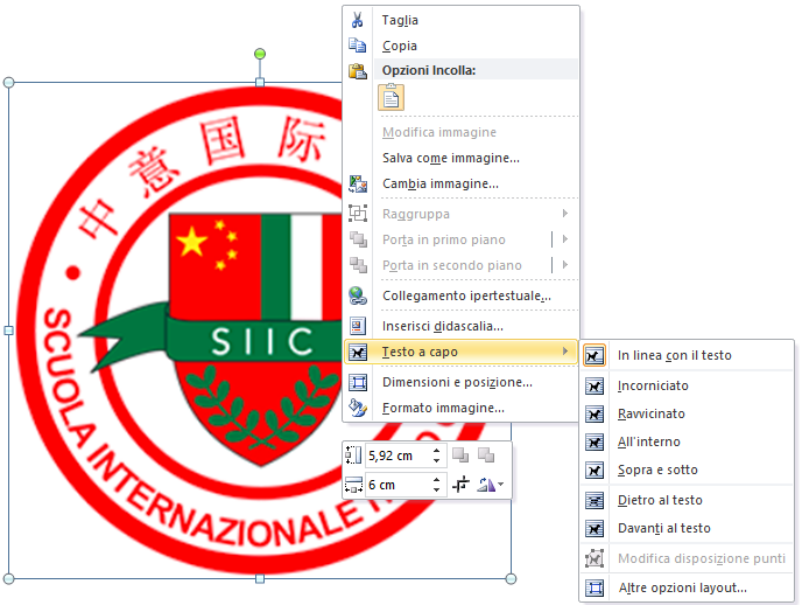
Al comando “inserisci”->”immagine” si apre il browser di navigazione cartelle, solitamente nella posizione immagini di documenti del sistema operativo.

In questa cartella è sufficiente cliccare sull’anteprima dell’immagine che ci interessa caricare.

Supponiamo che si voglia inserire un’immagine del logo della nostra scuola, la SIIC di Padova.

L’inserimento può avvenire tenendo conto di molte opzioni tra cui come l’immagine si integra nel testo già presente nella pagina.

Facendo tasto destro sull’immagine è possibile selezionare le opzioni di integrazione nel testo.



Le opzioni di integrazione dell’immagine sono molte, ad esempio è possibile mettere il testo in sovra impressione dell’immagine, oppure nascondere parte del testo inserendoci sopra l’immagine.

È anche possibile circondare l'immagine con il testo usando la funzione "incorniciato".
Ad esempio, ricavando del testo dal sito della scuola, e copiato questo in un'unica cella di una tabella, usando la funzione **incorniciato** si vedrà che automaticamente viene creato lo spazio attorno all'immagine e riempito con le parole.

Spett.le Presidente Bui,

Cari docenti e studenti,

Buongiorno!

Settembre porta in sé i colori dorati dell'autunno ed il profumo dell'Osmanto. Abbiamo davanti un nuovo anno scolastico e, dopo sei mesi di chiusura, i nostri studenti sono finalmente tornati a scuola ed è arrivato il momento di aprire le porte.

La scuola silenziosa e tranquilla torna ad essere chiassosa ed affollata.

Innanzitutto, voglio, a nome della scuola, dare il benvenuto a tutti i nostri docenti e studenti. Oggi è il primo giorno dell'anno scolastico e abbiamo l'onore di avere tra noi un ospite importante, il Presidente della Provincia di Padova, Fabio Bui. Un uomo molto impegnato che prontamente si è unito a noi in questo primo giorno, esprimendo l'importanza e l'interesse che ha dato al nostro istituto. Voglio a nome di tutta la scuola ringraziarlo di cuore.

Un uomo a cui la cultura cinese interessa particolarmente, appassionato, che è stato più volte in Cina nel corso della sua vita, terra che ha lasciato in lui un piacevole ricordo. Ci impegneremo per lasciare un ricordo altrettanto bello di questa giornata al presidente.

Il 2020 è stato un anno molto particolare. Abbiamo affrontato questa epidemia, siamo stati costretti a chiudere la scuola ma la SIIC non ha sospeso lo studio. Il 2 di marzo la Scuola aveva già attivato le lezioni a distanza. I programmi di tutte le materie sono stati svolti come fossero in presenza. I risultati sono stati fruttuosi e straordinari. Voglio ringraziare i nostri docenti perché, assieme ai nostri studenti, siamo tutti cresciuti insieme e abbiamo fatto ammirevoli progressi. Non solo abbiamo ricevuto le conoscenze, ma abbiamo anche accolto la responsabilità. Adesso il nuovo anno è già iniziato e, con la speranza nel cuore, dobbiamo dedicarci allo studio con più impegno.

Viene lasciato come esercizio il provare gli altri comandi disponibili in questo menù ad esempio "**Ravvicinato**".

Vediamo come delle immagini possano essere ritagliate direttamente dentro a Word senza usare altri strumenti.

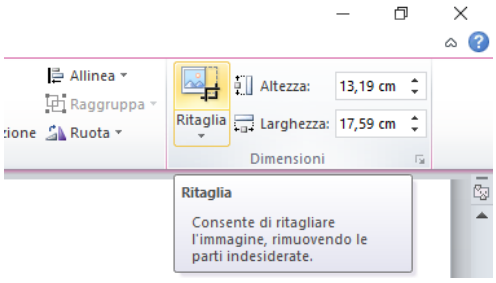
Inseriamo un'immagine che potrebbe risultare troppo grande o con parti che non interessano.

Questa foto scattata dal prof Gottardo Marco durante il viaggio in Giappone in agosto del 2011 rappresenta la porta del tempio “Torì” nell’isola di Miyajima, la citta che si vede di fronte è Hiroshima.

Vogliamo di questa bella foto mostrare il solo particolare del Torì.

Inseriamo nuovamente l’immagine:

Facendo trasto destro sull’immagine compare il menù di impostazione dell’area di ritagliare. È possibile procedere con il mouse oppure agendo sui campi Altezza e larghezza definendo manualmente la misura.



Esistono molte altre opzioni per modificare l’immagine, ad esempio agendo sugli effetti e sulla correzione dei colori.

Vediamo come diventa un’immagine presa da internet, trasformata da colore in bianco e nero o toni di grigio.

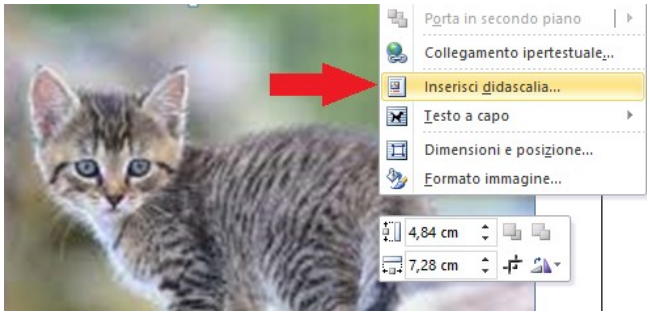
Immagine a colori

Immagine resa in bianco e nero

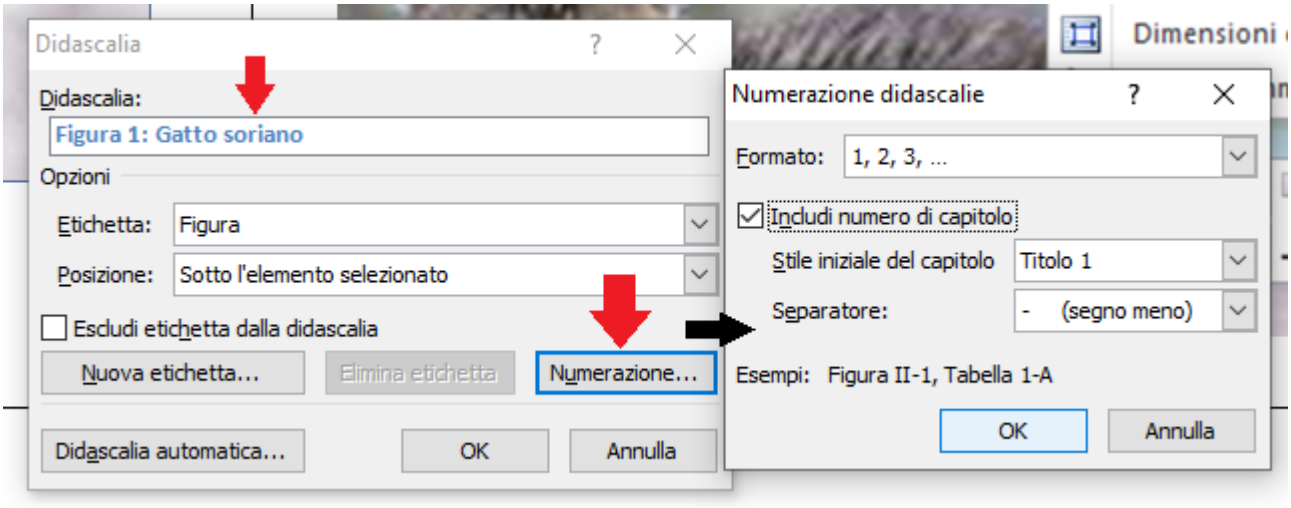
Vediamo i vari menù disponibili per elaborare l'immagine.

Figura 0-1: Gatto soriano

Il comando inserisci didascalia permette di numerare, nominare e rintracciare nel testo una qualsiasi immagine anche creando una lista somigliante a un sommario.

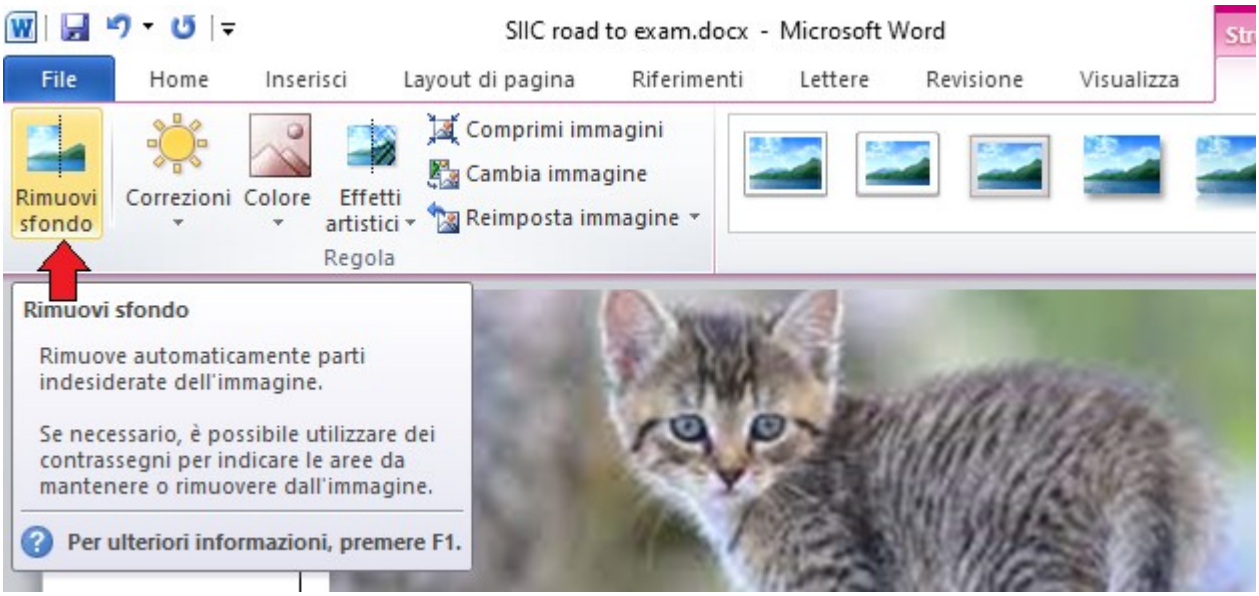


Il menù della didascali è questo:



Esiste la possibilità di togliere lo sfondo da un'immagine prima di inserirla in un'altra immagine oppure nel documento di testo.

Per fare questo si usa il menù "**rimuovi sfondo**":



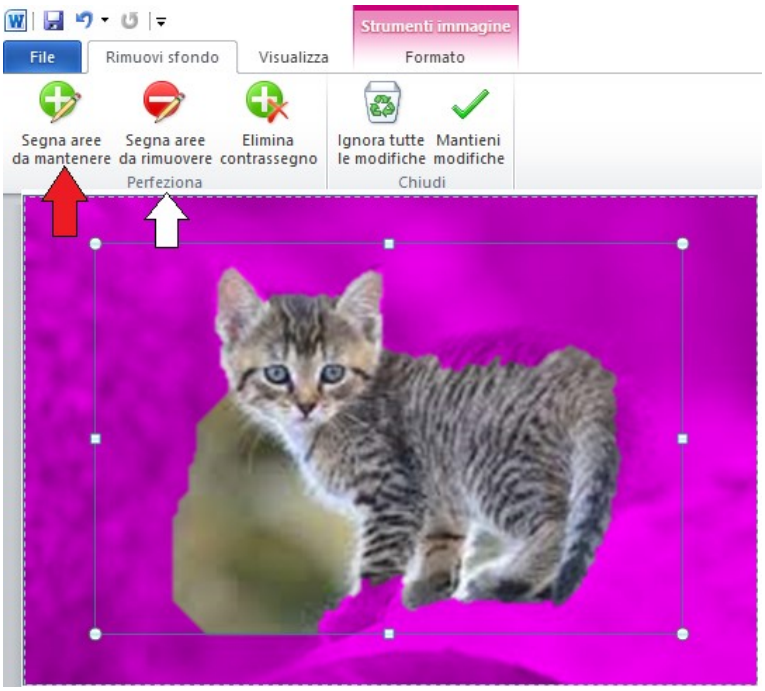
Vediamo l'effetto finale dopo avere rimosso lo sfondo:

Il gatto viene circondato da un alone viola che mostra quale area verrà eliminata dall'immagine, ma non sempre è corretta alla prima prova. Però possiamo correggere

Inseriamo nuovamente l'immagine per mostrare in dettaglio come si opera per mantenere le parti che ci interessano del gatto e rimuovere ulteriori parti dello sfondo che sono inutili.

Facciamo doppio click sull'immagine e vedremo comparire i menu di elaborazione dell'immagine tra cui il rimuovi sfondo e altre utilità come la correzione automatica.

Il rimuovi sfondo permette di aggiungere e togliere parti rispetto a ciò che è stato automaticamente selezionato.



Agiamo prima su “Segna aree da mantenere”, per rifilare con la matita la schiena del gatto, e poi su “Segna aree da rimuovere” per ripescare dall’immagine la zampetta che risulta confusa dallo sfondo a causa della similitudine del colore.

Gatto con aggiunte di parti prima rimosse automaticamente

Gatto con parti riaggiunte

Sfondo senza il gatto

Sfondo con il gatto

La gestione delle tabelle

Nei word processor, tra cui Word per office, è possibile inserire le stesse tabelle che vediamo in Excel benché si perda la funzione di calcolo automatico.

Per mantenere la funzione di calcolo automatico è possibile svolgere le operazioni in Excel e poi importare le tabelle così prodotte.

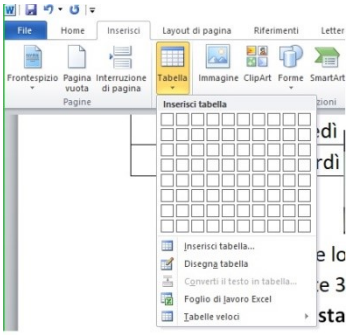
Mario Rossi Febbraio 2021		
giorno settimana	ore	€/giorno
lunedì	8,00	76,00 €
martedì	8,00	76,00 €
mercoledì	8,00	76,00 €
giovedì	8,00	76,00 €
venerdì	8,00	76,00 €
lunedì	8,00	76,00 €
martedì	8,00	76,00 €
mercoledì	8,00	76,00 €
giovedì	8,00	76,00 €
venerdì	8,00	76,00 €
lunedì	8,00	76,00 €
martedì	8,00	76,00 €
mercoledì	8,00	76,00 €
giovedì	8,00	76,00 €
venerdì	8,00	76,00 €
lunedì	8,00	76,00 €
martedì	8,00	76,00 €
mercoledì	8,00	76,00 €
giovedì	8,00	76,00 €
venerdì	8,00	76,00 €

Paga mese lordo	=	1.520,00 €
trattenute 35%		532,00 €
Netto in busta paga		**988,00 €**

Notiamo facilmente che tutte le funzioni di calcolo automatico che era presenti in Excel quando abbiamo creato la busta paga non sono più attive. Per fare una prova basta cambiare uno dei guadagni giornalieri e verificare che non si ripercuote nella paga mensile.

Possiamo inserire tabelle direttamente con la funzione word se non ci serve che questa esegua i calcoli, ad esempio è molto usata per vincolare le immagini in posizioni specifiche anche quando si importi il file dal formato docx al formato pdf.

Possiamo inserire tabella dal comando “Tabella”



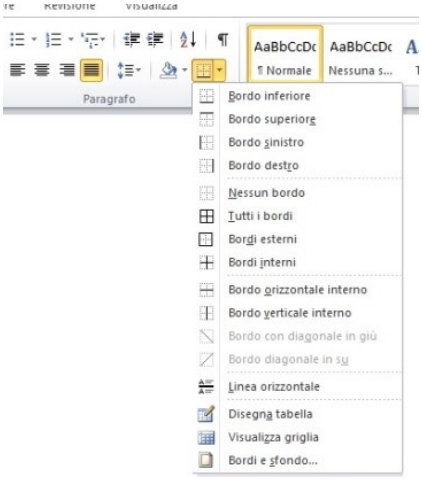
Il menu inserisci tabella permette di definire a priori il numero stimato di righe e di colonne della tabella, ma questo potrà essere variato in qualunque momento.

Selezionata una posizione all’interno della tabella potremmo eseguire il comando inserisci riga sotto o sopra oppure inserisci colonna a destra o a sinistra.

E’ anche possibile aggiungere dei sezionamenti di celle esistenti usando lo strumento disegna tabella che mette a disposizione una specie di matita per tracciare nuovi bordi.

Le celle possono essere visibili o invisibili per ognuno o tutti i suoi lati.

Esiste l’apposito comando



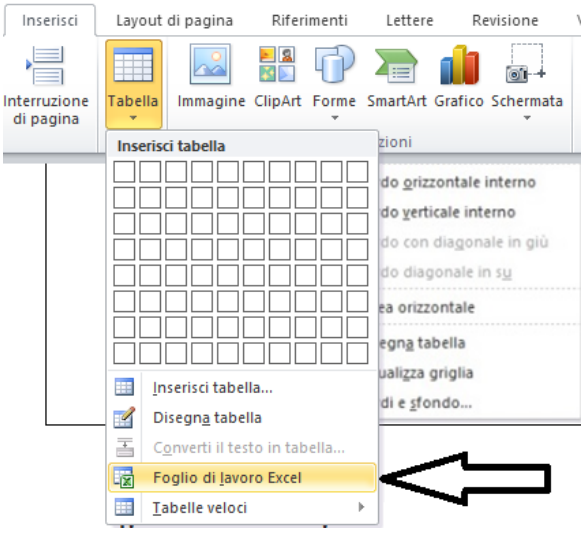
Nei comandi visibili nel menù a tendina si vede che data una cella o un gruppo di celle di una tabella è possibili visualizzare solo uno dei bordi, sopra, sotto, destra, sinistra, oppure tutti i bordi o solo quelli esterni.

È possibile cambiare la forma dei bordi, ingrossandoli o cambiando l’aspetto.

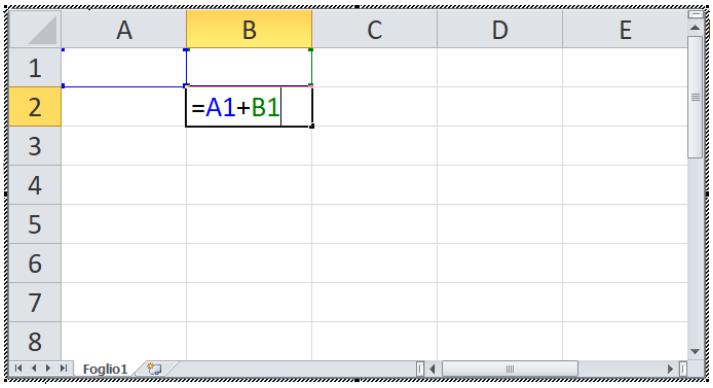
Le celle possono essere unite o disgiunte, esattamente come si fa con i fogli elettronici, del tipo Excel.

È possibile usare la funzione di riempimento con colore e inclinare le scritte contenute nelle celle.

Vediamo come sia possibile inserire una foglio Excel integrandone le funzioni direttamente in Word.

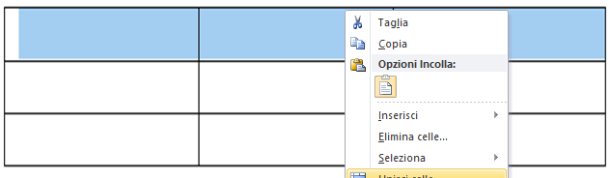


Cliccando su Foglio elettronico Excel compare una tabella.



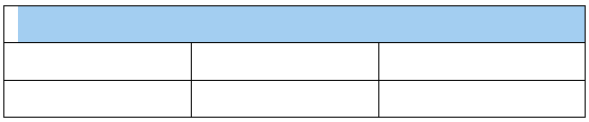
Cose più semplici riguardano l'unione di celle e la gestione delle grafiche e testi in esse connesse.

Selezioniamo usando il mouse più celle ottenendo la situazione in figura:



Agiamo poi nel menù unisci celle.

La situazione dopo avere unito le celle è:

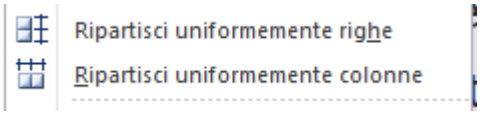


Ora è possibile scrivere titoli o intestazioni per gruppi di colonne.

Busta paga mese si marzo 2021

Le proprietà delle tabelle, soprattutto la posizione o la larghezza può avvenire per impostazione manuale oppure per impostazione automatica.

La necessità di reimpostare la dimensione di una tabella può derivare da un cambio di formato della pagina, ad esempio per riduzione dallo standard A4 a uno più piccolo, ad esempio il formato lettera o libretto.

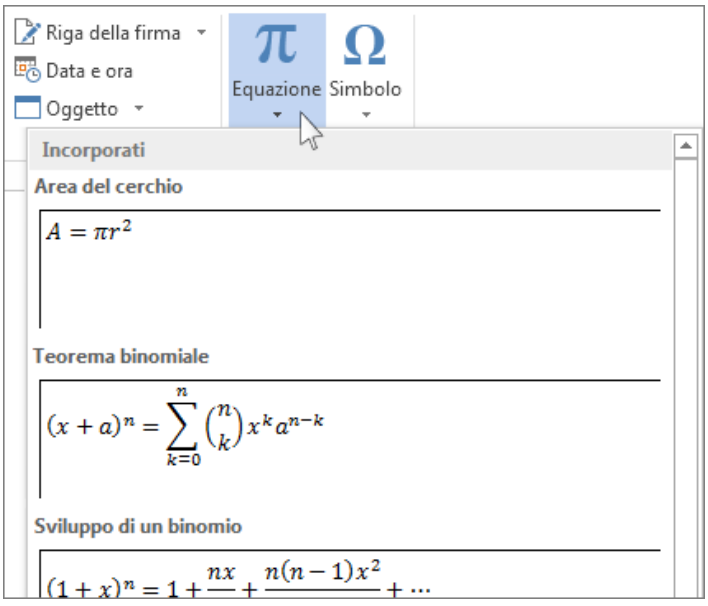


Eseguendo entrambi i comandi Ripartisci uniformemente righe e Ripartisci uniformemente colonne la tabella diventa come sotto.

L'inserimento di disegni, simboli e formule

Il Word di Office come l'equivalente di Open Office permette di inserire formule matematiche o simboli correlati alle materie scientifiche o unicode allo Scopo di esprimere al meglio tutto quanto concerne le materie scientifiche.

È possibile inserire qualunque formula a partire da dei modelli "template" come visibile nell'immagine.



Sono presenti tutti i simboli e gli operatori matematici allo scopo di descrivere polinomi, sommatorie, elevamenti a potenza, pedici, binomiali, fattoriali, ecc.

Le frazioni ad esempio necessitano che polinomi possano essere scritti a numeratore e a denominare.

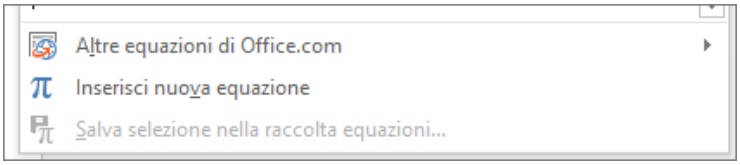
All'inserimento di una formula compaiono dei quadratini vuoti posizionati dove si devono mettere i componenti della formula quindi i numeri, gli operatori matematici, i simboli di supporto.

La distribuzione dei quadratini è prelevabile da librerie che comprendono tutti i possibili

casi.

Per una nuova equazione senza usare il mouse, premere “ALT+= “ sulla tastiera.

N’altra maniera per aggiungere formule è: Inserisci > Equazione, poi selezionare Inserisci nuova equazione nella parte inferiore della raccolta di equazioni. In questo modo viene inserito un segnaposto in cui è possibile digitare l'equazione.



FOGLIO ELETTRONICO

- Creare, salvare, aprire, modificare e chiudere una cartella di lavoro.

Eseguire semplici calcoli e espressioni con gli operatori matematici

Il foglio elettronico, anche se proposto da pacchetti software diversi, si compone di celle identificabili tramite l’intersezione della riga (un numero progressivo) con la colonna (una lettera progressiva).

Ad esempio A1 è la prima cella in alto a sinistra del foglio elettronico.

Se faccio tasto destro su questa cella, accedo al menu “formato celle” e la imposto come “numero”, così che possa farci dei calcoli.

La cella sottostante è la cella A2, che potrà essere impostata nella stessa maniera o trascinando con il mouse le caratteristiche di quella sopra oppure ripetendo tasto destro, proprietà della cella, “numero”.

Nella cella sottostante, A3, imposto la caratteristica numero.

Sempre nella cella A3, scrivo “=” seguito da una formula o un calcolo numerico, ad esempio posso scrivere:

=A1+A2

Quando eseguo “invio” o enter, la cella diventa somma delle due precedenti.

Se metto il numero 3, nella cella A1, e il numero 2 nella cella A2 allora nella cella A3 compare 5.

Se cambio i valori nelle cella A1 o A2 il risultato in A3 si aggiorna automaticamente.

Con questa tecnica posso eseguire qualunque calcolo o formula in qualunque cella, usando i formati numerici standard di cui il più comune è il numero Reale, ovvero quello con la virgola.

- Assegnare diversi formati numerici e dimensionare righe e colonne.

- Eseguire calcoli con le funzioni .

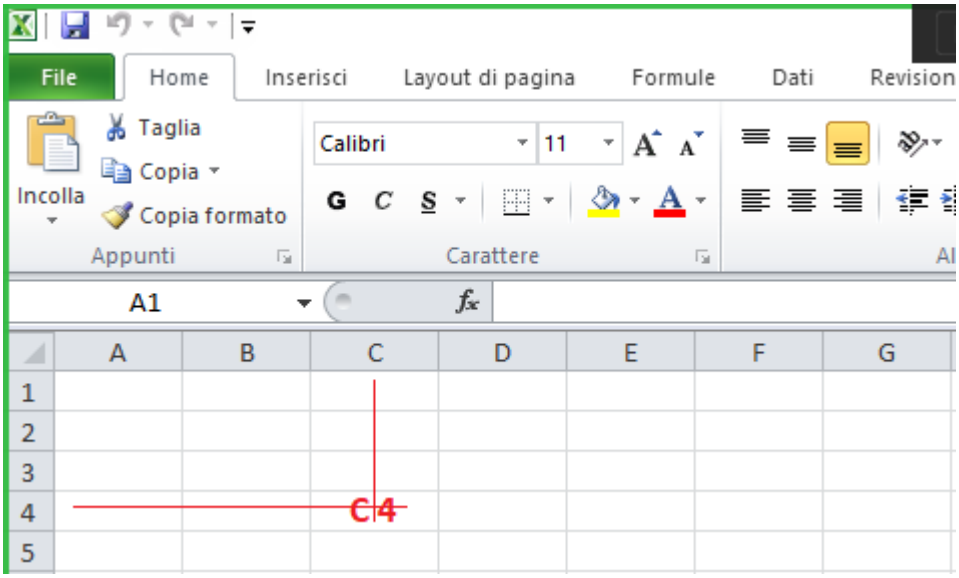
Il foglio elettronico

Esistono molti pacchetti software utili alla gestione di problematiche di ufficio, tra cui i fogli di calcolo.
I più utilizzati sono Excel per quanto riguarda Microsoft all'interno del pacchetto office, ma anche Calc all'interno del pacchetto Openoffice.
Il foglio elettronica ha l'aspetto di una matrice, ovvero celle identificabili con righe e colonne, in cui si posso invertire i vari tipi di dato riconosciuti dal calcolatore.

Caratteristiche e funzionalità del foglio elettronico

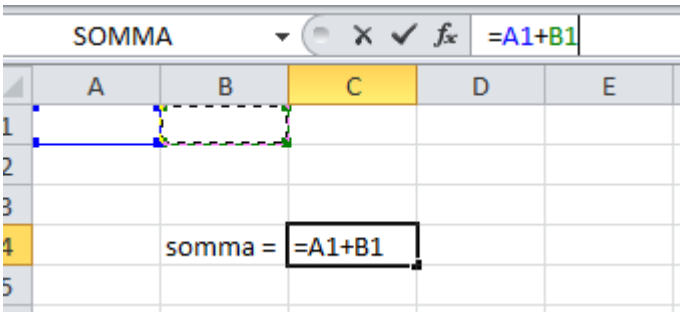
Il foglio elettronico, o foglio di calcolo, permette l'impostazione di formule e calcoli sui dati che in esso vengono inseriti.

Il foglio è diviso in cartelle di lavoro che contengono pagine con coordinate di celle.

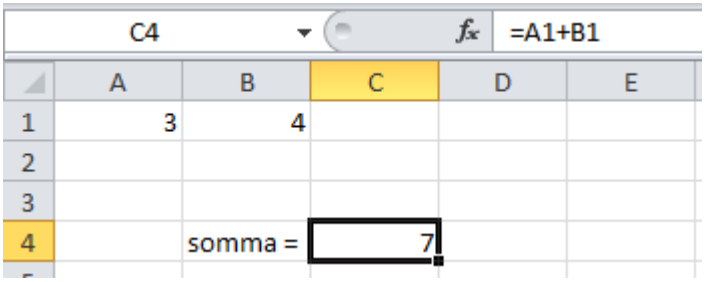


Ad ogni cella, identificata, è possibile associare un valore, una quantità economica, oppure una formula.

Il foglio elettronico, nel campo fx aspetta il simbolo = per impostare una formula. Questa formula ha come dati di ingresso altre celle, ad esempio la formula fx=A1+A2 esegue la somma dei dati in quelle coordinate.



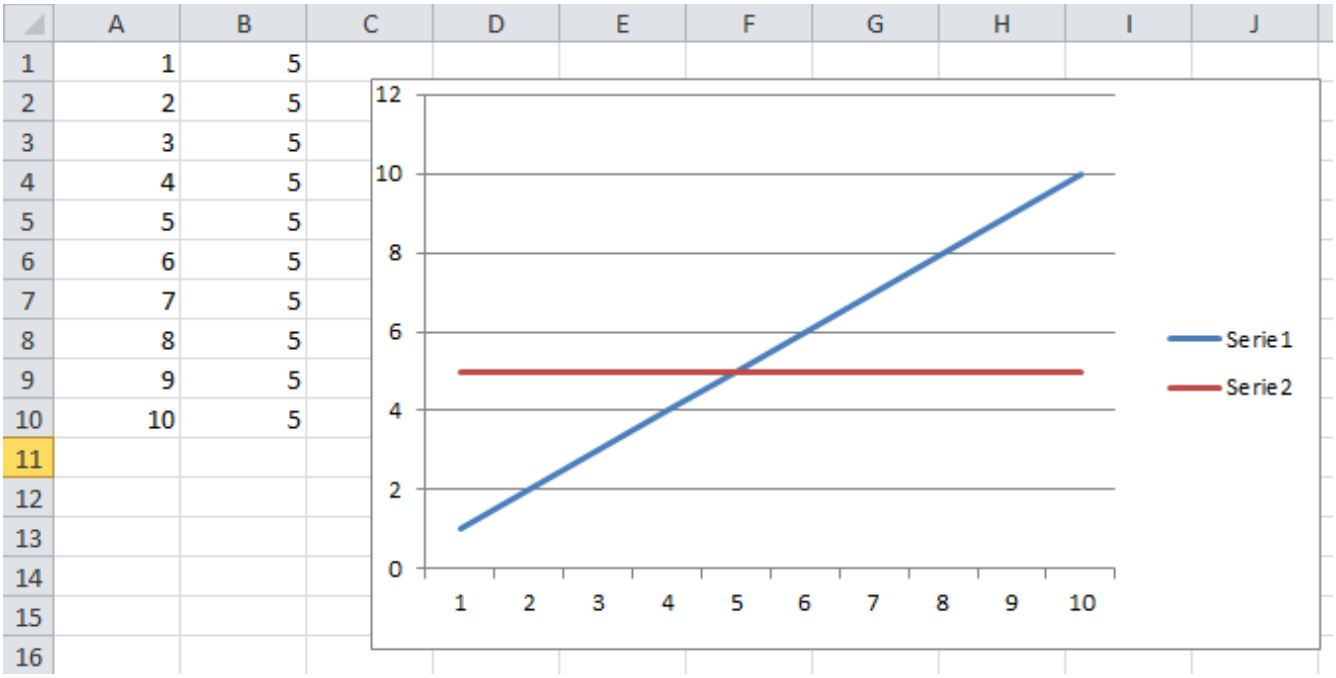
Con il tasto "invio" dalla tastiera la formula viene accetta e diventa subito operativa.



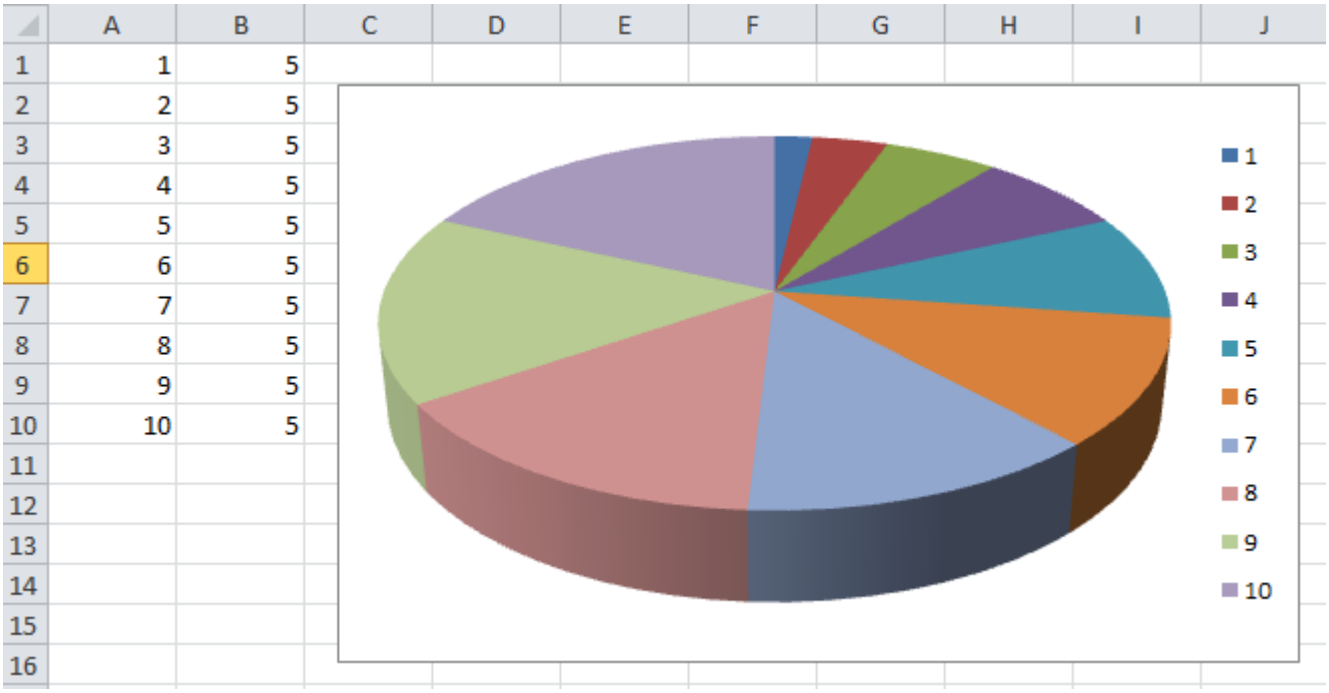
Le formule possono essere molto complesse e avere al loro interno altre formule.

Una cosa importante dei fogli elettronici è che possono manipolare i dati in input per ottenere in maniera diretta anche dei grafici, di vario tipo, ad esempio diagrammi a torta, oppure istogrammi (barre) oppure grafici di tipo linea continua su x,y.

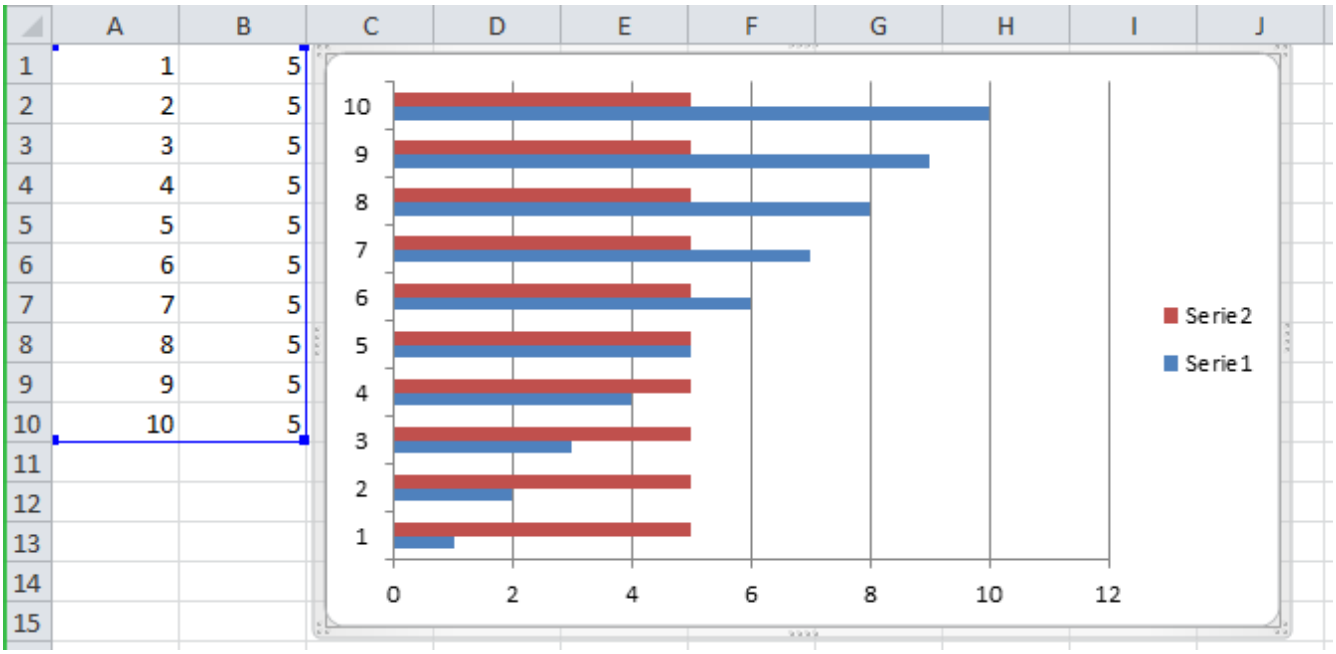
Vediamo un esempio di grafico su due set di dati, la colonna A e la colonna B.



Vediamo altre visualizzazioni degli stessi dati, ad esempio il diagramma a torta



Vediamo come vengono interpretati i dati con un diagramma a barre.

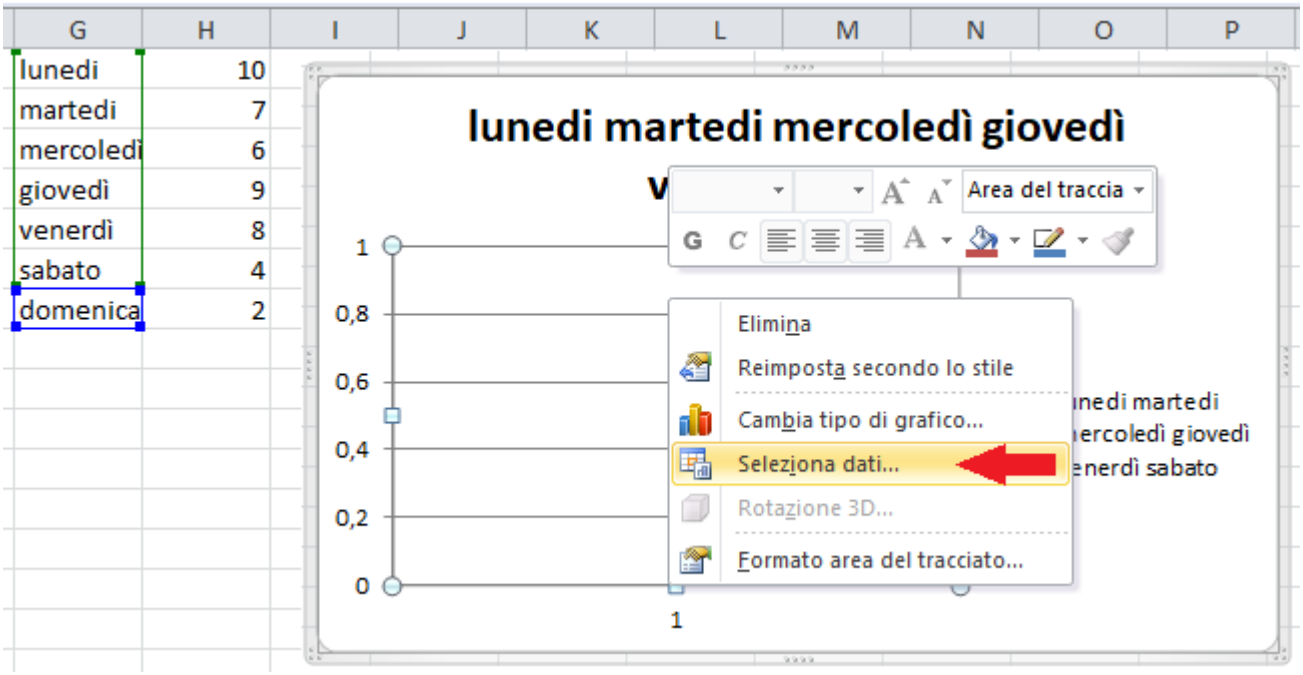


Vediamo un esempio di un grafico che rappresenta la spesa settimanale. In ascissa (asse orizzontale) mettiamo i giorni della settimana, da lunedì a domenica, mentre in ordinata (asse verticale) mettiamo i soldi spesi in euro.

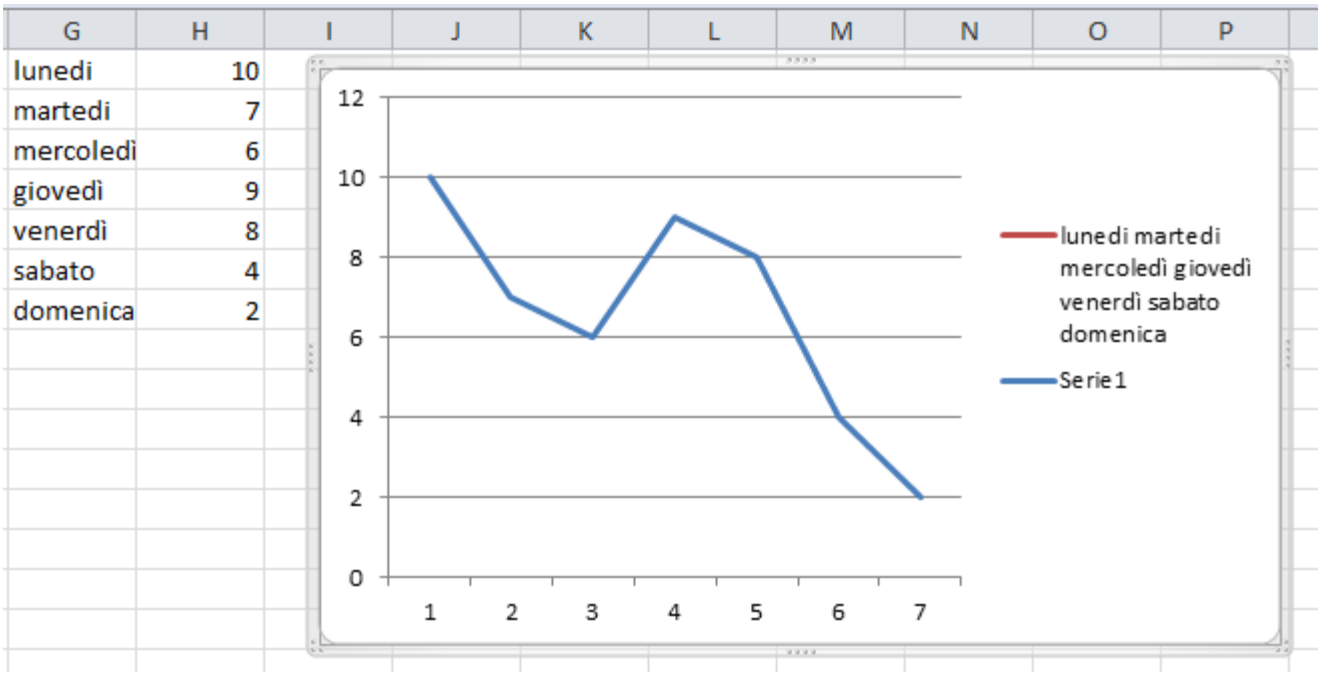
Bisogna indicare al foglio elettronico quali sono le serie che noi vogliamo mettere sul grafico, in questo caso la serie 1 è la colonna G, mentre la serie 2 è la colonna H.

Con il mouse indicare la selezione dei dati per colona.

Per attivare la selezione della serie numerica bisogna prima puntare al grafico vuota e attivare il menu di selezione dei dati.

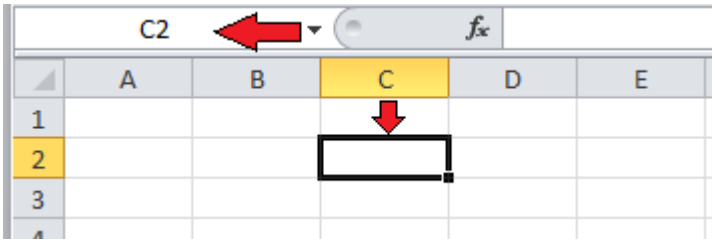


Provare a selezionare in maniera diversa le colonne fino ad ottenere il grafico che stiamo aspettando, ad esempio quello sotto.



Definizione di cella, zona, etichetta, valore e formula

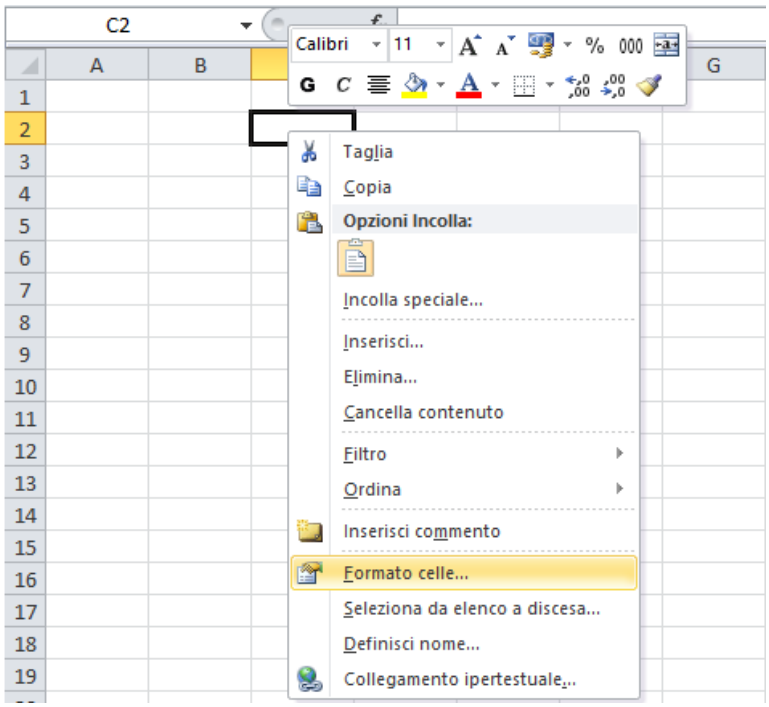
Le celle di Excel o Calc (per gli utenti OpenOffice) sono aree mappate del foglio elettronico puntabili grazie agli indici di riga (numeri) e di colonna (lettere).



Nell'immagine la cella puntata è alle coordinate del foglio elettronico C2, come indicato in alto a sinistra dalla freccia.

Quando una cella è attiva viene contornata da un rettangolo nero con linea grossa.

Sulle righe e colonne si accendono i numeri e le lettere corrispondenti.



Ogni cella rappresenta una variabile che si alloca nel foglio nel momento in cui viene utilizzata.

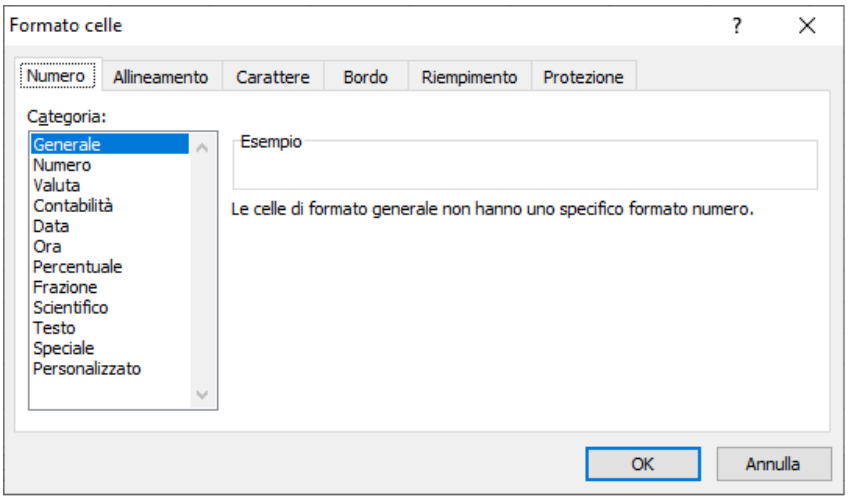
Come ogni variabile deve essere descritta con un nome, e un tipo.

Il nome della variabile coincide con le sue coordinate convertite in testo, quindi la cella mostrata in figura si chiama C2.

Il tipo della variabile associata alla cella viene definito dal programmatore utilizzando il comando "formato cella".

I formati più utilizzati sono i numerici, i testuali, e quelli associati alla valuta economica (soldi).

Ma esistono anche tanti altri tipi di dato, detti enumerati, ovvero che di numerano in maniera crescente o decrescente spesso con la possibilità di usare il riempimento automatico (una delle funzioni fondamentali del foglio elettronico).



Il formato valuta ci permette di definire le valute monetarie attive nel mondo, con i rispettivi simboli, ad esempio il dollaro americano $ oppure l'euro europeo €, e qualsiasi altro.

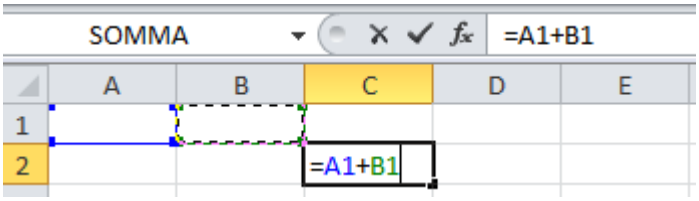
Il valore contenuto nella cella può essere predefinito nella sua precisione, ovvero il numero di cifre decimali sia prima che dopo lo zero che mostra la posizione della virgola.

Ad esempio 10,3€ ha una sola cifra decimale, mentre 10,35€ ne ha due.

Le celle possono oltre ad avere come caratteristica una zona anche un'etichetta identificativa che ne fissa un nome anche diverso da ciò che è la funzione associata.

Per abilitare una formula, bisogna scrivere sulla cella il simbolo =

Questo comparirà automaticamente sulla riga bianca della formula.

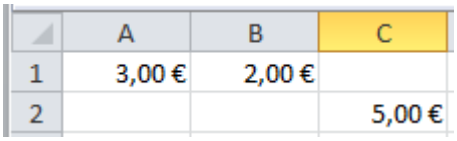


L'immagine mostra come predisporre una formula che esegue la somma tra la cella A1 e la cella B1 con risultato nella cella C2.

Se non ci sono formati numerici impostati diversi allora il tipo è numero intero senza virgola con segno.

Una volta impostata una funzione, ad esempi la somma, ad una cella, questa è sempre attiva, quindi inserendo i numeri nelle celle in cui ci sono gli addendi il valore della somma non attende un consenso ma viene subito eseguita.

La somma funziona con tutti formati compatibili, ad esempio, la valuta.



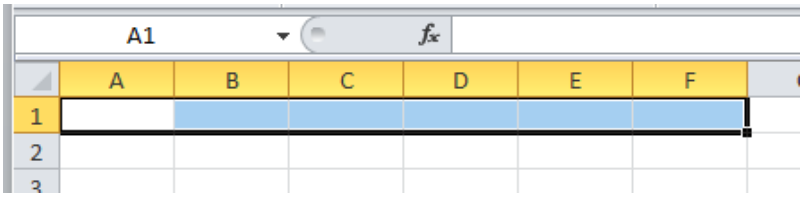
Cambiare il formato di A1 in valuta con due cifre decimali, poi lo stesso sia per la cella B1 che la cella C2.

Inserire dei valori di test, e notare che l'unità di misura euro appare automaticamente anche nella somma finale.

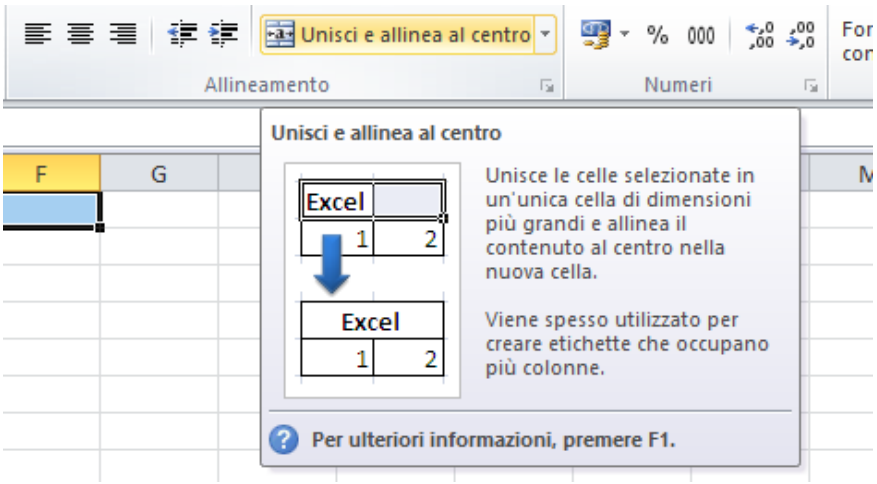
I formati testuali, sono molto agevoli e sempre attivi a meno che non sia stato assegnato un diverso formato alla cella, quindi se scrivo una stringa di caratteri questo vengono subito accettati.

Se dobbiamo scrivere delle etichette o dei titoli delle volte è necessario creare più spazio, ad esempio con la funzione unisci e centra.

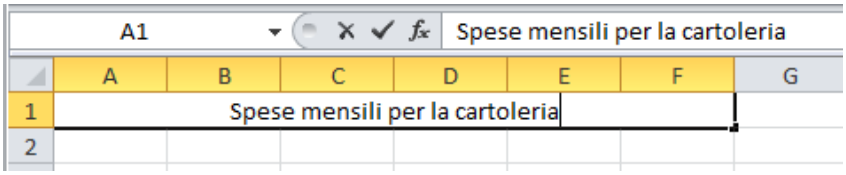
Selezioniamo un gruppo di celle, ad esempio da A a F, come nell'immagine:



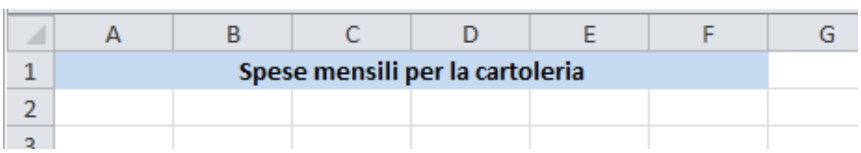
Abilitiamo la funzione unisci e centra.



Eseguito questo comando troveremo una sola cella in grado di contenere lunghe scritte, ad esempio il titolo del foglio elettronico che stiamo andando a creare.



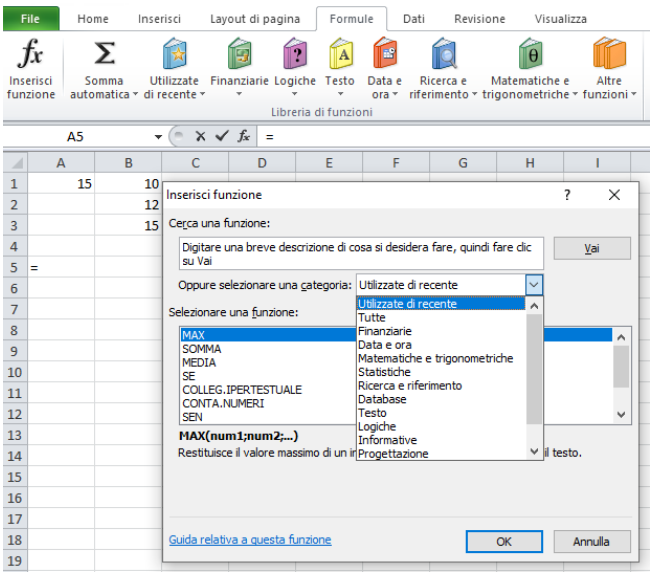
Potremmo poi agire sulle caratteristiche dello stile e del font per migliorare l'impressione visiva, ad esempio mettiamo il carattere grassetto con sfondo pastello azzurro.

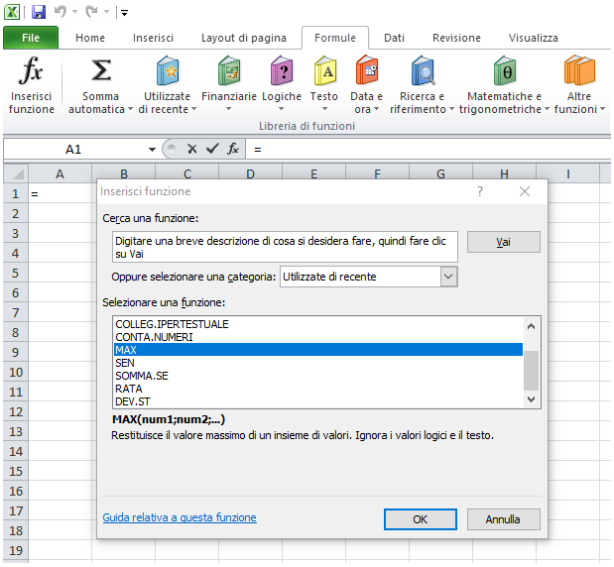


Struttura di una formula e i simboli degli operatori matematici

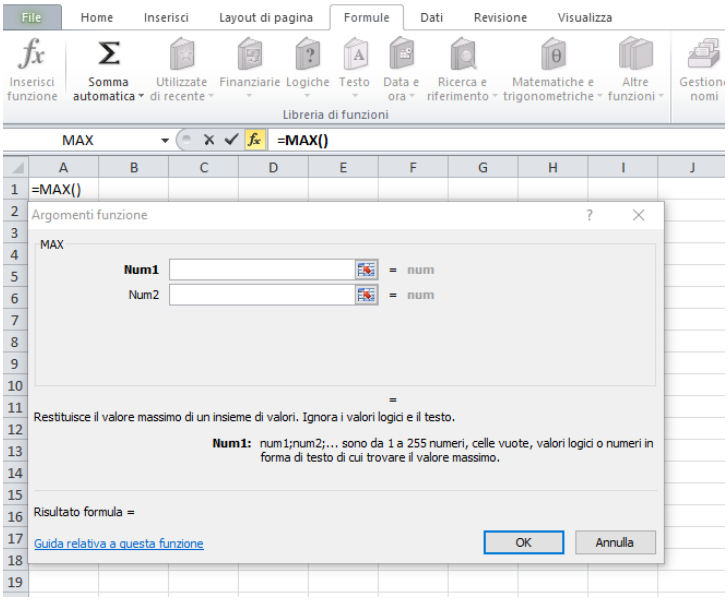
Inserimento formule:

In un foglio di calcolo si possono inserire molte formule già preformattate semplicemente accedendo al menu inserisci:

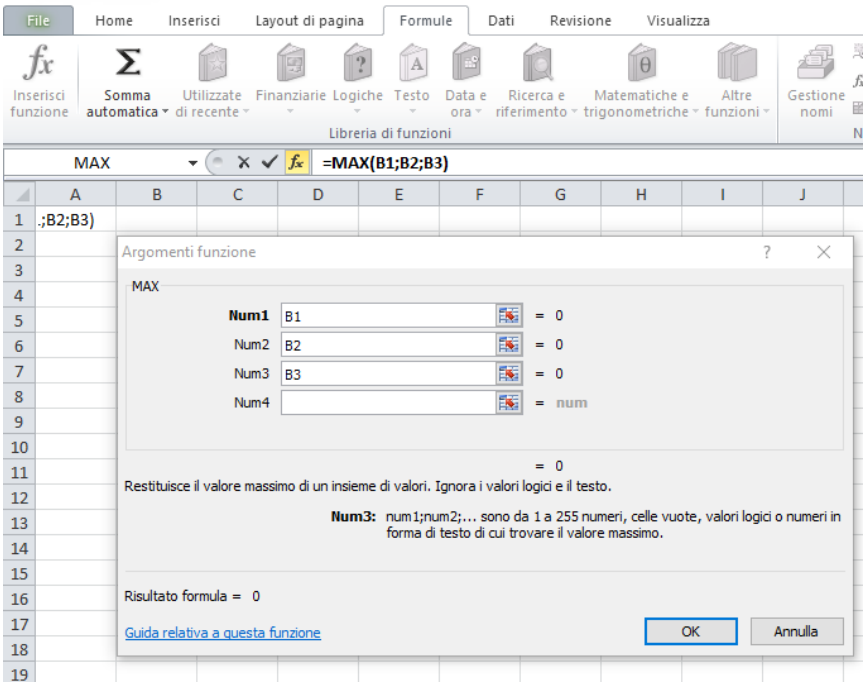




Cliccando su “ok” si apre una nuova finestra per l’inserimento contestuale dei dati:

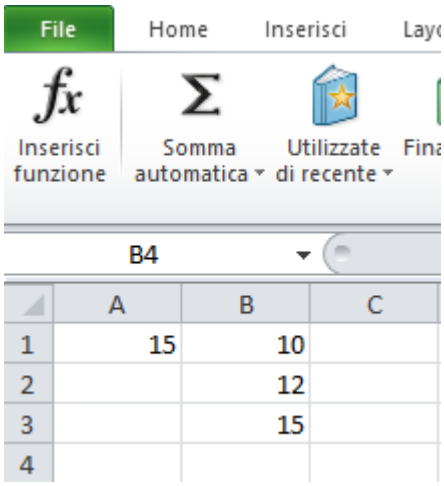


Si procede quindi con la configurazione della formula:



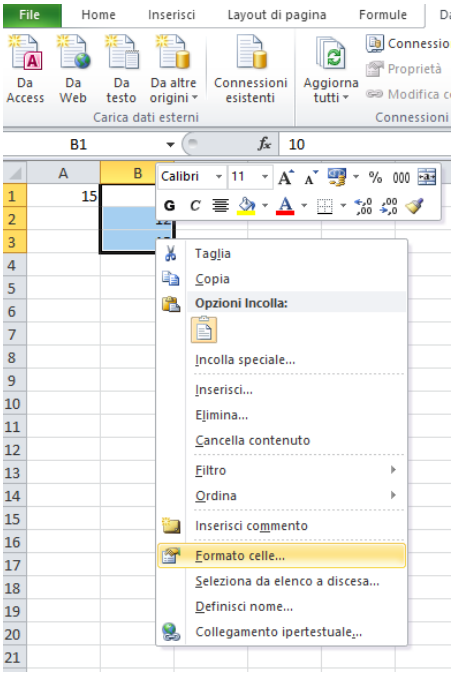
Non appena saranno inseriti i dati sulle caselle corrispondenti verrà applicata la formula ed il risultato apparirà nella casella nella quale è stata inserita la formula, in questo caso in “A1”.

Risultato:

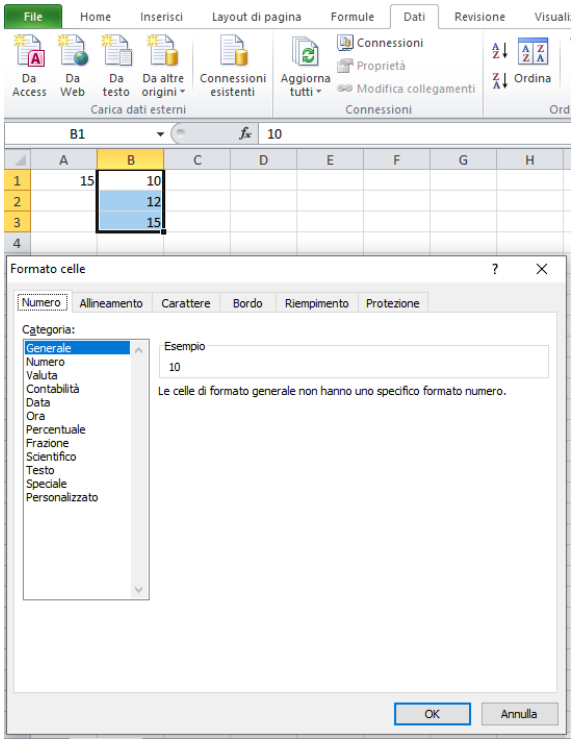


Formattazione:

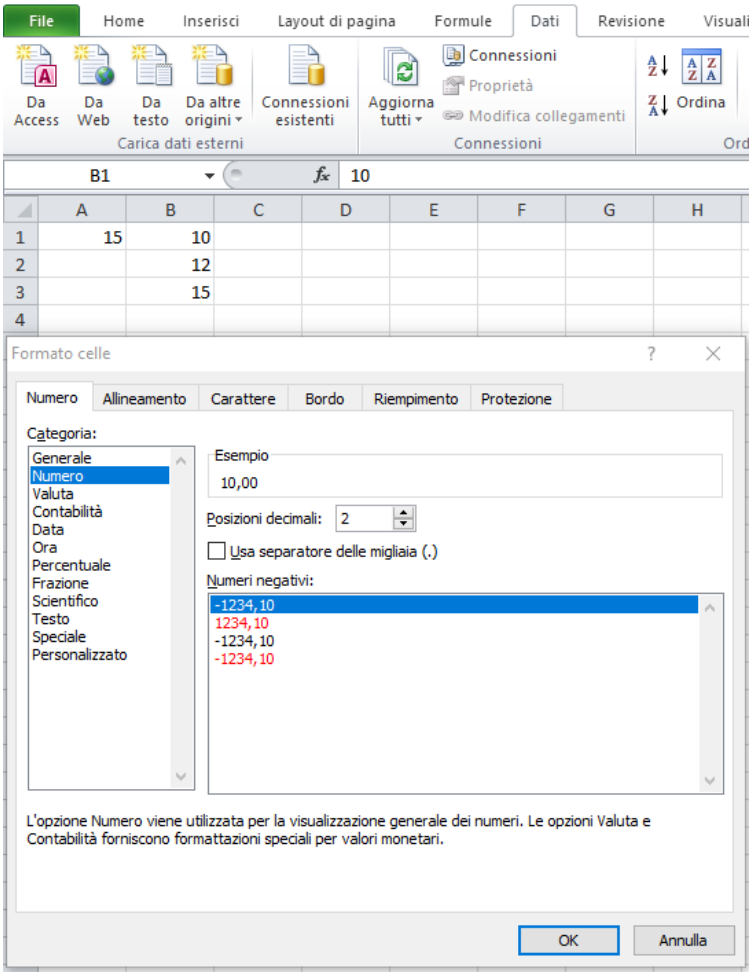
Il dato numerico contenuto nelle celle può essere formattato in relazione alle esigenze selezionando le celle contenenti i dati omogenei e alla pressione del tasto destro dei mouse selezionare il menu “Formato Celle”:



Selezionare di seguito il sottomenu “Numero”:



Applicare la formattazione desiderata scegliendo tra quelle proposte:



Operatori per calcoli algebrici

Per utilizzare uno di questi operatori matematici basta puntare ad una cella e scrivere il contenuto della terza colonna della tabella mostrata. Quando si da invio compare il risultato nella stessa tabella.

Operatore aritmetico	Significato	Esempio
+ (segno di addizione)	Addizione	=3+3
– (segno meno)	Sottrazione Negazione	= 3-3 =-3
Un asterisco (*)	Moltiplicazione	=3*3
/ (segno di divisione)	Divisione	=3/3
% (segno di percentuale)	Percentuale	30%
^ (accento circonflesso)	Elevamento a potenza	=3^3

Operatori per confronti, detti comparatori

Per utilizzare gli operatori di confronto digitare = nella cella in cui vogliamo che compaia il risultato della comparazione tra il contenuto delle celle indicate.

Il risultato è la scritta VERO oppure FALSO, o in inglese TRUE o FALSE.

Operatore di confronto	Significato	Esempio
= (segno di uguale)	Uguale a	=A1=B1
> (segno di maggiore)	Maggiore di	=A1>B1
< (segno di minore)	Minore di	=A1<B1
>= (segno di maggiore o uguale a)	Maggiore di o uguale a	=A1>=B1
<= (segno di minore o uguale a)	Minore o uguale a	=A1<=B1
<> (segno di diverso da)	Diverso da	=A1<>B1

Operatore per concatenare un testo

Si può usare la e commerciale (**&**) per unire, che in informatica si dice concatenare, una o più stringhe di testo creando una sola stringa.

Operatore di testo	Significato	Esempio
& (e commerciale)	Concatena due valori generando un singolo valore di testo	="North"&"wind" restituiscono "Northwind". Dove a1 contiene "cognome" e B1 contiene "First Name", = a1& "," &B1 restituisce "cognome, First Name".

Operatori di riferimento

Questi sono il due punti, il punto e virgole e lo spazio, da usarsi quando si deve considerare un intervallo di celle adiacenti, come avviene ad esempio nella somma automatica. A1:A10, fa la somma delle celle tra A1 e A10. Se le celle non sono contigue , si posso comunque sommare intervalli separati =SOMMA(B5:B15;D5:D15)

Elementi che compongono una formula
Una formula può contenere

1. Funzioni
2. Riferimenti
3. Operatori
4. Costanti

Una **funzione** è una formula predefinita che accetta uno o più valori, esegue un'operazione e restituisce uno o più valori. Utilizzare le funzioni per semplificare e abbreviare le formule in un foglio di lavoro.

Sono utili per eseguire calcoli lunghi e complessi.

I **riferimenti** sono indirizzi di celle, quindi coppie di coordinate che identificano la posizione di una cella nel foglio di lavoro. Ad esempio, il riferimento alla cella situata all'intersezione della colonna C con la riga 4 è C4.

Gli **operatori** sono segni o simboli che specificano il tipo di calcolo da eseguire in un'espressione. Esistono operatori matematici, di confronto, logici e di riferimento.

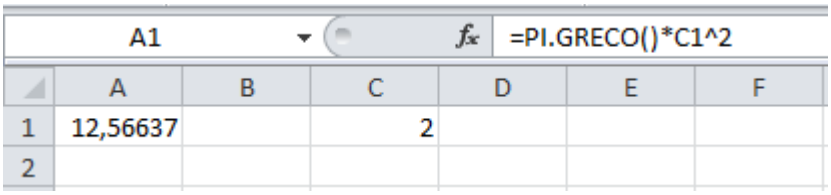
Le **costanti** sono un valore non calcolato che rimane invariato. Ad esempio, il numero 311 e il testo ''Guadagni mensili'' sono costanti. Un'espressione o un valore risultante da un'espressione non sono costanti.

Per impostare una formula bisogna puntare a una cella, in cui vogliamo visualizzare il risultato, e scriviamo "=", questo comparirà nella barra della formula in alto indicata con f_x

Per **calcolare l'area del cerchio** scriveremo =P1() * "cella del raggio" ^ 2 dove il simbolo "^" significa elevato a, in questo caso al quadrato.

Quando diamo l'invio comparirà il risultato nella cella in cui abbiamo impostato la formula.

Nella prossima immagine vediamo un esempio.



Elenco delle funzioni accettate da Excel

Nome funzione	Tipo e descrizione
AMMORT	**Finanziarie:** restituisce l'ammortamento di un bene per un periodo specificato utilizzando il metodo di ammortamento a doppie quote decrescenti o altri metodi specificati
ADESSO	**Data e ora:** restituisce il numero seriale della data e dell'ora correnti
AGGREGA	**Matematiche e trigonometriche:** restituisce un'aggregazione in un elenco o un database
AMBIENTE.INFO	**Informative:** restituisce informazioni sull'ambiente operativo corrente
AMMORT.ANNUO	**Finanziarie:** restituisce l'ammortamento a somma degli anni di un bene per un periodo specificato
AMMORT.COST	**Finanziarie:** restituisce l'ammortamento a quote costanti di un bene per un singolo periodo
AMMORT.DEGR	**Finanziarie:** restituisce l'ammortamento per ogni periodo contabile utilizzando un coefficiente di ammortamento
AMMORT.FISSO	**Finanziarie:** restituisce l'ammortamento di un bene per un periodo specificato utilizzando il metodo di ammortamento a quote fisse decrescenti
AMMORT.PER	**Finanziarie:** restituisce l'ammortamento per ogni periodo contabile
AMMORT.VAR	**Finanziarie:** restituisce l'ammortamento di un bene per un periodo specificato o parziale utilizzando il metodo a doppie quote proporzionali ai valori residui
ANNO	**Data e ora:** converte un numero seriale in un anno
ANNULLA.SPAZI	**Testo:** elimina gli spazi dal testo
ARCCOS	**Matematiche e trigonometriche:** restituisce l'arcocoseno di un numero
ARCCOSH	**Matematiche e trigonometriche:** restituisce l'inversa del coseno iperbolico di un numero
ARCSEN	**Matematiche e trigonometriche:** restituisce l'arcoseno di un numero
ARCSENH	**Matematiche e trigonometriche:** restituisce l'inversa del seno iperbolico di un numero
ARCTAN	**Matematiche e trigonometriche:** restituisce l'arcotangente di un numero
ARCTAN.2	**Matematiche e trigonometriche:** restituisce l'arcotangente delle coordinate x e y
ARCTANH	**Matematiche e trigonometriche:** restituisce l'inversa della tangente iperbolica di un numero
AREE	**Ricerca e riferimento:** restituisce il numero di aree in un riferimento
Funzione ARROTONDA	**Matematiche e trigonometriche:** arrotonda un numero a un numero specificato di cifre
ARROTONDA.DIFETTO	**Matematiche e trigonometriche:** arrotonda il valore assoluto di un numero per difetto
ARROTONDA.DIFETTO.PRECISA	**Matematiche e trigonometriche:** arrotonda un numero per eccesso all'intero più vicino o al multiplo più vicino a peso. Indipendentemente dal segno di num, il numero viene arrotondato per eccesso

ARROTONDA.ECCESSO	**Matematiche e trigonometriche:** arrotonda un numero per eccesso all'intero più vicino o al multiplo più vicino a peso
ARROTONDA.ECCESSO.PRECISA	**Matematiche e trigonometriche:** arrotonda un numero per eccesso all'intero più vicino o al multiplo più vicino a peso. Indipendentemente dal segno di num, il numero viene arrotondato per eccesso
ARROTONDA.MULTIPLO	**Matematiche e trigonometriche:** restituisce un numero arrotondato al multiplo desiderato
ARROTONDA.PER.DIF	**Matematiche e trigonometriche:** arrotonda il valore assoluto di un numero per difetto
ARROTONDA.PER.ECC	**Matematiche e trigonometriche:** Arrotonda il valore assoluto di un numero per eccesso
ASC	**Testo:** modifica le lettere inglesi o il katakana a doppio byte all'interno di una stringa di caratteri in caratteri a singolo byte
ASIMMETRIA	**Statistiche:** restituisce il grado di asimmetria di una distribuzione
ASS	**Matematiche e trigonometriche:** restituisce il valore assoluto di un numero
BAHTTESTO	**Testo:** converte un numero in testo, utilizzando il formato valuta ß (baht)
BESSEL.I	**Progettazione:** restituisce la funzione di Bessel modificata In(x)
BESSEL.J	**Progettazione:** restituisce la funzione di Bessel Jn(x)
BESSEL.K	**Progettazione:** restituisce la funzione di Bessel modificata Kn(x)
BESSEL.Y	**Progettazione:** restituisce la funzione di Bessel Yn(x)
BINARIO.DECIMALE	**Progettazione:** converte un numero binario in decimale
BINARIO.HEX	**Progettazione:** converte un numero binario in esadecimale
BINARIO.OCT	**Progettazione:** converte un numero binario in ottale
BOT.EQUIV	**Finanziarie:** restituisce il rendimento equivalente a un'obbligazione per un buono del tesoro
BOT.PREZZO	**Finanziarie:** restituisce il prezzo di un buono del tesoro dal valore nominale di € 100
BOT.REND	**Finanziarie:** restituisce il rendimento di un buono del tesoro
CAP.CUM	**Finanziarie:** restituisce il capitale cumulativo pagato per estinguere un debito fra due periodi
CASUALE	**Matematiche e trigonometriche:** restituisce un numero casuale compreso tra 0 e 1
CASUALE.TRA	**Matematiche e trigonometriche:** restituisce un numero casuale compreso tra i numeri specificati
CELLA	**Informative:** restituisce le informazioni sulla formattazione, la posizione o il contenuto di una cella
CERCA	**Ricerca e riferimento:** ricerca i valori in un vettore o in una matrice
CERCA.ORIZZ	**Ricerca e riferimento:** effettua una ricerca nella riga superiore di una matrice e restituisce il valore della cella specificata
CERCA.VERT	**Ricerca e riferimento:** effettua una ricerca nella prima colonna di una matrice e si sposta

	attraverso la riga per restituire il valore di una cella
CODICE	**Testo:** restituisce un codice numerico per il primo carattere di una stringa di testo
CODICE.CARATT	**Testo:** restituisce il carattere specificato dal numero di codice
COLLEG.IPERTESTUALE	**Ricerca e riferimento:** crea un collegamento a un documento archiviato in un server di rete, una rete Intranet o Internet
COLONNE	**Ricerca e riferimento:** restituisce il numero di colonne in un riferimento
COMBINAZIONE	**Matematiche e trigonometriche:** restituisce il numero delle combinazioni per un numero assegnato di oggetti, indipendentemente dal loro ordine
COMP.ARGOMENTO	**Progettazione:** restituisce l'argomento theta, un angolo espresso in radianti
COMP.CONIUGATO	**Progettazione:** restituisce il complesso coniugato di un numero complesso
COMP.COS	**Progettazione:** restituisce il coseno di un numero complesso
COMP.DIFF	**Progettazione:** restituisce la differenza fra due numeri complessi
COMP.DIV	**Progettazione:** restituisce il quoziente di due numeri complessi
COMP.EXP	**Progettazione:** restituisce il valore esponenziale di un numero complesso
COMP.IMMAGINARIO	**Progettazione:** restituisce il coefficiente immaginario di un numero complesso
COMP.LN	**Progettazione:** restituisce il logaritmo naturale di un numero complesso
COMP.LOG10	**Progettazione:** restituisce il logaritmo in base 10 di un numero complesso
COMP.LOG2	**Progettazione:** restituisce il logaritmo in base 2 di un numero complesso
COMP.MODULO	**Progettazione:** restituisce il valore assoluto (modulo) di un numero complesso
COMP.PARTE.REALE	**Progettazione:** restituisce il coefficiente reale di un numero complesso
COMP.POTENZA	**Progettazione:** restituisce il numero complesso elevato a una potenza intera
COMP.PRODOTTO	**Progettazione:** restituisce il prodotto di numeri complessi
COMP.RADQ	**Progettazione:** restituisce la radice quadrata di un numero complesso
COMP.SEN	**Progettazione:** restituisce il seno di un numero complesso
COMP.SOMMA	**Progettazione:** restituisce la somma di numeri complessi
COMPLESSO	**Progettazione:** converte i coefficienti reali e immaginari in numeri complessi
CONCATENA	**Testo:** unisce diversi elementi di testo in uno solo
CONFIDENZA	**Compatibilità:** restituisce l'intervallo di confidenza per una media di popolazione
CONFIDENZA.NORM	**Statistiche:** restituisce l'intervallo di confidenza per una media di popolazione
CONFIDENZA.T	**Statistiche:** restituisce l'intervallo di confidenza per una media di popolazione utilizzando una distribuzione t di Student

CONFRONTA	**Ricerca e riferimento:** ricerca i valori in un riferimento o in una matrice
CONTA.NUMERI	**Statistiche:** conta la quantità di numeri nell'elenco di argomenti
CONTA.PIÙ.SE	**Statistiche:** conta il numero di celle in un intervallo che soddisfano più criteri
CONTA.SE	**Statistiche:** conta il numero di celle in un intervallo che soddisfano i criteri specificati
CONTA.SET.CUBO	**Cubo:** restituisce il numero di elementi di un insieme
CONTA.VALORI	**Statistiche:** conta il numero di valori nell'elenco di argomenti
CONTA.VUOTE	**Statistiche:** conta il numero di celle vuote all'interno di un intervallo
CONVERTI	**Progettazione:** converte un numero da un sistema di misura in un altro
CORRELAZIONE	**Statistiche:** restituisce il coefficiente di correlazione tra due set di dati
COS	**Matematiche e trigonometriche:** restituisce il coseno di un numero
COSH	**Matematiche e trigonometriche:** restituisce il coseno iperbolico di un numero
COVARIANZA	**Compatibilità:** calcola la covarianza, la media dei prodotti delle deviazioni accoppiate
COVARIANZA.C	**Statistiche:** calcola la covarianza del campione, ovvero la media delle deviazioni dei prodotti di ogni coppia di coordinate in due set di dati
COVARIANZA.P	**Statistiche:** calcola la covarianza, la media dei prodotti delle deviazioni accoppiate
CRESCITA	**Statistiche:** restituisce i valori lungo una linea di tendenza esponenziale
CRIT.BINOM	**Compatibilità:** restituisce il valore più piccolo per il quale la distribuzione cumulativa binomiale risulta maggiore o uguale a un valore di criterio
CURTOSI	**Statistiche:** restituisce la curtosi di un set di dati
DATA	**Data e ora:** restituisce il numero seriale di una determinata data
DATA.CED.PREC	**Finanziarie:** restituisce un numero che rappresenta la data della cedola precedente alla data di liquidazione
DATA.CED.SUCC	**Finanziarie:** restituisce un numero che rappresenta la data della cedola successiva alla data di liquidazione
DATA.MESE	**Data e ora:** restituisce il numero seriale della data che rappresenta il numero di mesi indicato prima o dopo la data di inizio
DATA.VALORE	**Data e ora:** converte una data in formato testo in un numero seriale
DATITEMPOREALE	**Ricerca e riferimento:** recupera i dati in tempo reale da un programma che supporta l'automazione COM **Nota** Questa caratteristica non è disponibile in Excel Starter 2010 Per ulteriori informazioni sulle caratteristiche disponibili in Excel Starter, vedere Supporto delle caratteristiche in Excel Starter.

DB.CONTA.NUMERI	**Database:** conta le celle di un database contenenti numeri
DB.CONTA.VALORI	**Database:** conta le celle non vuote in un database
DB.DEV.ST	**Database:** restituisce una stima della deviazione standard sulla base di un campione di voci di un database selezionate
DB.DEV.ST.POP	**Database:** calcola la deviazione standard sulla base dell'intera popolazione delle voci di un database selezionate
DB.MAX	**Database:** restituisce il valore massimo dalle voci selezionate in un database
DB.MEDIA	**Database:** restituisce la media di voci del database selezionate
DB.MIN	**Database:** restituisce il valore minimo dalle voci selezionate in un database
DB.PRODOTTO	**Database:** moltiplica i valori in un determinato campo di record che soddisfano i criteri nel database
DB.SOMMA	**Database:** aggiunge i numeri nel campo colonna di record del database che soddisfano determinati criteri
DB.VALORI	**Database:** estrae da un database un singolo record che soddisfa i criteri specificati
DB.VAR	**Database:** restituisce una stima della varianza sulla base di un campione di voci di un database selezionate
DB.VAR.POP	**Database:** calcola la varianza sulla base di tutte le voci di un database selezionate
DECIMALE.BINARIO	**Progettazione:** converte un numero decimale in binario
DECIMALE.HEX	**Progettazione:** converte un numero decimale in esadecimale
DECIMALE.OCT	**Progettazione:** converte un numero decimale in ottale
DELTA	**Progettazione:** verifica se due valori sono uguali
DESTRA, DESTRA.B	**Testo:** restituisce il carattere più a destra di un valore di testo
DEV.Q	**Statistiche:** restituisce la somma dei quadrati delle deviazioni
DEV.ST	**Compatibilità:** stima la deviazione standard sulla base di un campione
DEV.ST.C	**Statistiche:** stima la deviazione standard sulla base di un campione
DEV.ST.P	**Statistiche:** calcola la deviazione standard sulla base di un'intera popolazione
DEV.ST.POP	**Compatibilità:** calcola la deviazione standard sulla base di un'intera popolazione
DEV.ST.POP.VALORI	**Statistiche:** calcola la deviazione standard sulla base dell'intera popolazione, inclusi i numeri, il testo e i valori logici
DEV.ST.VALORI	**Statistiche:** restituisce una stima della deviazione standard sulla base di un campione, inclusi i numeri, il testo e i valori logici
DISPARI	**Matematiche e trigonometriche:** arrotonda un numero per eccesso al più vicino intero dispari
DISTRIB.BETA	**Compatibilità:** restituisce la funzione di distribuzione cumulativa beta

DISTRIB.BETA.N	**Statistiche:** restituisce la funzione di distribuzione cumulativa beta
DISTRIB.BINOM	**Compatibilità:** restituisce la distribuzione binomiale per il termine individuale
DISTRIB.BINOM.N	**Statistiche:** restituisce la distribuzione binomiale per il termine individuale
DISTRIB.BINOM.NEG	**Compatibilità:** restituisce la distribuzione binomiale negativa
DISTRIB.BINOM.NEG.N	**Statistiche:** restituisce la distribuzione binomiale negativa
DISTRIB.CHI	**Compatibilità:** restituisce la probabilità a una coda per la distribuzione del chi quadrato
DISTRIB.CHIQUAD	**Statistiche:** restituisce la funzione densità di probabilità cumulativa beta
DISTRIB.CHIQUAD.DS	**Statistiche:** restituisce la probabilità a una coda per la distribuzione del chi quadrato
DISTRIB.EXP	**Compatibilità:** restituisce la distribuzione esponenziale
DISTRIB.F	**Compatibilità:** restituisce la distribuzione di probabilità F
DISTRIB.F.DS	**Statistiche:** restituisce la distribuzione di probabilità F
DISTRIB.GAMMA	**Compatibilità:** restituisce la distribuzione gamma
DISTRIB.GAMMA.N	**Statistiche:** restituisce la distribuzione gamma
DISTRIB.IPERGEOM	**Compatibilità:** restituisce la distribuzione ipergeometrica
DISTRIB.IPERGEOM.N	**Statistiche:** restituisce la distribuzione ipergeometrica
DISTRIB.LOGNORM	**Compatibilità:** restituisce la distribuzione lognormale cumulativa
DISTRIB.LOGNORM.N	**Statistiche:** restituisce la distribuzione lognormale cumulativa
DISTRIB.NORM	**Compatibilità:** restituisce la distribuzione cumulativa normale
DISTRIB.NORM.N	**Statistiche:** restituisce la distribuzione cumulativa normale
DISTRIB.NORM.ST	**Compatibilità:** restituisce la distribuzione cumulativa normale standard
DISTRIB.NORM.ST.N	**Statistiche:** restituisce la distribuzione cumulativa normale standard
DISTRIB.POISSON	**Statistiche:** restituisce la distribuzione di probabilità di Poisson
DISTRIB.T	**Statistiche:** restituisce i punti percentuali (probabilità) della distribuzione t di Student
DISTRIB.T	**Compatibilità:** restituisce la distribuzione t di Student
DISTRIB.T.2T	**Statistiche:** restituisce i punti percentuali (probabilità) della distribuzione t di Student
DISTRIB.T.DS	**Statistiche:** restituisce la distribuzione t di Student
DISTRIB.WEIBULL	**Statistiche:** restituisce la distribuzione di Weibull
DISTRIBF	**Statistiche:** restituisce la distribuzione di probabilità F
DURATA	**Finanziarie:** restituisce la durata annuale di un titolo con i pagamenti di interesse periodico

DURATA.M	**Finanziarie:** restituisce la durata Macauley modificata per un titolo con un valore presunto di € 100
E	**Logiche:** restituisce VERO se tutti gli argomenti hanno valore VERO
EFFETTIVO	**Finanziarie:** restituisce il tasso di interesse effettivo annuo
ERR.STD.YX	**Statistiche:** restituisce l'errore standard del valore previsto per y per ogni valore di x nella regressione
ERRORE.TIPO	**Informative:** restituisce un numero che corrisponde a un tipo di errore
ESATTO	**Testo:** verifica se due valori di testo sono uguali
ESC.PERCENT.RANGO	**Statistiche:** restituisce il rango di un valore in un set di dati come percentuale (0..1, estremi esclusi) del set di dati
ESC.PERCENTILE	**Statistiche:** restituisce il k-esimo dato percentile dei valori in un intervallo, dove k è nell'intervallo 0..1, estremi esclusi
ESC.QUARTILE	**Statistiche:** restituisce il quartile del set di dati, in base ai valori del percentile da 0..1, estremi esclusi
EUROCONVERT	**Componenti aggiuntivi e automazione:** consente di convertire un numero in euro, un valore dal formato euro a un formato in una valuta dei paesi membri dell'Unione Europea oppure un valore da una delle valute dei paesi dell'Unione Europea in quella di un altro stato utilizzando l'euro come intermediario (triangolazione) **Nota** Questa caratteristica non è disponibile in Excel Starter 2010 Per ulteriori informazioni sulle caratteristiche disponibili in Excel Starter, vedere Supporto delle caratteristiche in Excel Starter.
EXP	**Matematiche e trigonometriche:** restituisce il numero *e* elevato alla potenza di un numero assegnato
EXPON.DIST	**Statistiche:** restituisce la distribuzione esponenziale
FALSO	**Logiche:** restituisce il valore logico FALSO
FATT.DOPPIO	**Matematiche e trigonometriche:** restituisce il fattoriale doppio di un numero
FATTORIALE	**Matematiche e trigonometriche:** restituisce il fattoriale di un numero
FINE.MESE	**Data e ora:** restituisce il numero seriale dell'ultimo giorno del mese, prima o dopo un determinato numero di mesi
FISHER	**Statistiche:** restituisce la trasformazione di Fisher
FISSO	**Testo:** formatta un numero come testo con un numero fisso di decimali
FRAZIONE.ANNO	**Data e ora:** restituisce la frazione dell'anno che rappresenta il numero dei giorni compresi tra una data iniziale e una data finale
FREQUENZA	**Statistiche:** restituisce la distribuzione di frequenza come matrice verticale
FUNZ.ERRORE	**Progettazione:** restituisce la funzione di errore

FUNZ.ERRORE.COMP	**Progettazione:** restituisce la funzione di errore complementare
FUNZ.ERRORE.COMP.PRECISA	**Progettazione:** Restituisce la funzione FUNZ.ERRORE complementare integrata tra x e infinito
FUNZ.ERRORE.PRECISA	**Progettazione:** restituisce la funzione di errore
FURIGANA	**Testo:** estrae i caratteri fonetici (furigana) da una stringa di testo
GIORNI.CED	**Finanziarie:** restituisce il numero dei giorni relativi al periodo della cedola che contiene la data di liquidazione
GIORNI.CED.INIZ.LIQ	**Finanziarie:** restituisce il numero dei giorni compresi tra l'inizio del periodo di durata della cedola e la data di liquidazione
GIORNI.CED.NUOVA	**Finanziarie:** restituisce il numero di giorni compresi tra la data di liquidazione e la data della cedola successiva
GIORNI.LAVORATIVI.TOT	**Data e ora:** restituisce il numero totale dei giorni lavorativi compresi fra due date
GIORNI.LAVORATIVI.TOT.INTL	**Data e ora:** restituisce il numero totale di giorni lavorativi compresi fra due date utilizzando parametri per indicare quali e quanti giorni sono giorni festivi
GIORNO	**Data e ora:** converte un numero seriale in un giorno del mese
GIORNO.LAVORATIVO	**Data e ora:** restituisce il numero seriale della data prima o dopo un determinato numero di giorni lavorativi
GIORNO.LAVORATIVO.INTL	restituisce la data, espressa come numero seriale, del giorno precedente o successivo a un numero specificato di giorni lavorativi utilizzando parametri per indicare quali e quanti giorni sono giorni festivi
GIORNO.SETTIMANA	**Data e ora:** converte un numero seriale in un giorno della settimana
GIORNO360	**Data e ora:** calcola il numero di giorni compreso tra due date basandosi su un anno di 360 giorni
GRADI	**Matematiche e trigonometriche:** converte i radianti in gradi
GRANDE	**Statistiche:** restituisce il k-esimo valore più grande di un set di dati
HEX.BINARIO	**Progettazione:** converte un numero esadecimale in binario
HEX.DECIMALE	**Progettazione:** converte un numero esadecimale in decimale
HEX.OCT	**Progettazione:** converte un numero esadecimale in ottale
IDENTIFICATORE.REGISTRO	**Componenti aggiuntivi e automazione:** restituisce l'identificatore di registro della DLL o della risorsa codice specificata che è stata registrata in precedenza **Nota** Questa caratteristica non è disponibile in Excel Starter 2010 Per ulteriori informazioni sulle caratteristiche disponibili in Excel Starter, vedere Supporto delle caratteristiche in Excel Starter.
INC.PERCENT.RANGO	**Statistiche:** restituisce il rango di un valore in un set di dati come percentuale
INC.PERCENTILE	**Statistiche:** restituisce il k-esimo dato percentile di valori in un intervallo

INC.QUARTILE	**Statistiche:** restituisce il quartile di un set di dati
INDICE	**Ricerca e riferimento:** utilizza un indice per scegliere un valore da un riferimento o da una matrice
INDIRETTO	**Ricerca e riferimento:** restituisce un riferimento specificato da un valore di testo
INDIRIZZO	**Ricerca e riferimento:** restituisce un riferimento come testo in una singola cella di un foglio di lavoro
INFO.DATI.TAB.PIVOT	**Componenti aggiuntivi e automazione:** restituisce i dati memorizzati in un rapporto di tabella pivot
INT	**Matematiche e trigonometriche:** arrotonda un numero per difetto all'intero più vicino
INT.CUMUL	**Finanziarie:** restituisce l'interesse cumulativo pagato fra due periodi
INT.MATURATO.PER	**Finanziarie:** restituisce l'interesse maturato di un titolo che paga interessi periodici
INT.MATURATO.SCAD	**Finanziarie:** restituisce l'interesse maturato di un titolo che paga interessi alla scadenza
INTERCETTA	**Statistiche:** restituisce l'intercetta della retta di regressione lineare
INTERESSE.RATA	**Finanziarie:** calcola l'interesse di un investimento pagato durante un periodo specifico
INTERESSI	**Finanziarie:** restituisce il valore degli interessi per un investimento relativo a un periodo specifico
INV.BETA	**Compatibilità:** restituisce l'inversa della funzione distribuzione cumulativa per una distribuzione beta specificata
INV.BETA.N	**Statistiche:** restituisce l'inversa della funzione distribuzione cumulativa per una distribuzione beta specificata
INV.BINOM	**Statistiche:** restituisce il valore più piccolo per il quale la distribuzione cumulativa binomiale risulta maggiore o uguale a un valore di criterio
INV.CHI	**Compatibilità:** restituisce l'inversa della distribuzione a una coda del chi quadrato
INV.CHIQUAD	**Statistiche:** restituisce la funzione densità di probabilità cumulativa beta
INV.CHIQUAD.DS	**Statistiche:** restituisce l'inversa della distribuzione a una coda del chi quadrato
INV.F	**Statistiche:** restituisce l'inversa della distribuzione di probabilità F
INV.F.DS	**Statistiche:** restituisce l'inversa della distribuzione di probabilità F
INV.FISHER	**Statistiche:** restituisce l'inversa della trasformazione di Fisher
INV.GAMMA	**Compatibilità:** restituisce l'inversa della distribuzione cumulativa gamma
INV.GAMMA.N	**Statistiche:** restituisce l'inversa della distribuzione cumulativa gamma
INV.LOGNORM	**Compatibilità:** restituisce l'inversa di una distribuzione cumulativa lognormale
INV.LOGNORM.N	**Statistiche:** restituisce l'inversa di una distribuzione cumulativa lognormale
INV.NORM	**Compatibilità:** restituisce l'inversa della distribuzione cumulativa normale

INV.NORM.N	**Statistiche:** restituisce l'inversa della distribuzione cumulativa normale
INV.NORM.S	R**Statistiche:** estituisce l'inversa della distribuzione normale standard cumulativa
INV.NORM.ST	**Compatibilità:** restituisce l'inversa della distribuzione normale standard cumulativa
INV.T	**Compatibilità:** restituisce l'inversa della distribuzione t di Student
INV.T.2T	**Statistiche:** Restituisce l'inversa della distribuzione t di Student
INVF	**Statistiche:** restituisce l'inversa della distribuzione di probabilità F
INVT	**Statistiche:** restituisce il valore t della distribuzione t di Student come funzione della probabilità e dei gradi di libertà
LIBERA	**Testo:** rimuove dal testo tutti i caratteri che non possono essere stampati
LN	**Matematiche e trigonometriche:** restituisce il logaritmo naturale di un numero
LN.GAMMA	**Statistiche:** restituisce il logaritmo naturale di una funzione gamma, Γ(x)
LN.GAMMA.PRECISA	**Statistiche:** restituisce il logaritmo naturale di una funzione gamma, Γ(x)
LOG	**Matematiche e trigonometriche:** restituisce il logaritmo di un numero in una base specificata
LOG10	**Matematiche e trigonometriche:** restituisce il logaritmo in base 10 di un numero
LUNGHEZZA, LUNGB	**Testo:** restituisce il numero di caratteri di una stringa di testo
MAIUSC	**Testo:** Converte il testo in maiuscolo
MAIUSC.INIZ	**Testo:** converte in maiuscolo la prima lettera di ogni parola di un valore di testo
MATR.DETERM	**Matematiche e trigonometriche:** restituisce il determinante di una matrice
MATR.INVERSA	**Matematiche e trigonometriche:** restituisce l'inversa di una matrice
MATR.PRODOTTO	**Matematiche e trigonometriche:** Restituisce il prodotto di due matrici
MATR.SOMMA.PRODOTTO	**Matematiche e trigonometriche:** restituisce la somma dei prodotti dei componenti corrispondenti della matrice
MATR.TRASPOSTA	**Ricerca e riferimento:** restituisce la trasposizione di una matrice
MAX	**Statistiche:** restituisce il valore più grande di un elenco di argomenti
MAX.VALORI	**Statistiche:** restituisce il valore massimo in un elenco di argomenti, inclusi i numeri, il testo e i valori logici
MCD	**Matematiche e trigonometriche:** restituisce il massimo comune divisore
MCM	**Matematiche e trigonometriche:** restituisce il minimo comune multiplo
MEDIA	**Statistiche:** restituisce la media degli argomenti
MEDIA.ARMONICA	**Statistiche:** restituisce la media armonica

MEDIA.DEV	**Statistiche:** restituisce la media delle deviazioni assolute dei valori rispetto alla loro media
MEDIA.GEOMETRICA	**Statistiche:** restituisce la media geometrica
MEDIA.PIÙ.SE	**Statistiche:** restituisce la media aritmetica di tutte le celle che soddisfano più criteri
MEDIA.SE	**Statistiche:** restituisce la media aritmetica di tutte le celle in un intervallo che soddisfano un determinato criterio
MEDIA.TRONCATA	**Statistiche:** restituisce la media della parte interna di un set di dati
MEDIA.VALORI	**Statistiche:** restituisce la media degli argomenti, inclusi i numeri, il testo e i valori logici
MEDIANA	**Statistiche:** restituisce la mediana dei numeri specificati
MEMBRO.CUBO	**Cubo:** restituisce un membro o una tupla in una gerarchia di cubi. Consente di verificare l'esistenza del membro o della tupla nel cubo
MEMBRO.CUBO.CON.RANGO	**Cubo:** restituisce l'n-esimo membro o il membro ordinato di un insieme. Consente di restituire uno o più elementi di un insieme, ad esempio l'agente di vendita migliore o i primi 10 studenti
MEMBRO.KPI.CUBO	**Cubo:** restituisce il nome, la proprietà e la misura di un indicatore di prestazioni chiave (KPI) e visualizza il nome e la proprietà nella cella. Un KPI è una misura quantificabile, ad esempio il profitto lordo mensile o il fatturato trimestrale dei dipendenti, utilizzata per il monitoraggio delle prestazioni di un'organizzazione
MESE	**Data e ora:** converte un numero seriale in un mese
MIN	**Statistiche:** restituisce il valore più piccolo di un elenco di argomenti
MIN.VALORI	**Statistiche:** restituisce il valore più piccolo in un elenco di argomenti, inclusi i numeri, il testo e i valori logici
MINUSC	**Testo:** converte il testo in lettere minuscole
MINUTI	**Data e ora:** converte un numero seriale in un minuto
MODA	**Compatibilità:** restituisce il valore più comune in un set di dati
MODA.MULT	**Statistiche:** restituisce una matrice verticale dei valori più ricorrenti in una matrice o intervallo di dati
MODA.SNGL	**Statistiche:** restituisce il valore più comune in un set di dati
MULTINOMIALE	**Matematiche e trigonometriche:** restituisce il multinomiale di un insieme di numeri
NOMINALE	**Finanziarie:** restituisce il tasso di interesse nominale annuale
NON	**Logiche:** inverte la logica dell'argomento
NON.DISP	**Informative:** restituisce il valore errore #N/D
NORMALIZZA	**Statistiche:** restituisce un valore normalizzato
NUM	**Informative:** restituisce un valore convertito in numero
NUM.CED	**Finanziarie:** restituisce il numero di cedole pagabili fra la data di liquidazione e la data di scadenza

NUM.RATE	**Finanziarie:** restituisce il numero di periodi relativi a un investimento
NUM.SETTIMANA	**Data e ora:** converte un numero seriale in un numero che rappresenta la posizione numerica di una settimana nell'anno
O	**Logiche:** restituisce VERO se un argomento qualsiasi è VERO
OCT.BINARIO	**Progettazione:** converte un numero ottale in binario
OCT.DECIMALE	**Progettazione:** converte un numero ottale in decimale
OCT.HEX	**Progettazione:** converte un numero ottale in esadecimale
OGGI	**Data e ora:** restituisce il numero seriale relativo alla data odierna
ORA	**Data e ora:** converte un numero seriale in un'ora
ORARIO	**Data e ora:** restituisce il numero seriale di un determinato orario
ORARIO.VALORE	**Data e ora:** converte un orario in formato testo in un numero seriale
ORDINAMENTO.JIS	**Testo:** modifica le lettere inglesi o il katakana a singolo byte all'interno di una stringa di caratteri in caratteri a doppio byte
P.RATA	**Finanziarie:** restituisce il pagamento sul capitale di un investimento per un dato periodo
PARI	**Matematiche e trigonometriche:** arrotonda il valore assoluto di un numero per eccesso al più vicino intero pari
PEARSON	**Statistiche:** restituisce il coefficiente del momento di correlazione di Pearson
PENDENZA	**Statistiche:** restituisce la pendenza di una retta di regressione lineare
PERCENT.RANGO	**Compatibilità:** restituisce il rango di un valore in un set di dati come percentuale
PERCENTILE	**Compatibilità:** restituisce il k-esimo dato percentile di valori in un intervallo
PERMUTAZIONE	**Statistiche:** restituisce il numero delle permutazioni per un determinato numero di oggetti
PI.GRECO	**Matematiche e trigonometriche:** restituisce il valore di pi greco
PICCOLO	**Statistiche:** restituisce il k-esimo valore più piccolo di un set di dati
POISSON	**Compatibilità:** restituisce la distribuzione di probabilità di Poisson
POTENZA	**Matematiche e trigonometriche:** restituisce il risultato di un numero elevato a potenza
PREVISIONE	**Statistiche:** restituisce i valori lungo una tendenza lineare
PREZZO	**Finanziarie:** restituisce il prezzo di un titolo dal valore nominale di € 100 che paga interessi periodici
PREZZO.PRIMO.IRR	**Finanziarie:** restituisce il prezzo di un titolo dal valore nominale di € 100 avente il primo periodo di durata irregolare
PREZZO.SCAD	**Finanziarie:** restituisce il prezzo di un titolo dal valore nominale di € 100 che paga gli interessi alla scadenza

PREZZO.SCONT	**Finanziarie:** restituisce il prezzo di un titolo scontato dal valore nominale di € 100
PREZZO.ULTIMO.IRR	**Finanziarie:** restituisce il prezzo di un titolo dal valore nominale di € 100 avente l'ultimo periodo di durata irregolare
PROBABILITÀ	**Statistiche:** restituisce la probabilità che dei valori in un intervallo siano compresi tra due limiti
PRODOTTO	**Matematiche e trigonometriche:** moltiplica gli argomenti
PROPRIETÀ.MEMBRO.CUBO	**Cubo:** restituisce il valore di una proprietà di un membro del cubo. Consente di verificare l'esistenza di un nome di membro all'interno del cubo e di restituire la proprietà specificata per tale membro
QUARTILE	**Compatibilità:** restituisce il quartile di un set di dati
QUOZIENTE	**Matematiche e trigonometriche:** restituisce il quoziente di una divisione
RADIANTI	**Matematiche e trigonometriche:** converte i gradi in radianti
RADQ	**Matematiche e trigonometriche:** restituisce una radice quadrata positiva
RADQ.PI.GRECO	**Matematiche e trigonometriche:** restituisce la radice quadrata di (num * pi)
RANGO	**Compatibilità:** restituisce il rango di un numero in un elenco di numeri
RANGO.MEDIA	**Statistiche:** Restituisce il rango di un numero in un elenco di numeri
RANGO.UG	**Statistiche:** restituisce il rango di un numero in un elenco di numeri
RATA	**Finanziarie:** restituisce il pagamento periodico per un'annualità
REGR.LIN	**Statistiche:** restituisce i parametri di una tendenza lineare
REGR.LOG	**Statistiche:** restituisce i parametri di una linea di tendenza esponenziale
REND	**Finanziarie:** restituisce il rendimento di un titolo che frutta interessi periodici
REND.PRIMO.IRR	**Finanziarie:** restituisce il rendimento di un titolo avente il primo periodo di durata irregolare
REND.SCAD	**Finanziarie:** restituisce il rendimento annuo di un titolo che paga interessi alla scadenza
REND.TITOLI.SCONT	**Finanziarie:** restituisce il rendimento annuale di un titolo scontato, ad esempio un buono del tesoro
REND.ULTIMO.IRR	**Finanziarie:** restituisce il rendimento di un titolo avente l'ultimo periodo di durata irregolare
RESTO	**Matematiche e trigonometriche:** restituisce il resto della divisione
RICERCA, CERCA.B	**Testo:** rileva un valore di testo all'interno di un altro (non distingue tra maiuscole e minuscole)
RICEV.SCAD	**Finanziarie:** restituisce l'ammontare ricevuto alla scadenza di un titolo interamente investito
RICHIAMA	**Componenti aggiuntivi e automazione:** chiama una routine in una DLL o in una risorsa codice **Nota** Questa caratteristica non è disponibile in Microsoft Excel Starter 2010 Per ulteriori informazioni sulle caratteristiche disponibili in Excel Starter, vedere Supporto delle

	caratteristiche in Excel Starter.
RIF.COLONNA	**Ricerca e riferimento:** restituisce il numero di colonna di un riferimento
RIF.RIGA	**Ricerca e riferimento:** restituisce il numero di riga di un riferimento
RIGHE	**Ricerca e riferimento:** restituisce il numero di righe in un riferimento
RIMPIAZZA, SOSTITUISCI.B	**Testo:** sostituisce i caratteri all'interno di un testo
RIPETI	**Testo:** ripete un testo per il numero di volte specificato
ROMANO	**Matematiche e trigonometriche:** converte un numero in numero romano, sotto forma di testo
RQ	**Statistiche:** restituisce la radice quadrata del coefficiente di momento di correlazione di Pearson
SCARTO	**Ricerca e riferimento:** restituisce uno scarto di riferimento da un determinato riferimento
SCEGLI	**Ricerca e riferimento:** sceglie un valore da un elenco di valori
SE	**Logiche:** specifica un test logico da eseguire
SE.ERRORE	**Logiche:** restituisce un valore specificato dall'utente se la formula restituisce un errore. In caso contrario, restituisce il risultato della formula
SECONDI	**Data e ora:** converte un numero seriale in un secondo
SEGNO	**Matematiche e trigonometriche:** restituisce il segno di un numero
SEN	**Matematiche e trigonometriche:** restituisce il seno dell'angolo specificato
SENH	**Matematiche e trigonometriche:** restituisce il seno iperbolico di un numero
SET.CUBO	**Cubo:** definisce un insieme di tuple o membri calcolati mediante l'invio di un'espressione di insieme al cubo sul server. In questo modo l'insieme viene creato e restituito a Microsoft Excel.
SINISTRA, SINISTRAB	**Testo:** restituisce il carattere più a sinistra di un valore di testo
SOGLIA	**Progettazione:** verifica se un numero è maggiore del valore di soglia
SOMMA	**Matematiche e trigonometriche:** somma gli argomenti
SOMMA.DIFF.Q	**Matematiche e trigonometriche:** restituisce la somma della differenza dei quadrati dei valori corrispondenti di due matrici
SOMMA.PIÙ.SE	**Matematiche e trigonometriche:** somma le celle di un intervallo che soddisfano più criteri
SOMMA.Q	**Matematiche e trigonometriche:** restituisce la somma dei quadrati degli argomenti
SOMMA.Q.DIFF	**Matematiche e trigonometriche:** restituisce la somma dei quadrati delle differenze dei valori corrispondenti di due matrici
SOMMA.SE	**Matematiche e trigonometriche:** somma le celle specificate secondo un criterio assegnato
SOMMA.SERIE	**Matematiche e trigonometriche:** restituisce la somma di una serie di potenze data dalla formula

SOMMA.SOMMA.Q	**Matematiche e trigonometriche:** restituisce la somma della somma dei quadrati dei valori corrispondenti di due matrici
SOSTITUISCI	**Testo:** sostituisce il nuovo testo al testo contenuto in una stringa
SQL.REQUEST	**Componenti aggiuntivi e automazione:** stabilisce collegamenti con origini dati esterne, esegue una query da un foglio di lavoro e quindi restituisce il risultato, ad esempio una matrice, senza richiedere la programmazione di una macro **Nota** Questa caratteristica non è disponibile in Excel Starter 2010 Per ulteriori informazioni sulle caratteristiche disponibili in Excel Starter, vedere Supporto delle caratteristiche in Excel Starter.
STRINGA.ESTRAI, MEDIA.B	**Testo:** restituisce un numero specifico di caratteri di una stringa di testo a partire dalla posizione specificata
SUBTOTALE	**Matematiche e trigonometriche:** restituisce un subtotale in un elenco o in un database
T	**Testo:** converte gli argomenti in testo
TAN	**Matematiche e trigonometriche:** restituisce la tangente di un numero
TANH	**Matematiche e trigonometriche:** restituisce la tangente iperbolica di un numero
TASSO	**Finanziarie:** restituisce il tasso di interesse per un periodo di un'annualità
TASSO.INT	**Finanziarie:** restituisce il tasso di interesse per un titolo interamente investito
TASSO.SCONTO	**Finanziarie:** restituisce il tasso di sconto per un titolo
TENDENZA	**Statistiche:** restituisce i valori lungo una linea di tendenza lineare
TEST.CHI	**Compatibilità:** restituisce il test per l'indipendenza
TEST.CHIQUAD	**Statistiche:** restituisce il test per l'indipendenza
TEST.F	**Compatibilità:**
TEST.T	**Compatibilità:** restituisce la probabilità associata a un test t di Student
TEST.Z	**Compatibilità:** restituisce il valore di probabilità a una coda di un test z
TESTF	**Statistiche:** restituisce il risultato di un test F
TESTO	**Testo:** formatta un numero e lo converte in testo
TESTT	**Statistiche:** restituisce la probabilità associata a un test t di Student
TESTZ	**Statistiche:** restituisce il valore di probabilità a una coda di un test z
TIPO	**Informative:** restituisce un numero che indica il tipo di dati relativo a un valore
TIR.COST	**Finanziarie:** restituisce il tasso di rendimento interno per una serie di flussi di cassa
TIR.VAR	**Finanziarie:** restituisce il tasso di rendimento interno in cui i flussi di cassa positivi e negativi

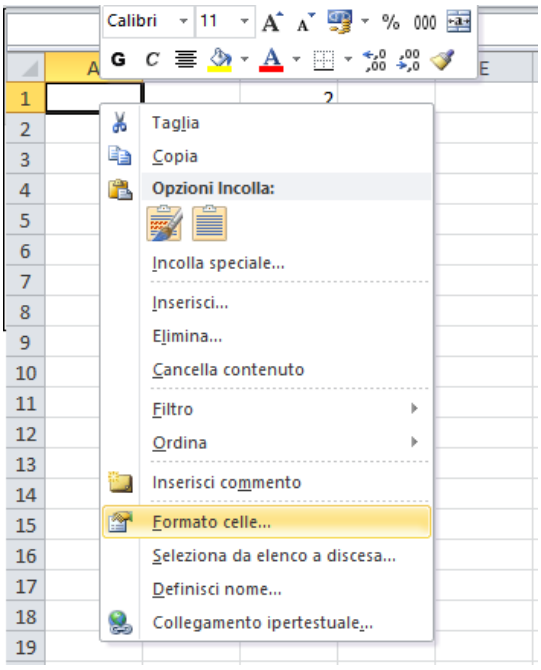
	sono finanziati a tassi diversi
TIR.X	**Finanziarie:** restituisce il tasso di rendimento interno di un impiego di flussi di cassa non necessariamente periodico
TRONCA	**Matematiche e trigonometriche:** tronca la parte decimale di un numero
TROVA, TROVA.B	**Testo:** rileva un valore di testo all'interno di un altro (distingue tra maiuscole e minuscole)
VA	**Finanziarie:** restituisce il valore attuale di un investimento
VAL.DISPARI	**Informative:** restituisce VERO se il numero è dispari
VAL.ERR	**Informative:** restituisce VERO se il valore è un valore di errore qualsiasi tranne #N/D
VAL.ERRORE	**Informative:** restituisce VERO se il valore è un valore di errore qualsiasi
VAL.FUT	**Finanziarie:** restituisce il valore futuro di un investimento
VAL.FUT.CAPITALE	**Finanziarie:** restituisce il valore futuro di un capitale iniziale dopo aver applicato una serie di tassi di interesse composti
VAL.LOGICO	**Informative:** restituisce VERO se il valore è un valore logico
VAL.NON.DISP	**Informative:** restituisce VERO se il valore è un valore di errore #N/D
VAL.NON.TESTO	**Informative:** restituisce VERO se il valore non è in forma di testo
VAL.NUMERO	**Informative:** restituisce VERO se il valore è un numero
VAL.PARI	**Informative:** restituisce VERO se il numero è pari
VAL.RIF	**Informative:** restituisce VERO se il valore è un riferimento
VAL.TESTO	**Informative:** restituisce VERO se il valore è in forma di testo
VAL.VUOTO	**Informative:** restituisce VERO se il valore è vuoto
VALORE	**Testo:** converte un argomento di testo in numero
VALORE.CUBO	**Cubo:** restituisce un valore aggregato da un cubo
VALUTA	**Testo:** converte un numero in testo, utilizzando il formato valuta € (euro)
VALUTA.DEC	**Finanziarie:** converte un prezzo valuta espresso come frazione in prezzo valuta espresso come numero decimale
VALUTA.FRAZ	**Finanziarie:** converte un prezzo valuta espresso come numero decimale in prezzo valuta espresso come frazione
VAN	**Finanziarie:** restituisce il valore attuale netto di un investimento basato su una serie di flussi di cassa periodici e sul tasso di sconto
VAN.X	**Finanziarie:** restituisce il valore attuale netto di un impiego di flussi di cassa non necessariamente periodico
VAR	**Compatibilità:** stima la varianza sulla base di un campione

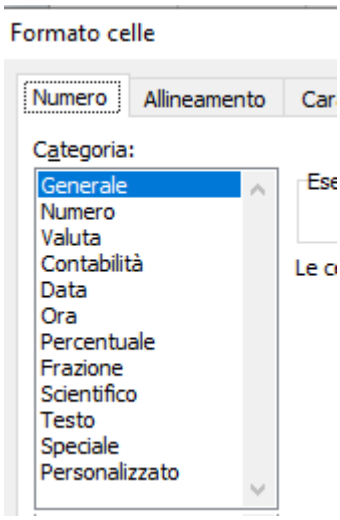
VAR.C	**Statistiche:** stima la varianza sulla base di un campione
VAR.P	**Statistiche:** restituisce la varianza sulla base dell'intera popolazione
VAR.POP	**Compatibilità:** restituisce la varianza sulla base dell'intera popolazione
VAR.POP.VALORI	**Statistiche:** calcola la varianza sulla base dell'intera popolazione, inclusi i numeri, il testo e i valori logici
VAR.VALORI	**Statistiche:** stima la varianza sulla base di un campione, inclusi i numeri, il testo e i valori logici
VERO	**Logiche:** restituisce il valore logico VERO
WEIBULL	**Compatibilità:** calcola la varianza sulla base dell'intera popolazione, inclusi i numeri, il testo e i valori logici

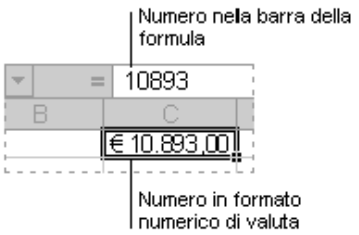
I diversi formati numerici e le loro proprietà

In Excel per ogni cella numerica è possibile definire un formato, tra questi uno molto utile è il formato valuta che permette di fare calcoli economici direttamente in tutte le monete del mondo.

Per abilitare il formato valuta è sufficiente cliccare sulla cella e accedere alle proprietà da dove si può definire il formato.







Tecniche per formattare il foglio di lavoro

La formattazione del foglio Excel è utile per rendere più chiara la lettura dei dati che mostrano quantità o schedulazioni.

Esistono degli stili standard che Excel mette a disposizione e che possono essere associate porzione della pagina oppure tutta la griglia.

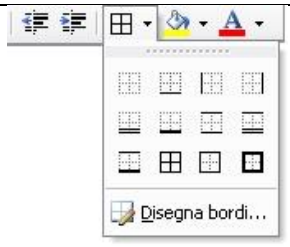
Lo stile viene suggerito sotto forma di elenco e va scelto quello che sia più adeguato al lavoro da realizzare, ad esempio ad una tabella sarà opportuno associare uno stile "Contabilità" se menziona valute economiche o simili.

Per scegliere il nostro stile, basterà selezionare una cella qualsiasi del foglio di calcolo, e usare il comando Formato e scegliere Formattazione automatica. Apparirà la finestra Formattazione automatica, da cui sarà possibile scegliere uno degli stili presenti al suo interno, come mostrato in figura.

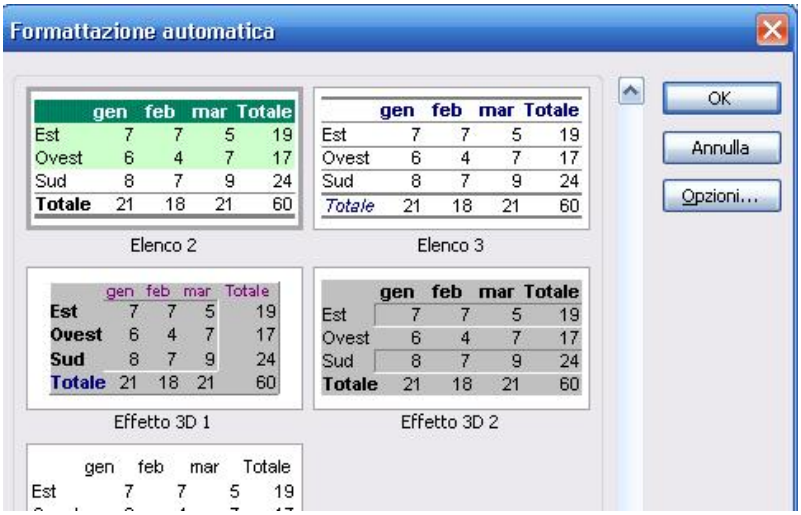
Per formattare manualmente il foglio elettronico scegliere il comando Modifica -> Cancella -> Formati.

Per l'attivazione della formattazione manuale che è una procedura che è meglio usare nel momento in cui ci accorgiamo che il nostro prospetto è abbastanza complesso e difficilmente formattabile attraverso la funzione automatica, bisogna tenere a mente che i principali elementi di formattazione si assegnano a una cella o a un intervallo di celle usando i comandi presenti all'interno della corrispondente barra degli strumenti Formattazione.

La barra Formattazione presenta diversi gruppi di pulsanti a cui corrispondono diverse funzioni. Vediamo alcune regole di stile.

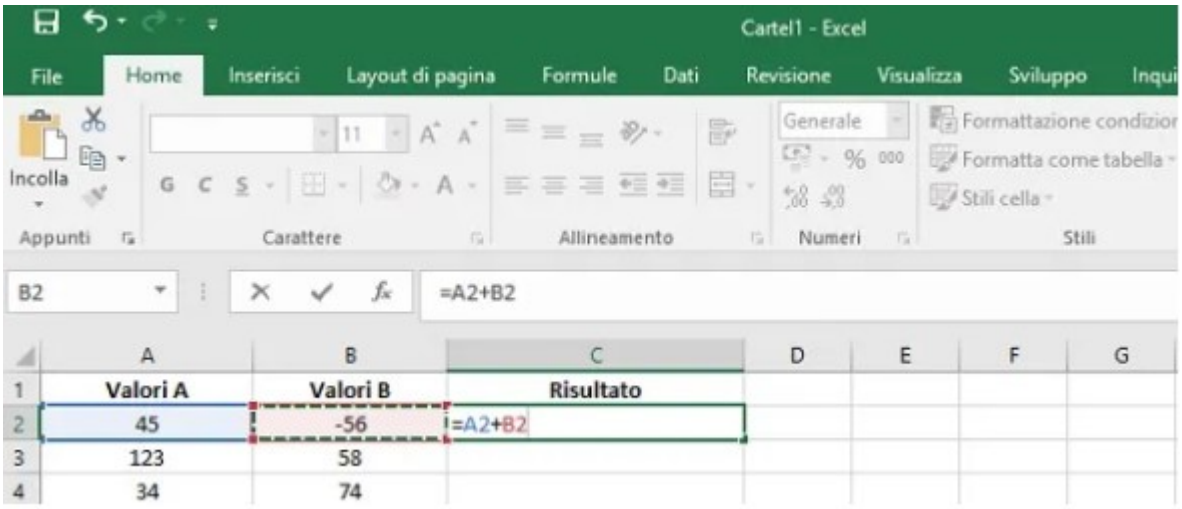
	 Questa barra permette di allineare il testo a destra al centro o sinistra, azione che si chiama "giustificazione". Unisce a centra il contenuto di più celle. Assegna significato monetario al contenuto numerico "valuta".	
Formattazione testo	Definisce le cifre decimali del numero	Formata i bordi di tabelle

La formattazione automatica porta a questi template.



Eseguire calcoli con il foglio elettronico

È possibile eseguire calcoli semplici e diretti tenendo a mente che gli operandi sono variabili identificati da celle, quindi se devo fare una somma è sufficiente scrivere "=" sulla cella finale e cliccare in sequenza sulle celle che voglio sommare, anche più di due, mettendo il "+" tra un operando e l'altro, come mostrato in figura.



I simboli per gli operatori aritmetici semplici sono:

- Il segno + per la somma algebrica
- Il – per la sottrazione
- L'asterisco * per la moltiplicazione
- Lo slash, / per la divisione

La colonna del risultato, in questo esempio C, contiene la formula =A2+B2, se volessi applicare la stessa formula a tutte le celle sottostanti o alcune celle sottostanti basterà trascinare con il mouse questa cella verso il basso copiandone automaticamente le proprietà.

Le celle saranno parametrizzate, quindi in C3 verranno cambiati automaticamente gli indici da A2 e B2 in A3 e B3 e così via per quanto si trascinerà in basso il piccolo quadratino nero nel margine destro della cella C2.

Somma automatica

Questa funzione permette di addizionare tutte le celle di un set contiguo, ad esempio la colonna sulla verticale sovrastante oppure un set di più sezioni, interrotte dal simbolo : il comando per eseguire la somma automatica è presente sia nella scheda home che nella scheda formule.

Sintassi delle funzioni

La scrittura di funzioni in Excel implica l'utilizzo di formule che sono già in libreria, in grado di eseguire una moltitudine di calcoli a cui dobbiamo passare i dati per l'occasione.

Le funzioni sono suddivise in categorie ad esempio Matematiche, Statistiche, Testo, Logiche, Data e ora, Ricerca e riferimento.

Le funzioni sono 400 circa, il numero esatto varia a seconda della versione.

Tutte le funzioni hanno questa struttura:

nella cella in cui deve restituire il risultato, oppure nella barra fx scriviamo:

=nome_funzione(parametri)

Ecco alcuni esempi

1. =SOMMA(A1:A100),
2. =SE(A1>100;“Alto”;“Basso”),
3. CERCA.VERT(B4;A10:F99;2;0)

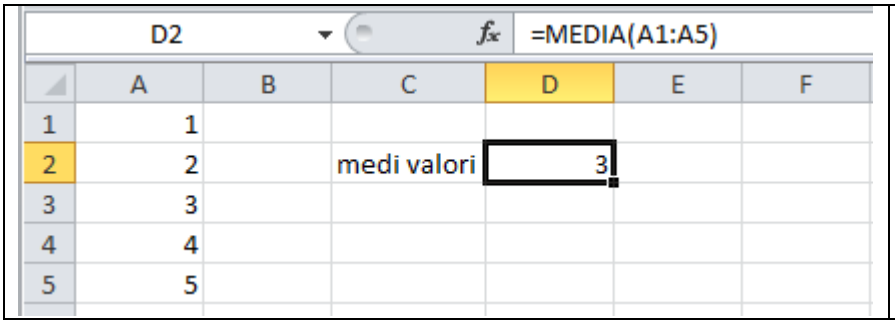
Il numero di parametri varia da funzione a funzione. I parametri possono essere obbligatori o facoltativi e sono divisi da punto e virgola.

La lista delle funzioni disponibili si possono ottenere dall’help in linea di Excel.

Ad esempio, per fare la media, si può fare così:

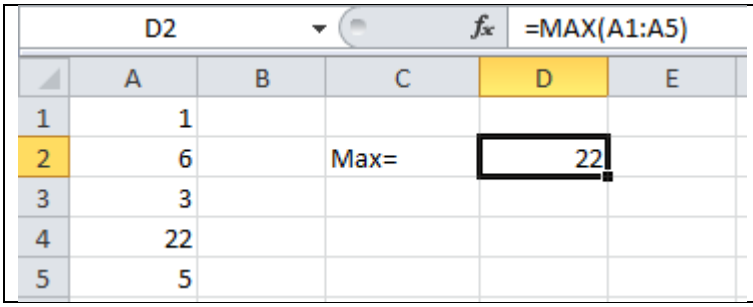
Restituisce la Media aritmetica

Categoria: Statistiche
Sintassi: =MEDIA(intervallo1;intervallo2;ecc.)
Esempio: =MEDIA(A1:A1OO)



Nell’immagine affianco viene mostrata la media dei primi 5 valori progressivi posti in colonna dal valore A1 al valore A5.
La funzione ha la sintassi:
=MEDIA(A1:A5)

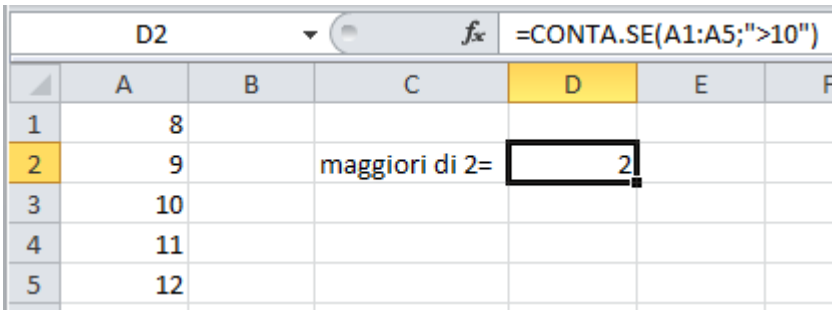
Ecco un esempio di come si può trovare il massimo in un intervallo predefinito



Per trovare il valore maggiore contenuto in un intervallo usare la funzione MAX come mostrato affianco.

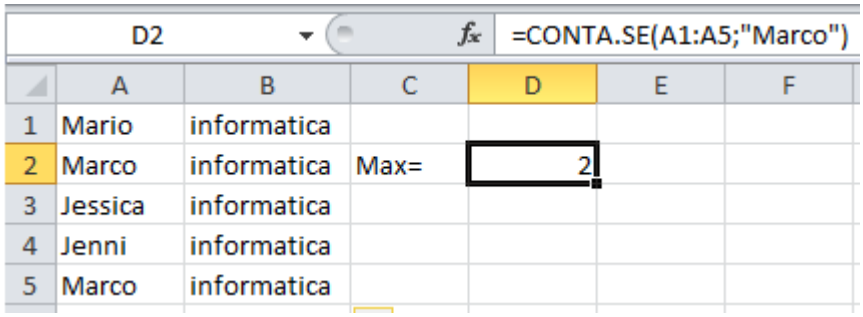
La sintassi è:
=MAX(A1:A5)

Le funzioni condizionali SE in inglese IF permettono di fare del scelte di calcolo in base agli operatori che useremo.
In questo esempio contiamo il numero di celle il cui contenuto numerico sia “>=10”.



È possibile creare una funzione abbastanza semplice che, in una azienda di spedizioni, conta quante destinazioni sono verso una specifica città, ad esempio “Milano”.

La funzione CONTA.SE conta il numero di celle di un intervallo che soddisfano un singolo criterio specificato. È ad esempio possibile contare tutte le celle che iniziano con una determinata lettera oppure tutte le celle in cui è contenuto un numero maggiore o miniore di un numero specificato.
Si supponga ad esempio che un foglio di lavoro contenga le lezioni di informatica nella colonna B e il nome del professore a cui è assegnato l’insegnamento nella colonna A.
È possibile utilizzare la funzione CONTA.SE per contare quanto volte si ripete il nome del professore nella colonna A e determinare in questo modo quante ore sono assegnate a quel professore.
Ad esempio:

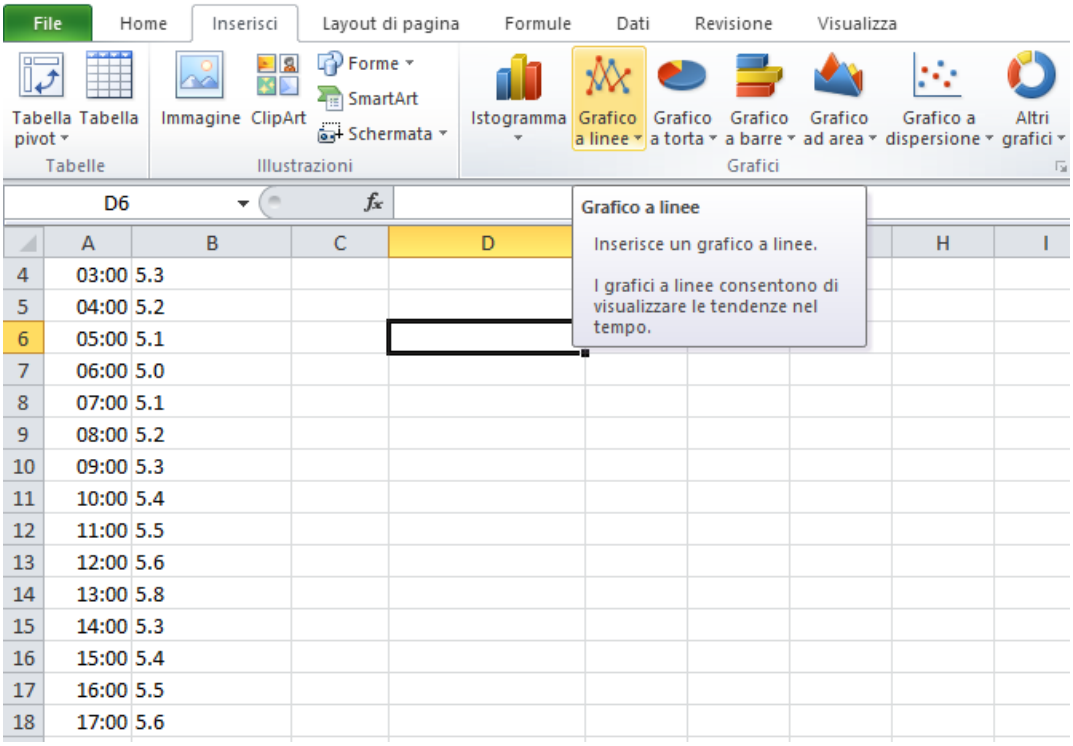


Creare grafici e operare con fogli e riferimenti

Consideriamo due colonne, A e B, contenenti set di dati, o come nel caso in esame, a colonna A contiene le ore del giorno mentre la colonna B un parametro, ad esempio un livello di acqua in una vasca.

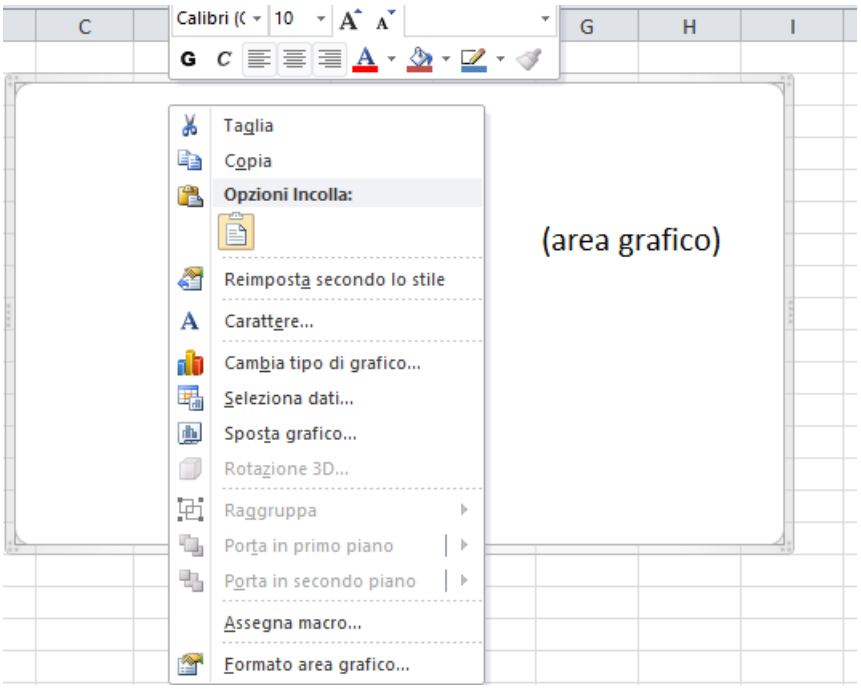
Vogliamo tracciare il grafico in funzione del tempo. Selezionare una cella in cui vogliamo introdurre il grafico, ad esempio D6.

Agire sul menù a tendina inserisci -> grafico.



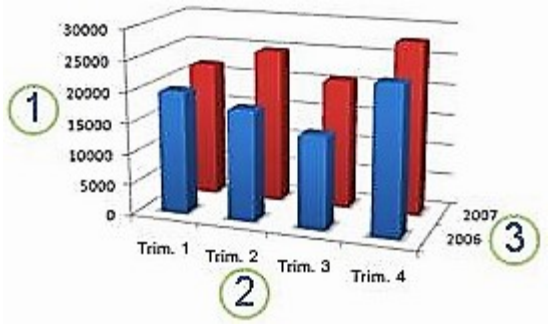
	A	B
4	03:00	5.3
5	04:00	5.2
6	05:00	5.1
7	06:00	5.0
8	07:00	5.1
9	08:00	5.2
10	09:00	5.3
11	10:00	5.4
12	11:00	5.5
13	12:00	5.6
14	13:00	5.8
15	14:00	5.3
16	15:00	5.4
17	16:00	5.5
18	17:00	5.6

Comparirà il rettangolo “Area grafico” di cui andremo a editare le proprietà.



Scegliere il tipo di grafico. Personalizzare grafici

In genere i grafici includono due assi utilizzati per misurare e suddividere in categorie i dati: un asse verticale (asse dei valori o asse Y) e un asse orizzontale (asse delle categorie o asse X). Gli istogrammi 3D e i grafici a cono e a piramide dispongono di un terzo asse, l'asse antero-posteriore (asse delle serie o asse Z), che consente di tracciare i dati nel grafico sulla relativa profondità. Nei grafici radar non è presente l'asse orizzontale, o asse delle categorie, mentre i grafici a torta e ad anello sono privi di assi.

	Passaggio 1 Asse verticale (asse dei valori) Passaggio 2 Asse orizzontale (asse delle categorie) Passaggio 3 Asse antero-posteriore (asse delle serie)

Funzioni matematiche

Le funzioni matematiche sono un sotto insieme delle funzioni disponibili in Excel, altri insiemi sono le statistiche le logiche per le enumerazioni ecc.

Le funzioni di enumerazione permettono di immettere una vasta libreria di funzioni predefinite dei fogli di lavoro per eseguire molteplici operazioni, oltre le classiche formule che eseguono calcoli matematici come le addizioni, sottrazioni, moltiplicazioni e divisioni.

Creare una formula richiede sempre due passaggi:

1. Cliccare su una cella e scegliere la formula, anche digitando le iniziale e confermando il suggerimento dell'intellisense (quello strumento software che cerca di intuire cosa stavo per scrivere e mostra un elenco di suggerimenti compatibili con ciò che ho iniziato a scrivere).
2. Fornire gli ARGOMENTI, ovvero i parametri che la specifica funzione matematica richiede.

Per la funzione ASS ad esempio viene utilizzato un numero come argomento. Per la funzione MAIUSC, che consente di convertire il testo minuscolo in testo maiuscolo, viene utilizzata invece una stringa di testo come argomento. Per la funzione PI.GRECO infine non viene utilizzato alcun argomento, poiché restituisce semplicemente il valore di pi greco (3,14159).

	A	B	C
1	=ASS(-2,34)		
2		=MAIUSC("ciao")	
3			=PI.GRECO()
4			

È possibile creare formule che contengono formule, ovvero annidate.

La lista delle funzioni matematiche riconosciute da Excel e la loro parametrizzazione è già stata fornita qualche pagina addietro.

Organizzare i fogli di lavoro

L'organizzazione dei fogli di lavoro è ciò che rende leggibili e presentabili i dati.

Il progetto Excel è contenuto in un file che si chiama cartella e la cartella a sua volta può contenere più fogli accessibili dal margine in basso a sinistra dello schermo.

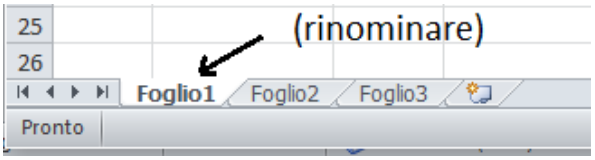
All'interno del foglio la suddivisione è in righe e colonne con intestazioni numeriche sulle righe e alfabetiche per le colonne.

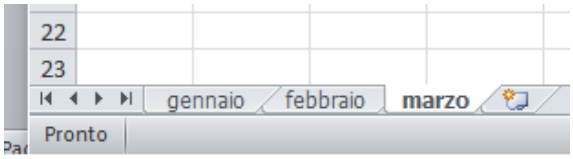
Se una colonna non è abbastanza larga per contenere il dato da mostrare vengono mostrati al posto del dato una serie di hastag "#", per risolvere il problema basta allargare tutta la colonna in questione ysando il mouse.

I fogli invece costituisco più matrici di righe e colonne contenute nella stessa cartella.

È importante organizzare bene i fogli in modo che richiamino subito l'attenzione sul loro contenuto anche se sono ancora chiusi.

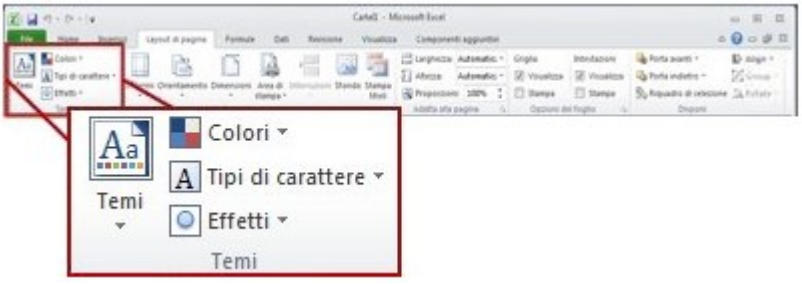
Ad esempio è possibile rinominare i fogli con i mesi dell'anno in cui sia contenuti le stesse tabelle di, ad esempio acquisti mensili.





Per migliorare l'aspetto del foglio di lavoro, al fine di renderlo più presentabile è possibile applicare dei temi scegliendo da una libreria ben fornita.

Nel menù Temi della scheda Layout di pagina fare clic su Temi



Si potranno applicare temi predefiniti oppure personalizzati.

I temi personalizzati sono accessibili solo dopo che sono stati creati. Per creare un nuovo tema bisogna Fare clic su Crea nuovi colori tema, poi nella voce Colori tema fare clic sul pulsante del colore desiderato cliccando su quello che si desidera cambiare.

Analogamente è possibile cambiare i caratteri e gli effetti del tema, definendo un nuovo stile.

Funzioni logiche

Le quattro funzioni logiche di base per Excel sono E, O, XOR e NON e sono applicabili anche in più successioni di celle.

le funzioni logiche di Excel restituiscono VERO o FALSO quando i loro argomenti vengono valutati.

La seguente tabella fornisce una breve sintesi di ciò che fa ogni funzione logica, per aiutarvi a scegliere la formula giusta per le vostre necessità.

Funzione	Descrizione	Formula di esempio	Descrizione della formula
E	Restituisce VERO se tutti gli argomenti vengono valutati come veri.	=E(A2>=20; B2<10)	La formula restituisce VERO se il valore in cella A2 è uguale o maggiore di 20, e se il valore in cella B2 è minore di 10. Restituisce FALSO in caso contrario.
O	Restituisce VERO se uno qualsiasi degli argomenti viene valutato come vero.	=O(A2>=20; B2<10)	La formula restituisce VERO se A2 è uguale o maggiore di 20, oppure se B2 è minore di 10, oppure se entrambe le condizioni sono vere. Se nessuna delle condizioni viene soddisfatta, la formula restituisce FALSO.
XOR	Restituisce un OR esclusivo logico di tutti gli argomenti.	=XOR(A2>=20; B2<10)	La formula restituisce VERO sia se A2 è uguale o maggiore di 20, sia se B2 è minore di 10. Se nessuna delle condizioni è soddisfatta o se sono soddisfatte entrambe le condizioni, la formula restituisce FALSO.
NON	Restituisce il valore logico inverso al suo argomento. Ad es., se l'argomento è FALSO, viene restituito VERO e viceversa.	=NON(A2>=20)	La formula restituisce FALSO se il valore in cella A2 è uguale o maggiore di 20; altrimenti restituisce VERO.

In aggiunta alle quattro funzioni di cui sopra, tra le funzioni logiche Microsoft Excel offre anche le seguenti funzioni condizionali: SE, PIÙ.SE, SWITCH, SE.ERRORE e SE.NON.DISP.

PRESENTAZIONI MULTIMEDIALI

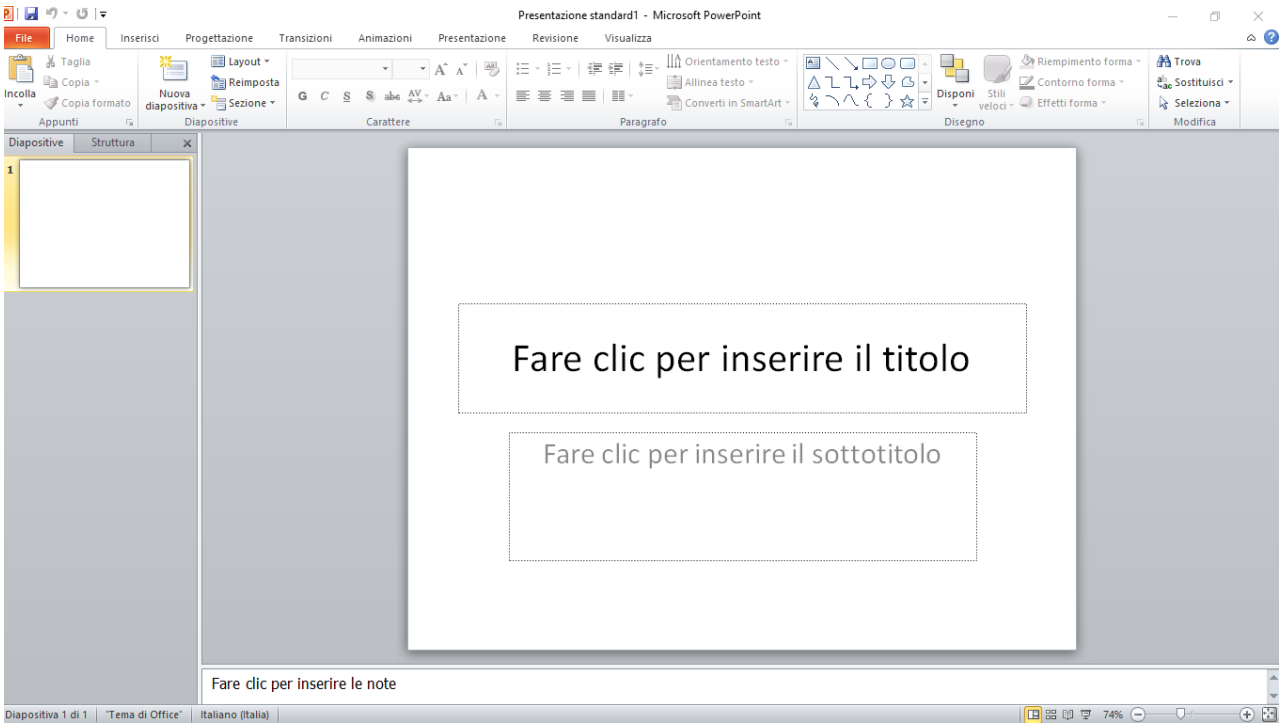
- Utilizzare opportunamente gli elementi della finestra.
- Operare con le diverse visualizzazioni.
- Applicare layout adeguati al contenuto della slide.
- Creare presentazioni ipertestuali e multimediali.
- Inserire collegamenti ipertestuali, suoni e applicare animazioni e transizioni.
- Il software per creare presentazioni .
- La gestione del testo e delle immagini Disegni e schemi.
- Presentare con ipertesti e multimedialità.
- Collegamenti ipertestuali.
- Animare una presentazione.

Le presentazioni

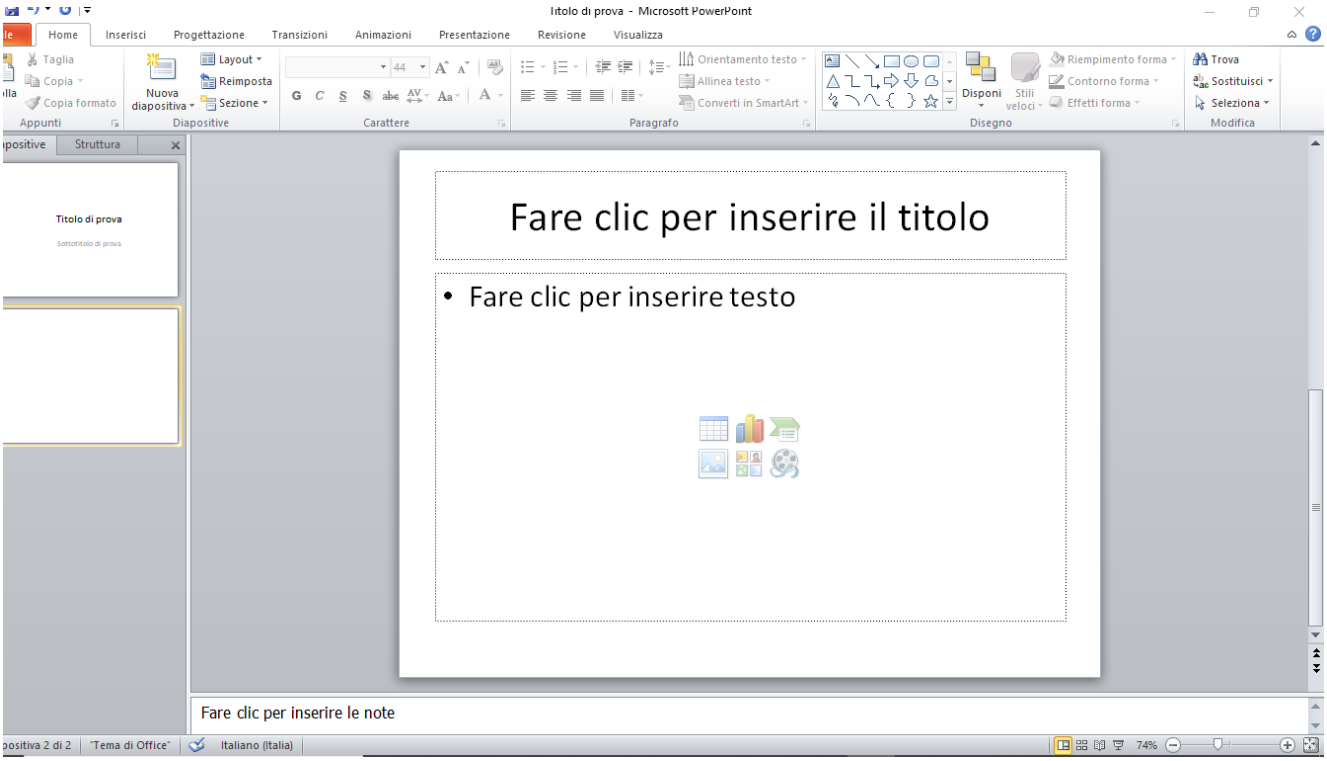
Le presentazioni sono un utile strumento divulgativo e vengono adottate in modo particolare in ambienti didattici o per conferenze professionali o meno.

Uno dei programmi più utilizzati è sicuramente quello proposto da Microsoft, ma ve ne sono altri, la cui funzione comune è quella di usufruire in un modo semplice.

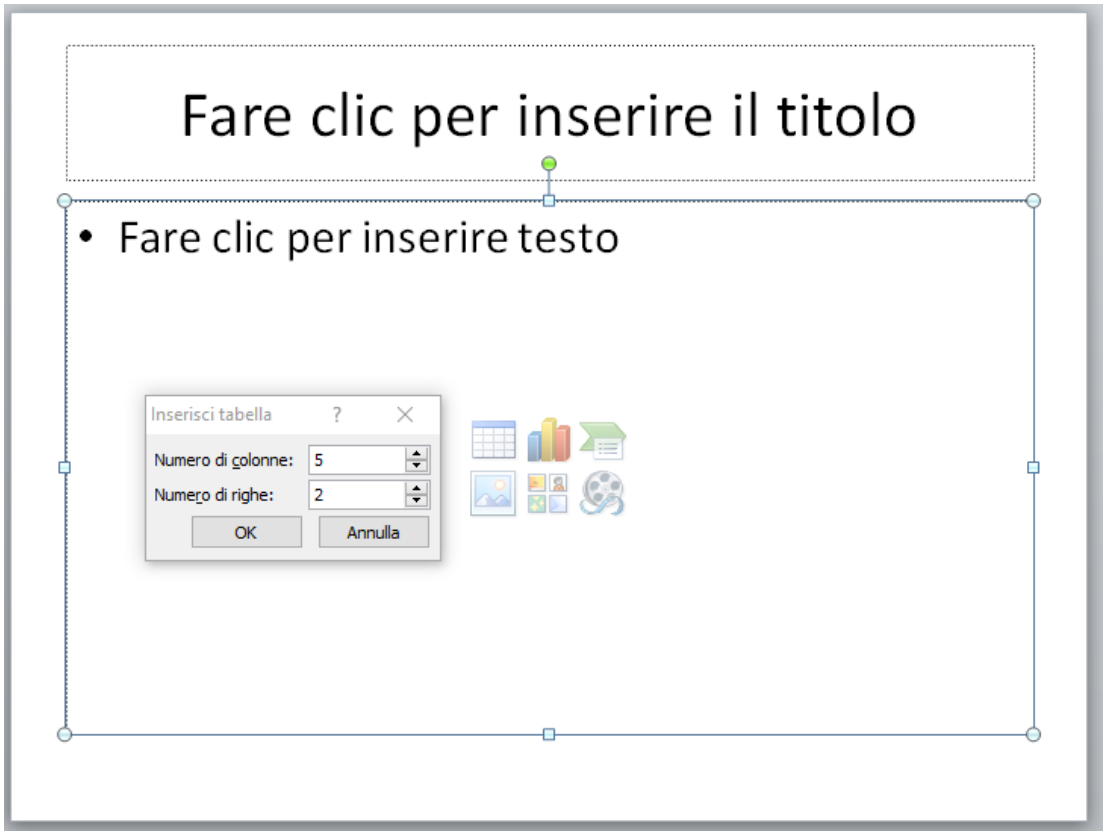
Le presentazioni solitamente sono proiettate su uno schermo con delle schermate definite “slide”, all’apertura, per esempio di “PowerPoint”, viene proposto di inserire il titolo ed il sottotitolo della presentazione.



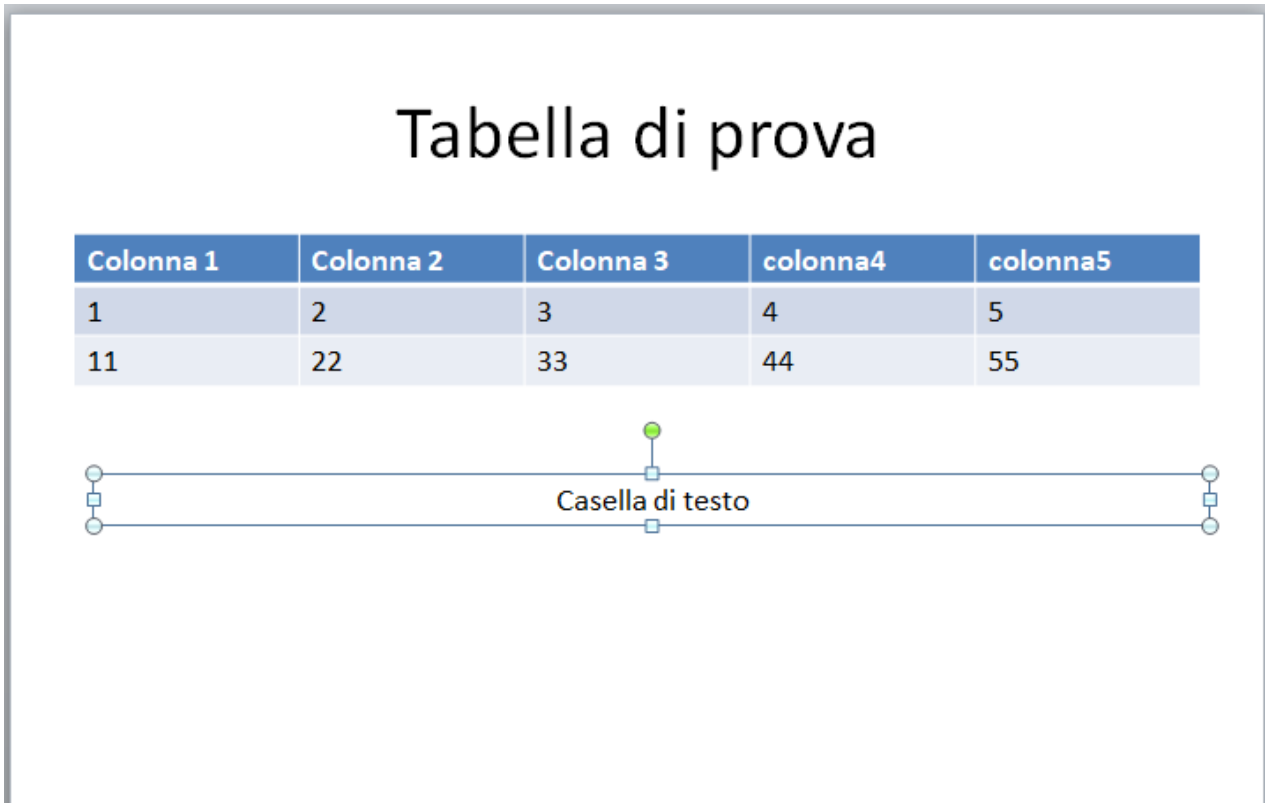
Alla seconda schermata si può inserire il titolo, il testo di base e le varie opzioni inerenti alle esigenze della presentazione, dalle tabelle ai grafici, alle immagini, fino ai link ai videoclip.



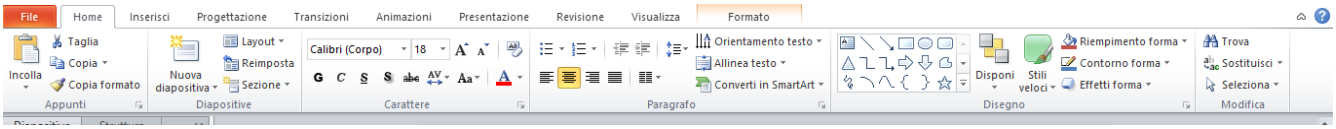
Per esempio, nel caso si volesse **introdurre una tabella**, facendo click sulla icona, si presenta la richiesta del modulo di inserimento per i parametri dimensionali della tabella, come da immagine successiva:



Per poi proseguire nell'inserimento di altri componenti la schermata:

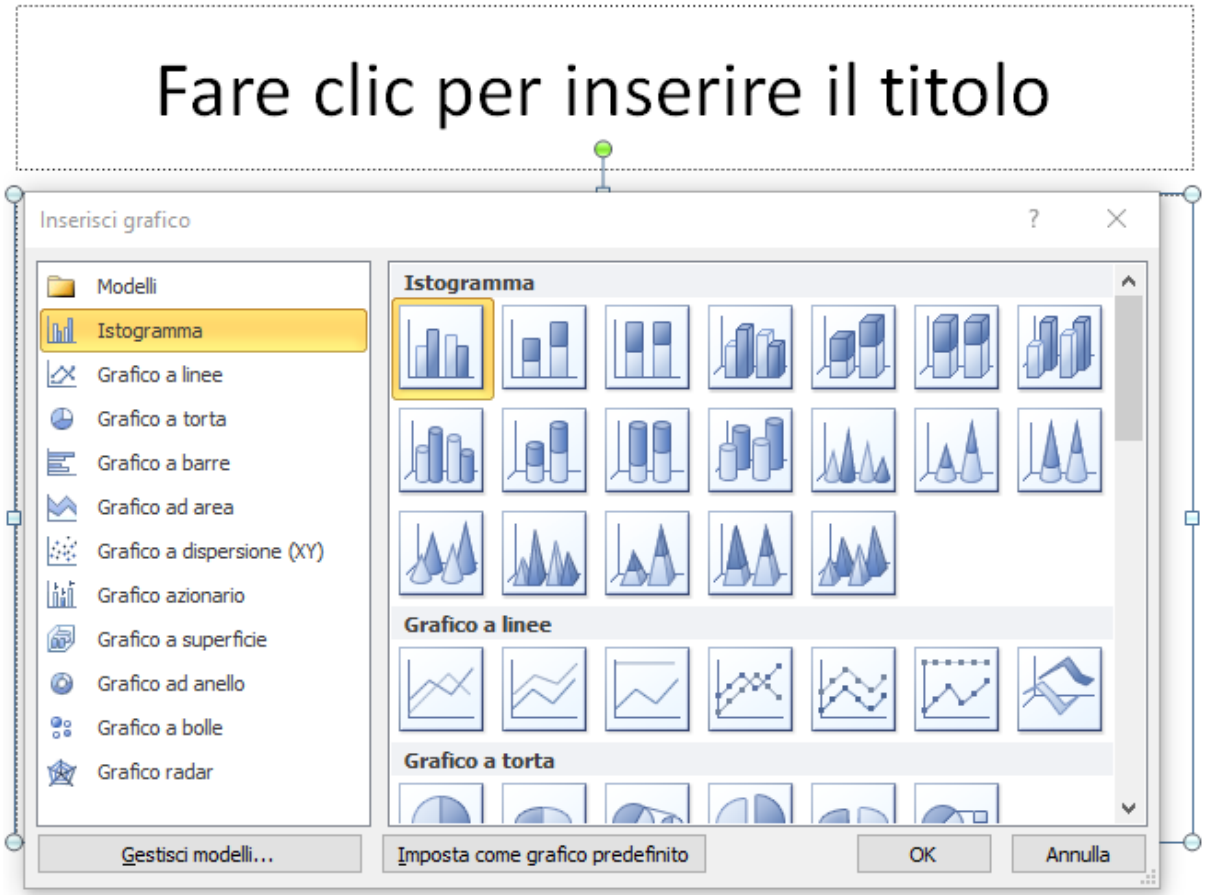


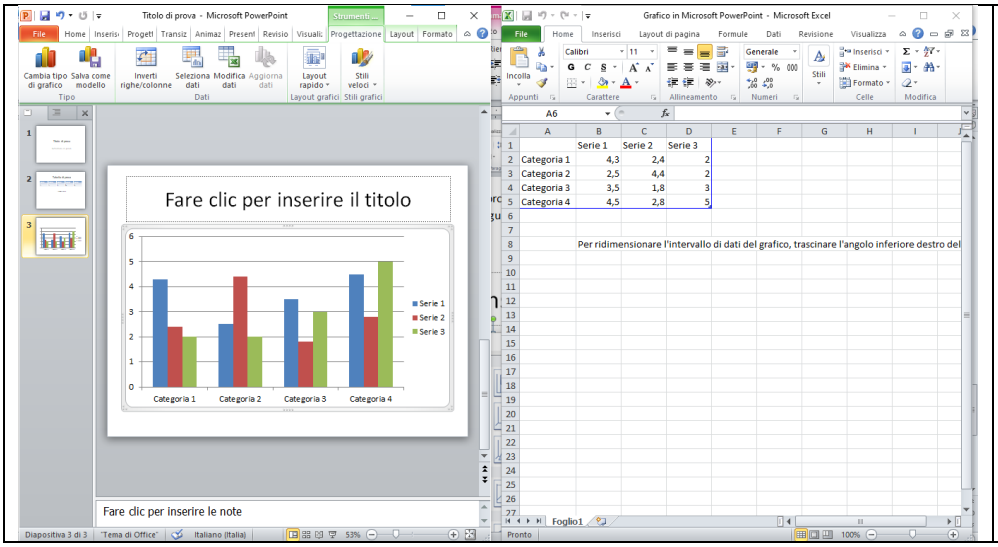
Usufruendo delle varie possibilità di formattazione:



inserire un grafico

Se si volesse **inserire un grafico**, ad esempio, viene proposto un menu per la scelta, come quello indicato in figura:

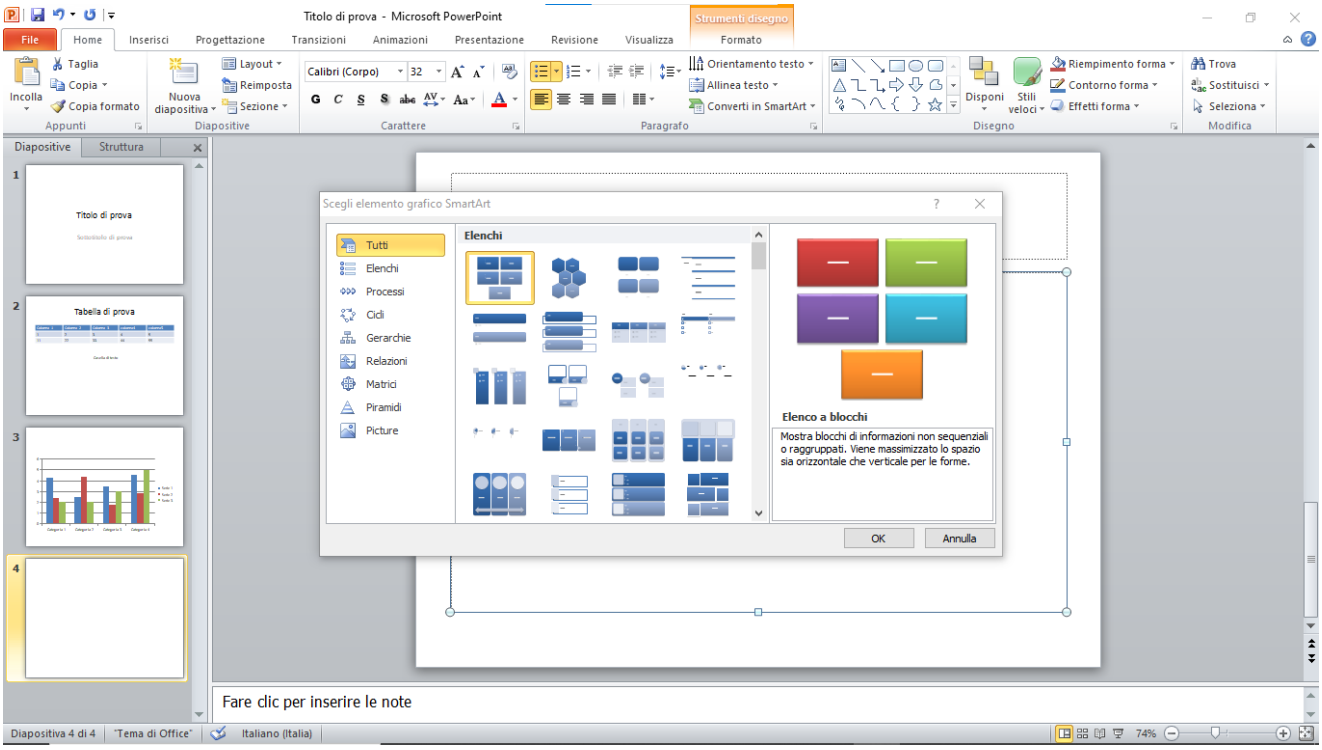




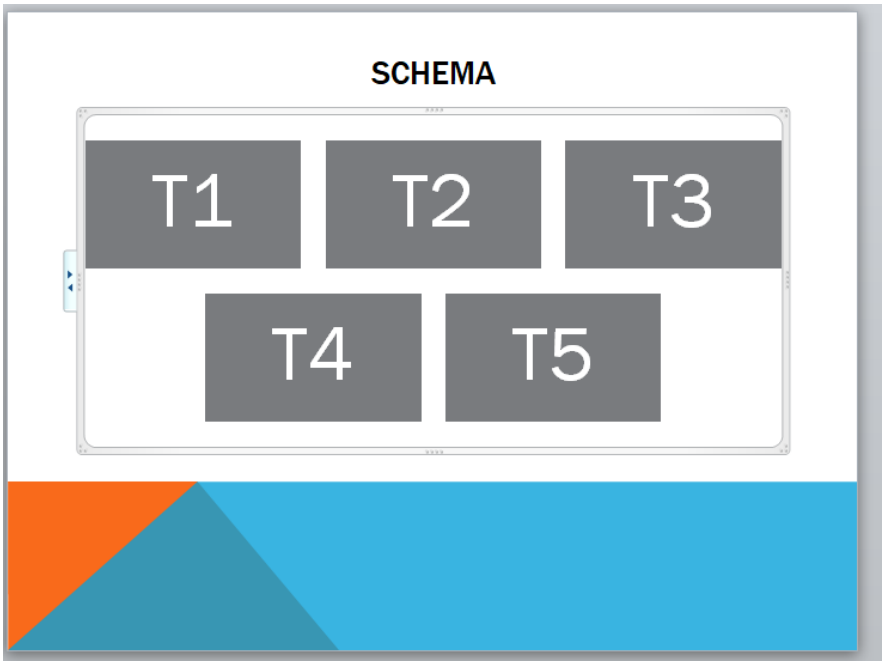
Una volta scelto il tipo di grafico, vi si presenta la schermata relativa all'associazione ad un foglio Excel come in figura a lato.

schema a blocchi

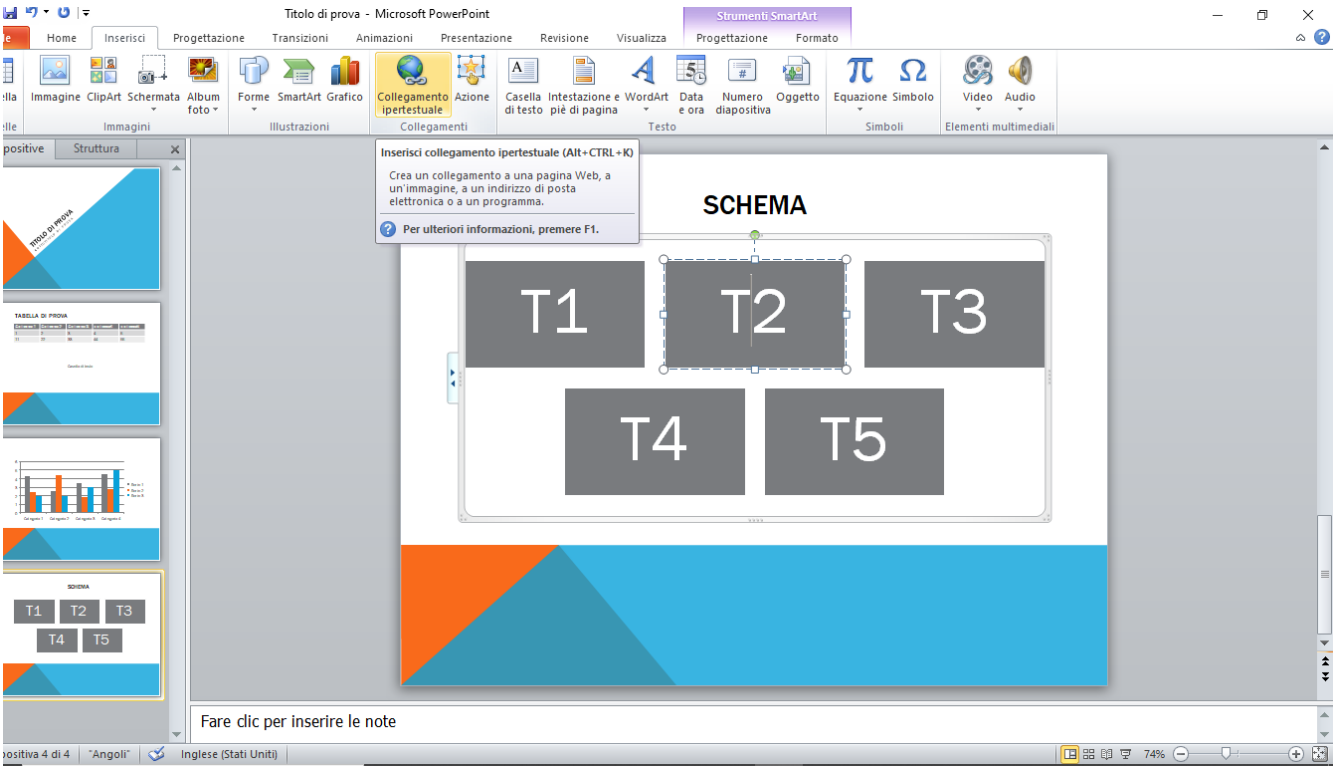
Nel caso si volesse inserire uno schema a blocchi o qualcosa di simile, viene proposto l'inserimento del tipo di "smartArt" da utilizzare:



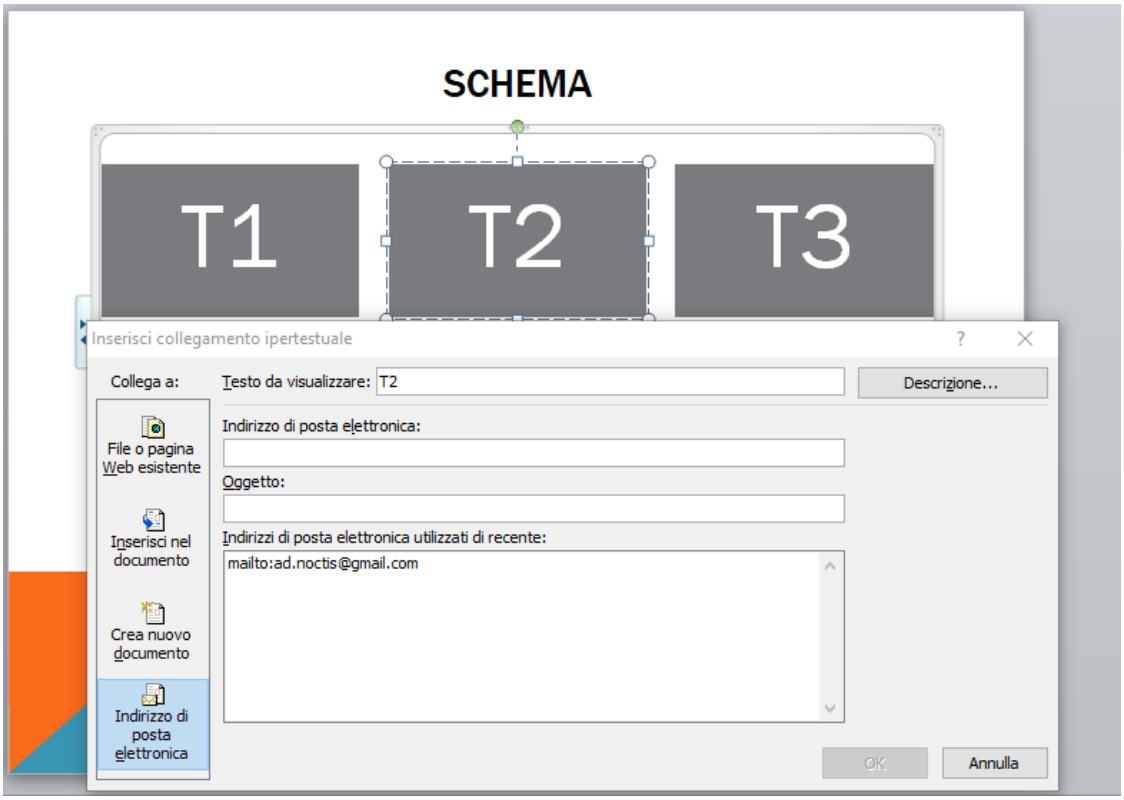
Un esempio di schema:



I collegamenti ipertestuali sono utilizzati per aggiungere ulteriori informazione ai testi:

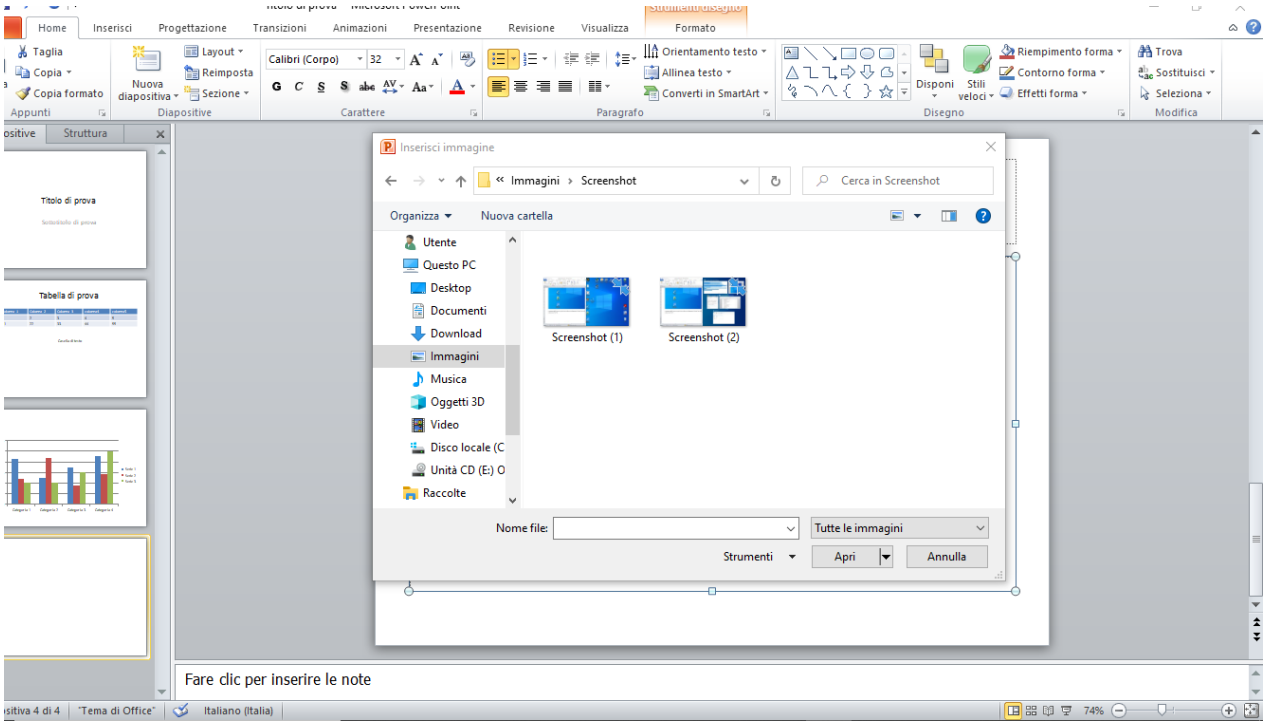


Ad esempio un indirizzo mail associato al testo di una casella di uno schema:

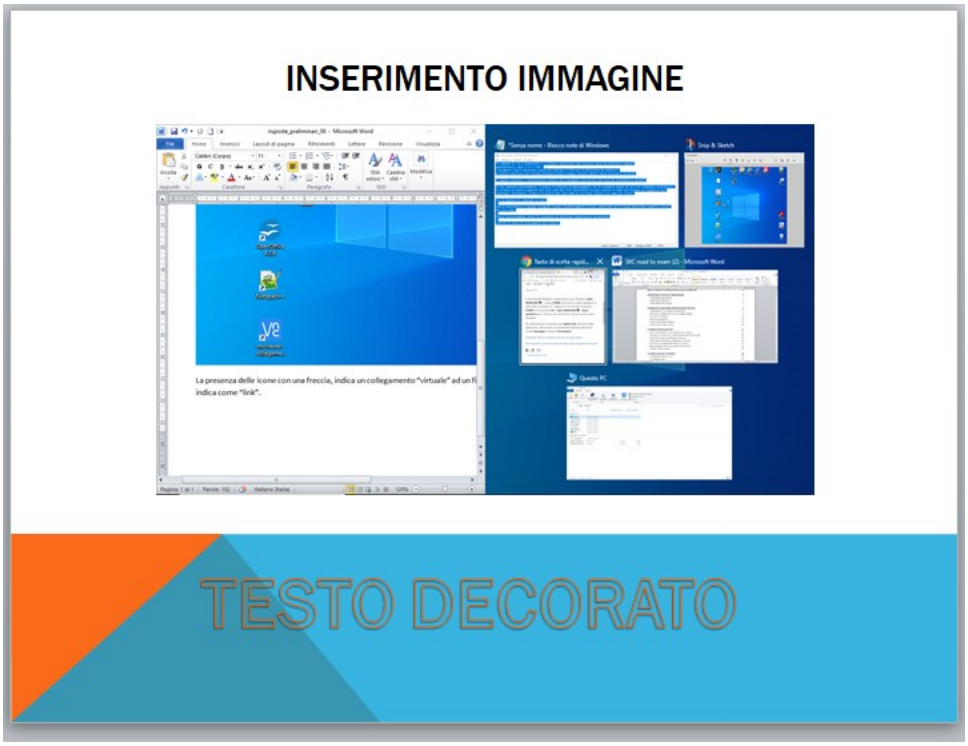


inserire delle immagini

Se invece si volesse inserire delle immagini, viene proposta una schermata analoga:

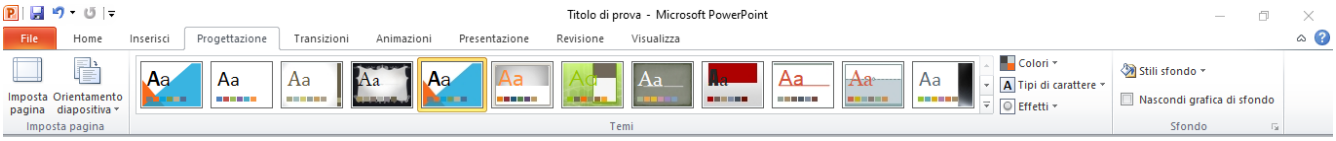


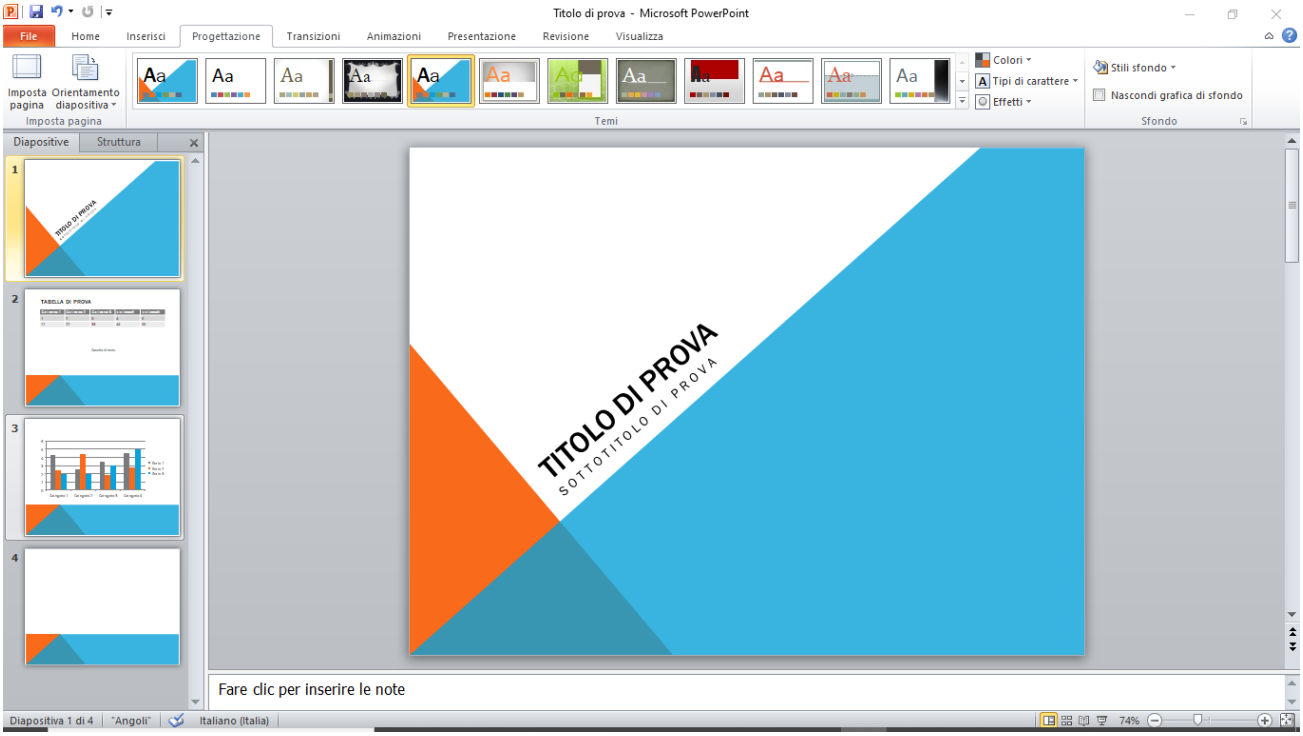
Da cui si può scegliere l'immagine da inserire per avere un risultato simile a questo:



Si possono ovviamente aggiungere altri elementi come ad esempio del testo decorato.

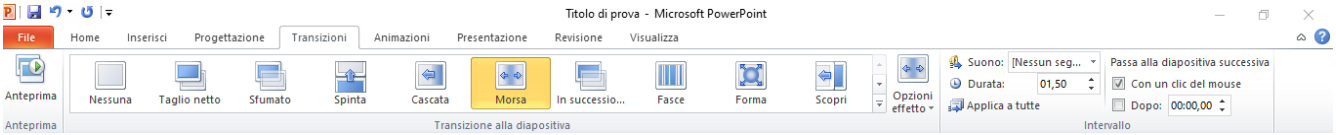
Una volta inseriti i contenuti, si può passare al menu “progettazione” per dare una veste grafica, come illustrato in figura:



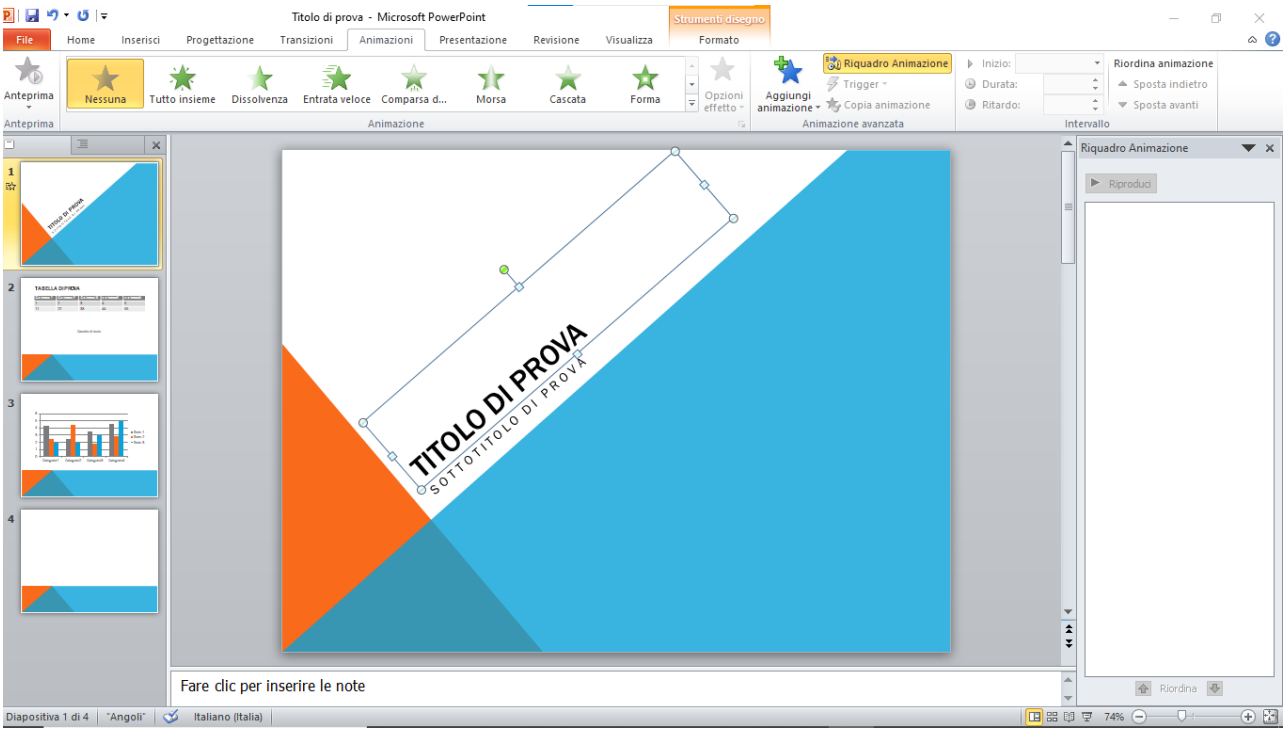


Inserire animazioni e transizioni

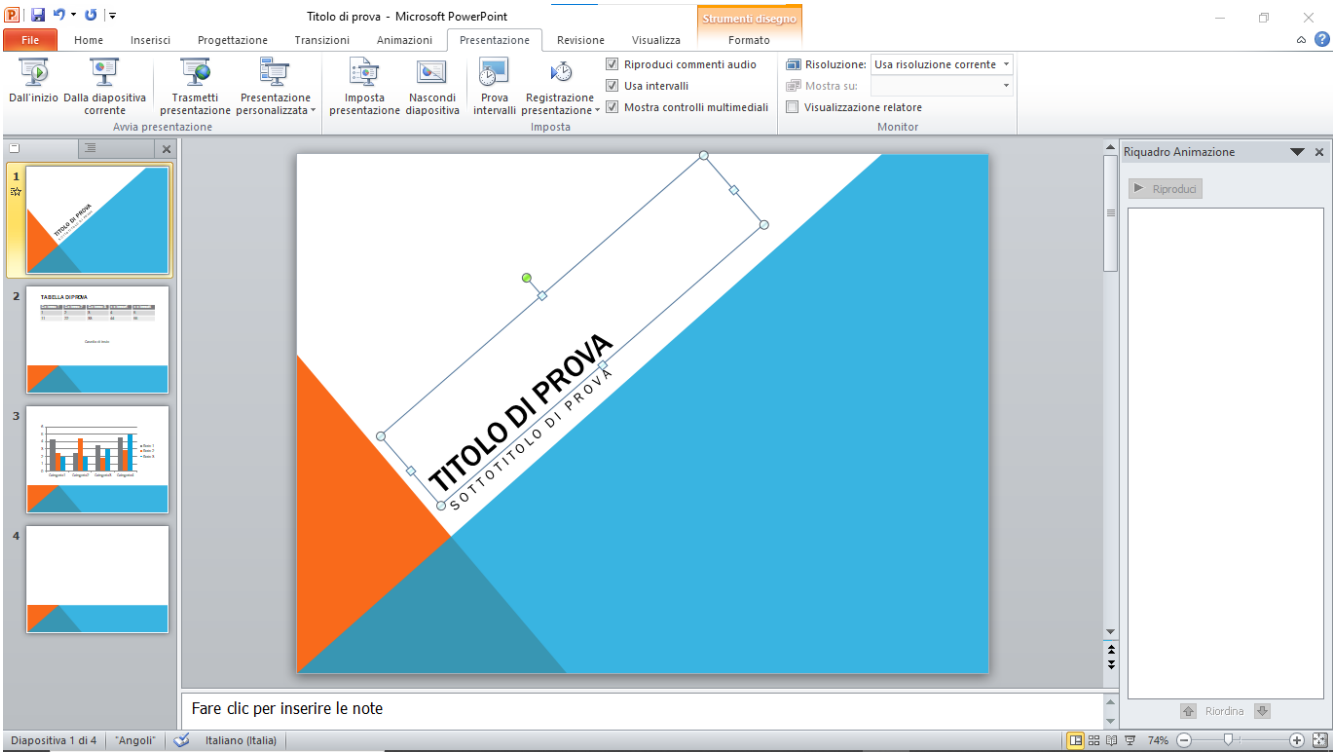
Con il menu “transizioni”, si potrà decidere il metodo in apparizione della immagine:



Con il menu “animazioni”, si possono inserire altri nuovi effetti:



Con il menu “Presentazione”, si potrà quindi visualizzare il risultato finale:



Si raccomanda di non arricchire troppo di immagini e testo, l'interprete principale è chi presenta e non la presentazione.

Slide molto dense e con disegni troppo psichedelici distraggono dal tema della presentazione e mettono in cattiva luce chi presenta.

Infatti potrebbe essere interpretato che il relatore ha bisogno di leggere cosa deve dire piuttosto che avere una traccia rivolta al pubblico.

DAL PROBLEMA ALL'ALGORITMO

- Analizzare, risolvere problemi e codificarne la soluzione con il linguaggio degli algoritmi
- Costruire strategie risolutive non ambigue.
- Azioni e istruzioni.
- Il concetto di algoritmo.
- Rappresentazione degli algoritmi.
- I diagrammi a blocchi.
- Lo pseudolinguaggio.
- Rappresentazione di variabili e costanti.

Analizzare, risolvere problemi e codificarne la soluzione con il linguaggio degli algoritmi

Costruire strategie risolutive non ambigue.

Azioni e istruzioni.

Il concetto di algoritmo.

Rappresentazione degli algoritmi.

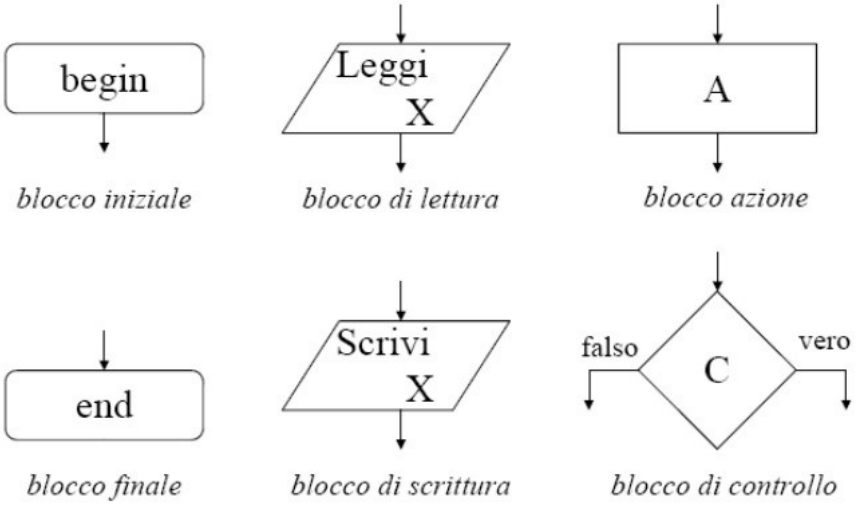
I diagrammi a blocchi

Il diagramma a blocchi è definito anche diagramma di flusso o flow chart ed è una rappresentazione grafica che utilizza delle forme geometriche per descrivere gli algoritmi.

Le forme principali sono gli ovali, i rettangoli , i trapezi, i rombi, sono anche detti blocchi.

Attraverso il diagramma a blocchi (o flow chart) si può indicare l'ordine di esecuzione delle istruzioni. I blocchi fondamentali sono collegati fra loro tramite frecce che indicano il susseguirsi delle istruzioni.

I blocchi elementari sono:



- ciascun blocco azione, lettura/scrittura ha una sola freccia entrante e una sola freccia uscente;
- ciascun blocco di controllo ha una sola freccia entrante e due frecce uscenti;
- ciascuna freccia entra in un blocco o si innesta su un'altra freccia;
- ciascun blocco è raggiungibile dal blocco iniziale;
- il blocco finale è raggiungibile da qualsiasi altro blocco.

Lo pseudo linguaggio

Basiamo la definizione di pseudo linguaggio su alcune parole della lingua italiana, sul simbolo di assegnazione :=, sui normali operatori aritmetici e di confronto.

E' necessario esprimere, prima che l'algoritmo abbia inizio, le variabile su cui andrà ad operare, a tal fine non è neppure necessario definirne il tipo.

Suddividiamo l'esecuzione in passi elementari eseguibili da una macchina senza ambiguità.

Con questo pseudolinguaggio svolgeremo tutte le normali fasi di descrizione di un algoritmo che sono:

- Definizione delle variabili (istruzione VAR);
- Acquisizione dei dati (istruzione ACCETTA);
- Elaborazione (istruzioni varie tra cui l'assegnazione :=);
- Emissione dei risultati (istruzione COMUNICA).

L'inizio dell'algoritmo (dopo la definizione delle variabili) sarà contrassegnato dalla istruzione INIZIO e la fine dalla parola FINE.
Come esempio vediamo il pseudo linguaggio per l'algoritmo per il calcolo del perimetro del quadrato:

```
VAR L,P;
INIZIO
ACCETTA (L);
P := L*4;
COMUNICA (P);
FINE.
```

Rappresentazione di variabili e costanti
Nei pseudo linguaggi le variabili sono descritte in forma di lista di nomi, e i tipi sono lasciati sottointesi, ad esempio:

VAR A,B,Som;

Som:=A+B;

Nei linguaggi veri le variabili hanno specifiche posizioni in cui sono definite e quindi visibili.

Le regole di ambito definiscono la visibilità dai vari punti del programma di ogni singola variabile.

Ogni variabile, allocata in memoria, occupa in realtà due locazioni, dette valore destro e valore sinistro.

Alcuni linguaggi di programmazione utilizzano l'idea di valori L e valori R della stessa variabile, in funzione del modo di valutazione assegnato sul lato sinistro e variabile attuale sul lato destro di un'istruzione di assegnazione.

Un L-value si riferisce a un oggetto che esiste oltre l'utilizzo attuale durante l'esecuzione del programma.

Un R-value è un valore temporaneo che non persiste oltre l'espressione che lo utilizza

In sostanza si può anche dire che il valore contenuto nella variabile è il suo R-Value mentre l'indirizzo che occupa nella memoria è il suo L-Value.

Le variabile definite all'interno di una funzione sono locali a questa funzione e per diventare utilizzabili in un'altra funzione, chiamata da questa, possono essere comunicate tramite il L-Value oppure R-Value, quindi possono essere passate per indirizzo ovvero tramite puntatore o by reference, oppure possono essere passate tramite il loro R-Value quindi in forma di copia del valore, infatti questo metodo è detto by value e al termine della funzione in cui sono stati passati i valori andranno persi ripristinando il valore presente prima di entrare nella funzione.

Riassumendo: nel passaggio by reference viene fornito l'indirizzo in memoria della variabile in cui operare e quindi le variazioni sono permanenti, mentre nel passaggio by value viene passata una copia della variabile e le elaborazioni sono circoscritte a queste copie. Una volta rilasciate dalla memoria le variabili in uso sono ancora disponibili i valore antecedenti all'elaborazione, quindi le variabili originali.

Le variabili possono essere di tipo standard predefinito oppure definito dall'utente, a tal proposito esistono le struct che raggruppano più variabili tra loro non dello stesso tipo in un nuovo tipo caratterizzato da un nome e un elenco di campi.

Quando tra i campi della struct compaiono anche funzioni la nuova variabile definita dall'utente si avvicina alla definizione di classe.

COSTRUIAMO ALGORITMI CON LA PROGRAMMAZIONE STRUTTURATA

Formalizzare la soluzione del problema con le regole della programmazione strutturata

Un diagramma a blocchi di tipo strutturato è più facilmente comprensibile e modificabile infatti in un diagramma strutturato non apparirà mai un'istruzione di salto incondizionato.

Esiste il Teorema di **Böhm-Jacopini** che dice: Ogni diagramma a blocchi non strutturato è sempre trasformabile in un diagramma a blocchi strutturato ad esso equivalente.

Due diagrammi sono equivalenti se, partendo dagli stessi dati iniziali, producono gli stessi risultati.

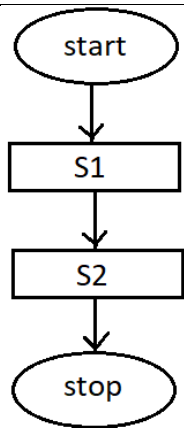
Una descrizione è di tipo strutturato se i blocchi sono collegati tramite i seguenti schemi di flusso strutturato:

- schema di sequenza
- schema di selezione
- schema di iterazione.

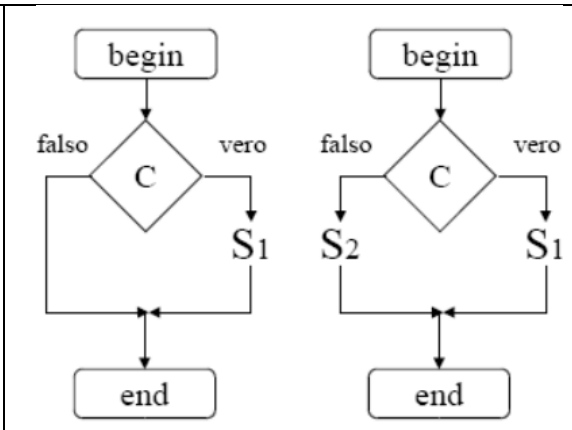
Nello **schema di sequenza** sono presenti i due blocchi ovali di inizio e fine algoritmo, con archi solo discendenti che attivano le azioni di sequenza, ad esempio sequenza 1 e sequenza 2 rispettivamente indicate con S1 e S2.

Nello **schema di selezione**: ν esiste un blocco di controllo che permette di scegliere quale schema di flusso debba essere eseguito tra due schemi, in funzione del valore di verità del controllo.

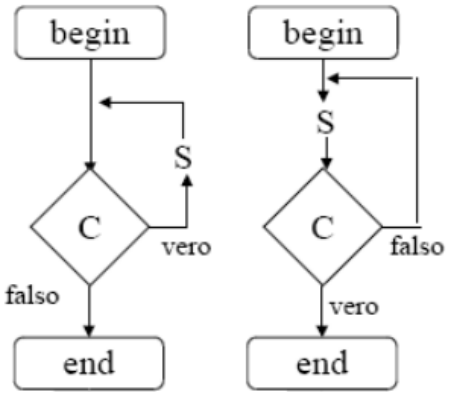
Schema di iterazione (ciclo o loop): è un modo conciso per descrivere azione che devono essere ripetute



Schema sequenza



Schema di selezione



Schema di iterazione

I due schemi di iterazione, mostrati nella tabellina, non sono equivalenti; in un caso lo schema S è eseguito almeno una volta, nell'altro potrebbe non essere mai eseguito. Le condizioni vero/falso per il controllo possono essere invertite: si parla di iterazione per vero quando S è eseguito finché la condizione su C è vera e iterazione per falso nell'altro caso.

Riassumendo i tre casi, mostrati nei diagrammi a blocchi si possono anche chiamare: si possono anche chiamare

- Le strutture fondamentali di un algoritmo
- Le strutture di controllo.
- La sequenza.

La selezione

La selezione si esegue testando con il costrutto if l'espressione che restituisce sempre una decisone di tipo booleano, ovvero TRUE o FALSE.

Prendiamo ad esempi la situazione in cui l'avanzamento del programma o la selezione di un ramo di esso avvenga se una variabile numerica risulti > di un certo valore di soglia, genericamente chiamato setpoint.

Consideriamo due funzioni che hanno lo scopo di muovere un robot avanti o indietro, a seconda del risultato di un calcolo.

La struttura avrà la conformazione software mostrata.

```
void select( ){
void select( ){
int a, b;
    cout<<"inserire un numero maggiore di 10 per spostare avanti il robot";
    cin>>a;
    cout<<"inserire un numero minore di 5 per spostare avanti il robot";
    cin>>b;

    if (a>b) {
            cout<<"Il robot si sposta in avanti";
    }
    else {
    cout<<"Il robot si sposta all\'indietro";
    }
}
```

La selezione multipla

La selezione multipla avviene usando il costrutto **switch – case** il quale prevede più possibilità per il valore acquisito alla variabile di controllo.

Definiamo una variabile di tipo char che ci permetta con il suo valore di selezionare delle strade diverse per l'elaborazione del nostro programma.

Ad esempio, un piccolo Robot, può andare avanti, andare indietro, girare a destra oppure girare a sinistra in funzione delle lettere inserite dalla tastiera del computer.

Queste potranno ad esempio essere:

1. 'A' per andare avanti
2. 'I' per andare indietro
3. 'D' per andare a destra
4. 'S' per andare a sinistra

Definiamo quindi una variabile di tipo char a cui assegniamo il carattere letto, poi, con il costrutto (in inglese si dice statement) switch – case chiamiamo la corretta funzione per muovere le ruote del robot.

```
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

void RobotAvanti(){
	cout<<"il robot si sposta in avanti \n";
}

void RobotIndietro(){
	cout<<"il robot si sposta all indietro \n";
}

void RobotDestra(){
	cout<<"il robot gira a destra \n";
}

void RobotSinistra(){
	cout<<"il robot gira a sinistra \n";
}

void select_move( ){
char select;
cout<<" digita A per avanti, I per indietro, D per destra, S per sinsitra ";
select = getch(); //associamo un carattere che viene dalla tastiera a una variabile
switch (select){
```

```
case 'A':  //caso robot si sposta in avanti
RobotAvanti();
break;
case 'I': //caso robot si sposta all'indietro
RobotIndietro();
break;
case 'D':   //caso robot gira a destra
RobotDestra();
break;
case 'S':  //caso robot gira a sinistra
RobotSinistra();
break;
}
}

int main(int argc, char** argv) {
        select_move( );
        return 0;
}
```

L'iterazione

L'iterazione è una costruzione informatica detta loop, che ripete un gruppo di istruzioni fino a che una certa condizione si avvera.

```
MENTRE ( condizione )

   blocco di istruzioni

FINE-MENTRE
```

L'operazione potrà essere testata prima o dopo l'accesso alla prima iterazione.
L'iterazione "MENTRE" controlla la condizione e successivamente, se la condizione lo permette, esegue il blocco di istruzioni che contiene.

In caso la condizione richiesta fosse già soddisfatta il loop potrebbe non essere eseguito nemmeno una volta (questo accade quando la condizione risulti falsa fin dal primo controllo).

Esempio in C:

```
while ( condizione ) {
   ...;
   ...;
   }
RIPETI
   blocco di istruzioni
FINCHÉ condizione
```

L'iterazione RIPETI termina quando la condizione diventa falsa.

Dato che in questo caso il test della condizione è eseguita dopo aver eseguito il blocco di istruzioni, si esegue sempre almeno una volta il ciclo.

Esempio in linguaggio C e C++:

```
do {
   ...;
   ...;
   }
while ( condizione );
```

```
PER N volte
   blocco di istruzioni
FINE-PER
```

L'iterazione PER è paragonabile al MENTRE ma contiene una condizione implicita che deriva dal fatto che in questo tipo di ciclo il numero di volte che si dovrà eseguire il blocco di istruzioni è noto sin dall'inizio, cosa che in C si realizza con il “costrutto for to do”.

Quasi tutti i linguaggi di programmazione prevedono una variabile intera che funga da contatore di cicli, spesso nominata index, o con nomi simili, che incrementa ad ogni ciclo un numero intero, mentre la condizione solitamente è una determinata variabile sia minore, maggiore o uguale della variabile indice. L’indice si incrementa ad ogni ciclo partendo da un valore iniziale fino ad arrivare al valore finale. Il minimo e il masso sono chiamanti margini e vengono definiti esplicitamente dal programmatore.

Quindi si ottiene:

```
PER variabile = inizio A fine PASSO = passo

   blocco di istruzioni

FINE-PER
```

Anche questo è il pseudo linguaggio del ciclo for.

L’algebra booleana e il suo ruolo nella programmazione strutturata

I test di uscita dai cicli di iterazione rispondono a interrogazioni di tipo Vero o Falso, ovvero la risposta di un test booleano.

Ad esempio se valore_A > valore_B risulterà vero o falso ed quindi un booleano il test di uscita (come anche di entrata dai cicli.

I booleani possono essere stati in OR “||” oppure in AND “&”, come anche il NOT e l’inversione ~

Obiettivi minimi disciplinari da conseguire al termine del primo anno :

- Conoscere la struttura di un elaboratore elettronico
- Conoscere le principali classificazioni dei software.
- Conoscere le funzioni base del sistema operativo dal punto di vista dell'utente.
- Conoscere le funzioni base di un programma di videoscrittura.
- Conoscere le funzioni base di un programma di foglio di calcolo.
- Conoscere le funzioni base di un programma di presentazione multimediale.
- Conoscere la rappresentazione di semplici algoritmi.
- Conoscere i fondamenti della programmazione strutturata.

INFORMATICA - CLASSE SECONDA – OPZIONE SCIENZE APPLICATE

Nota iniziale: Per provare un programma in C o C++ è possibile usare un compilatore online oltre che il classico DevC++.

L'indirizzo del compilatore online è, ad esempio: https://www.onlinegdb.com/online_c++_compiler

Ma ce ne sono molti altri.

I compilatori online possono essere utilizzati anche se si accede con il tablet o il telefono.

CAPACITA' CONOSCENZE DAL PROBLEMA AL PROGRAMMA

1. Analizzare, risolvere problemi e codificarne la soluzione con il linguaggio degli algoritmi
2. Gli obiettivi della programmazione.
3. I linguaggi di programmazione.
4. Il problem solving applicato alla programmazione:
5. analisi di un problema, progettazione di un algoritmo, codifica di un algoritmo.
6. La progettazione di un algoritmo:flow chart
7. pseudocodice Le notazioni in uso negli pseudolinguaggi
8. COSTRUIAMO ALGORITMI CON LA PROGRAMMAZIONE STRUTTURATA
9. Formalizzare la soluzione del problema con le regole della programmazione strutturata
10. Le istruzioni di un algoritmo.
11. Le strutture di controllo.
12. La sequenza.
13. La selezione.
14. L'iterazione.
15. L'algebra booleana e il suo ruolo nella programmazione strutturata.

FONDAMENTI DI TEORIA DEI LINGUAGGI

- Riconoscere le differenze fra linguaggi naturali e linguaggi formali.
- Riconoscere le caratteristiche di un linguaggio di programmazione.
- Cosa è un paradigma di programmazione.
- Come lavorano i compilatori e gli interpreti.
- Il software.
- Linguaggi naturali e linguaggi formali.
- Linguaggi di programmazione a basso livello.
- Linguaggi di programmazione ad alto livello.
- I paradigmi di programmazione.
- I programmi traduttori: compilatori e interpreti.

LINGUAGGI DI PROGRAMMAZIONE

- Comprendere le differenze di funzione e di stile inerenti i vari linguaggi di programmazione.
- Risolvere problemi con l'ausilio delle strutture astratte dei dati.

Codificare algoritmi in un linguaggio di programmazione.

Per programmare in un qualsiasi linguaggio di sviluppo è necessario procurarsi la lista dei costrutti, in inglese detti statements e la lista delle librerie che questo dispone.

Le librerie, in molti linguaggi, sono contenute in moduli esterni, detti header file, che possono essere inclusi nel file principale, con un comando del tipo #include <nomefile>

Questo comporta una macro espansione del codice sorgente, al fine di creare un unico file che viene con un passaggio successivo assemblato.

La codifica di algoritmi richiede l'utilizzo di comandi di selezione IF, comandi di sequenza FOR e comandi di ciclo WHILE.

L'algoritmo, rappresentato con un diagramma di flusso, il flowchart, deve avere un inizio e una fine eventualmente forzabile se si tratta di un loop infinito.

Durante l'esecuzione dei cicli l'algoritmo può chiedere delle variabili in input e restituire dei dati in output.

L'algoritmo è più buono, quando converge verso la soluzione con un numero minimo di passi, questa situazione si chiama complessità computazionale.

Quelle più scarse sono le complessità lineari in cui il numero di passi da eseguire crese in maniera lineare con la grandezza dei dati in input rendendo l'algoritmo inutilizzabile quando la grandezza è molto ampia.

Programmiamo in C

Il linguaggio C è uno dei piu potenti anche se datato perché risale agli anni 80.

E' stato sviluppato da Kernigham e Ritchie, due ricercatori ri recente scomparsi.

Il linguaggio è di tipo funzionale e modulare, espandibile grazie alla tecnica della dichiarazione di tipi di dato astratto definito dall'utente, come iterazione dei tipi di dato predefiniti.

Modulare significa che si compone di più file separati, di due tipi, con estensione .h e estensione dove i primi contengono la dichiarazione e la struttura delle funzioni, ovvero i prototipi (header file, o file di intestazione), mentre i file sorgenti C contengono l'utilizzo di queste.

L'evoluzione del C in C++ consiste nel potenziare le strutture di dati, intese come insieme di dati tra loro non omogeni identificati da un unico nome di accesso (una variabile di variabili) inserendo tra i campi anche delle funzioni dette metodi.

Le struct o strutture, così modificate assumono il nome di “classi” e le funzioni in esse contenute si chiamano metodi della classe.

La classe ha la caratteristica di poter ereditare le funzioni di una classe in essa contenuta, di modificare la forma dell’oggetto in uso rendendolo polimorfo, e di proteggere i dati dall’accesso da parte di altre funzioni o classi non autorizzate che si chiamo incapsulamento dei dati.

Ne consegue che: Ereditarietà, Polimorfismo, Incapsulamento, sono le caratteristiche fondamentali della OOP ovvero della programmazione orientata agli oggetti.

Tutti i compilatori detti Visual sono di tipo orientato agli oggetti, quindi manipolano e creano classi.

Quando si programma in C si deve quindi definire a priori se stiamo usando un compilatore ANSI C o un compilatore Visual o Cpp perché la programmazione benché simile è diversa grazie alla presenza delle classi nella versioni C++.

Nel caso di programma ANSI C, predisponiamo creando dei moduli (file esterni) con estensione del nome .h

In questi header files metteremo la struttura delle funzioni, con la loro parametrizzazione effettuabile alle funzioni che la chiamano.

Ad esempio possiamo creare un file settaggi.h in cui all’interno memorizziamo i comandi che inizializzano il microcontrollore che pilota il robot oppure delle funzioni che servono all’esecuzione del programma nel PC.

Il primo ambito si chiama programmazione embedded, in cui il programma prodotto verrà compilato e nella sua versione esadecimale scaricato nella memoria del Microcontrollore che è il cervello del robot.

Nel linguaggio C di base, ovvero ANSI C, ci sono costrutti che saranno accettati anche nel linguaggio C++ che li eredita.

In primo luogo bisogna sapere come e dove dichiarare le variabili.

Le variabili possono essere di tipo bit quando serve un singolo booleano per prendere una decisione, oppure char -> carattere, se devo ricevere o inviare un numero o simbolo contenuto nella tabella ASCII (America Standard Code for Information Interchange), tramite un protocollo seriale oppure WiFi, ecc.

Posso definire i numeri interi a 16 bit senza virgola, che possono essere positivi o negativi nel range +32767 fino a -32768.

Se devo usare numeri interi piu grandi li definisco di tipo long int, ovvero intero a 32 bit invece che a 16.

I numeri con la virgola si definisco come float e sono espressi secondo il formato IEEE con il primo bit a sinistra che esprime il segno 0 positivo, 1 negativo, segue un byte che rappresenta l’esponente del numero normalizzato con la prima cifra a destra della virgola diversa da zero, mentre i successivi 23 bit rappresentano la mantissa, allineata a destra.

Esiste il tipo struct che permette di racchiudere più variabili di tipo diverso sotto un unico nome identificativo, e queste variabili si chiamano campi della struct.

Le variabili definite fuori da ogni funzione, ovvero dopo gli include, sono dette globali e valgono per ogni posizione del programma.

Sono poco sicure e sconsigliate, il programmatore esperto non le usa.

Le variabili definite dentro alle funzioni o le void sono dette locali e valgono solo per il periodo in cui la funzione è attiva.

Per trasmettere variabili e valori delle variabili da una funzione all'altra esistono due maniere, il metodo per copia e il metodo per indirizzo.

Il metodo per copia opera su duplicati che vengono rilasciati alla fine dell'esecuzione della funzione quindi i valori originali non sono cambiati, mentre i passaggi per indirizzo, che necessitano dell'uso dei puntatori modificano in maniera permanente il valore della variabile.

Si possono definire anche Array come variabile contenete un numero preciso e predefinito di altre variabili uguali, e si possono definire array di array quindi delle matrici.

Esistono un gran numero di librerie e di file header già inclusi che permetto molte applicazioni, ad esempio #include <math.h> permette di attivare le funzioni matematiche e trigonometriche più comuni.

La lista degli headers files è interrogabile entrando nella cartella include dell'installazione del compilatore fatta dal sistema operativo.

Il costrutto sequenza

La sequenza in C o C++ si può ottenere sfruttando i cicli del main program. Esistono dei comandi che permettono di incrementare o decrementare dei valori numerici o non numerici usando il comando variabile ++ oppure ++variabile. Analogamente variabile- - oppure - - variabile.

Vediamo un esempio in cui la variabile viene incrementata fino a che non si raggiunge un valore di soglia.

Per provare un programma in C o C++ è possibile usare un compilatore online oltre che il classico DevC++.

L'indirizzo del compilatore online è, ad esempio: https://www.onlinegdb.com/online_c++_compiler

Ma ce ne sono molti altri.

È possibile usare il compilatore online anche da dispositivi mobile, ad esempio, telefoni o tablet.

```
/*****************************************
Prof Marco Gottardo Liceo SIIC 2020/2021
Lezione di implementazione delle sequenze in C++
Il computer esegue l'incremento di una variabile
se viene premuto il tasto M oppure il decremento
se ne viene premuto il tasto L.
La sequenza è composta da tre passi sia in incremento
```

```
Che in decremento
*****************************************/
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

int main()
{
   char step;
   int value=0;
   cout<<"press M for more and L for less";
   cin>>step;
   if (step=='M') {
      value++; //incrementa il valore
      cout<<"primo passo "<<value<<"\n";
   }
   if (step=='L') {
      value--; //decrementa il valore
      cout<<"primo passo "<<value<<"\n";;
   }
   // esegue il secondo passo delle sequenza
      if (step=='M') {
      value++; //incrementa il valore
      cout<<"secondo passo "<<value<<"\n";;
   }
   if (step=='L') {
      value--; //decrementa il valore
      cout<<"secondo passo "<<value<<"\n";;
   }
   // esegue il terzo passo
         if (step=='M') {
      value++; //incrementa il valore
      cout<<"terzo passo "<<value<<"\n";;
   }
   if (step=='L') {
      value--; //decrementa il valore
      cout<<"terzo passo "<<value<<"\n";;
   }
   cout<<"Fine della sequenza: Bye bye";
   return 0;
}
```

Questo programma quando eseguito chiede all'utente se il valore iniziale, in questo caso 0, viene aumentato o diminuito per tre volte.
La lunghezza della sequenza è facilmente modificabile.

Scriviamo un programma

Un programma in C o in Cpp come in qualunque altro linguaggio deve seguire la regola dello sviluppo TOP DOWN, ovvero si cerca di suddividerlo in questioni più semplici da risolvere alle quali assegniamo della funzioni compatte e brevi.

A queste funzioni si passano i parametri (numeri e risultati) delle funzioni precedenti che hanno ricavato risultati parziali.

La struttura di un programma in C++ inizia con l'inclusione delle librerie, tramite la direttiva al preprocessore #include.

Si possono includere file di libreria usando il costrutto < > oppure file costruiti dall'utente tramite il costrutto " ".

Il file che citiamo tra parentesi angolari o apici verrà aperto e il contenuto sostituirà la riga in cui è chiamato.

Si definisco le funzioni, nello stesso file oppure in file esterni, e queste potranno restituire valori oppure no.

Se non restituiscono valori sono definite di tipo void ovvero vuoto, non dovremmo quindi aspettare nessun valore nel nome della funzione.

Se invece sono definite di un tipo all'ultima riga della funzione comparirà il costrutto return seguita dal valore o variabile che vogliamo fare trovare nel nome della funzione del tipo dichiarato. Ecco un esempio per la funzione somma di due numeri.

```
#include <iostream>
#include <math.h>
using namespace std;

int somma(int a, int b){
        int result;
        result = a+b;
return result;
}

int main( ){
        int calcola;
        calcola=somma(3,4);
        cout<<calcola;
return 0;
}
```

Dichiarazione di variabili

Dichiarare una variabile significa riservare a questa un'area di memoria della misura adeguata al tipo di cui essa è definita.

L'assegnare un indirizzo di memoria dal punto di vista informatica assume il nome di allocazione della memoria.

La memoria può essere allocata in maniera statica o dinamica.

Le variabili statiche sono quelle che richiedono subito la dimensione e questa risulta occupata anche se la variabile non viene utilizzata all'interno dell'algoritmo, mentre le variabili ad allocazione dinamica possono cambiare dimensione in memoria perché composte da elementi uguali allocati nel momento del bisogno ovvero runtime.

Esempi di variabili di tipo statico sono tutti i pipi principali e predefiniti, ad esempio la variabile bit, di tipo bool, che occupa una sola cella di memoria, mentre la char, che può contenere un intero corto, compreso tra 0 e 255, oppure da -128 fino a + 127 nel caso fosse definita con segno, occupa 1 byte ovvero 8 bit.

Ci sono poi gli interi standard con segno a 16 bit, che possono valere da -32768 a più 32767.

Anche gli interi a 16 bit possono essere dichiarati di tipo unsigned ovvero solo positivi in modo da poter arrivare a oltre 65mila valori possibili.

Si passa poi alle variabili a 32 bit che possono raggiungere valori molto grandi come anche molto piccoli, se definiti float con lo standard IEEE detto anche modalità esponenziale scientifica.

La definizione delle variabili è soggetta, in qualunque linguaggio, alle regole di ambito.

Le regole di ambito definiscono dove le variabili sono leggibili/scrivibili e dove invece risultino non accessibili.

Esistono delle specifiche posizioni all'interno del codice sorgente che definisce se la variabile è oppure non è di tipo globale.

Le variabili definite fuori da ogni funzione sono di tipo globale e di solito non danno origine a programmi ben ingegnerizzati in quanti la regola di base è quella di mascherare alle altre funzioni l'utilizzo della variabile in altre funzioni.

Ad esempio una variabile definita nella funzione void calcola( ){} non è visibile nella altre funzioni compreso il main quindi esiste un meccanismo di passaggio dei parametri tra funzioni che permette di utilizzare i valori allocati in due possibili maniere:

1. Passaggio delle variabili by value, che crea un copia della variabile all'interno della funzione, allocando nuovo spazio che verrà rilasciato, con conseguente perdita del valore contenuto, all'uscita dalla funzione.
2. Passaggio delle variabili by reference, ovvero per indirizzo, che fornisce alla funzione locale i puntatori di accesso alle locazioni di memoria in cui il dato è alloggiato. Ne consegue che ogni

modifica effettuata sulla variabile all'interno della funzione attuale risulta permanente e disponibile anche dopo l'uscita dalla stessa funzione.

Consideriamo ad esempio il linguaggio C oppure C++, le variabili si dichiarano definendo prima il tipo e poi il nome.

È anche possibile inizializzare le variabili all'atto della dichiarazione, ad esempio:

int addendo; //dichiara e alloca un variabile di tipo intero a 16 bit con nome addendo.

Se invece pondo dopo la dichiarazione il costrutto = posso associargli un valore detto valore iniziale.

int addendo=0; //dichiara l'intero addendo a 16 bit e lo pone uguale a zero.

Per allocare una variabile nella funzione principale che poi chiama le altre passando il valore per indirizzo, passaggio by value, va definita con il costrutto **&**, mentre per prendere questo indirizzo e poter usare il valore all'interno della funzione ricevente si usa il costrutto *****.

Ad esempio:

```
#include <iostream>  //codice testato da prof. Gottardo Marco alle lezione del 29 marzo 2021 in 2à Liceo SIIC

void swap(int *a, int *b){
	int temp;
	temp =*a;
	*a=*b; //il valore puntato da a assume il valore puntato da b
	*b=temp; //il valore puntato da b viene posto uguale allo swap
}

int main(){
	int a,b;
	a=3;
	b=5;
	swap(&a,&b);
	std::cout<<"a = "<<a<<" b = "<<b;
return 0;
}
```

Tipi di dati

I tipi di dato fondamentali del linguaggio C e C++ sono divisi in tre categorie:

1. Interi
2. Reali
3. Caratteri

Gli interi sono i numeri senza la virgola, e possono essere definiti a 8,16,32 bit e con segno oppure senza segno.

La disposizione dei bit all'interno della dimensione disponibile definisce il formato, in effetti 16 bit ovvero una word, definiscono un formato e quindi il valore che il numero assume.

Ecco un programma che estrae i valori minimi e massimi dai tipi di dati interi fondamentali.

```
#include <iostream>
#include <limits.h>

using namespace std;

int main(int argc, char** argv) {
        cout<<"Prof. Marco Gottardo Ph.D. stampa dei margini dei tipi di dato\n\n";
        cout<<"il minimo char a 8 bit e\' "<<CHAR_MIN<<"il massimo e\' "<<CHAR_MAX<<"\n";
        cout<<"il minimo intero a 32 bit e\' "<<INT_MIN<<"il massimo e\' "<<INT_MAX<<"\n";
        cout<<"il minimo intero a 16 bit e\' "<<SHRT_MIN <<" Il massimo e\' "<<SHRT_MAX<<"\n";
        cout<<"il minimo intero senza segno a 32 bit e\' "<<0<<" Il massimo e\' "<<UINT_MAX<<"\n";
        cout<<"il minimo intero corto senza segno a 16 bit e\' "<<0<<" Il massimo e\' "<<USHRT_MAX<<"\n";
        cout<<"il minimo intero lungo senze segno a 32 bit e\' "<<0<<" Il massimo e\' "<<ULONG_MAX<<"\n";
        return 0;
}
```

È possibile convertire i tipi di dato anche in maniera automatica, con o senza perdita di informazione o con estensione del possibile valore.
Le conversioni di questo tipo si chiamano cast.
Ecco un esempio di conversione da tipi alfanumerici a numerici e viceversa.

Conversioni di tipo esplicite

Per eseguire le conversioni esplicite tra tipi scalari, al fine di cambiarne il tipo si puo effettuare il cast, a questo scopo si usa la sintassi: (nuovotipo)espressione
Da stringa a numero

```
gets(line) ;
x = atoi(line) ; /* int */
x = atol(line) ; /* long */
x = atof(line) ; /* float o double */
```

- Assegnazione di valori alle variabili.

- Il costrutto selezione.
- Selezione semplice e doppia,
- selezione con blocchi di istruzioni.
- Il costrutto selezione multipla.
- L'algebra della logica proposizionale.
- I connettivi logici: AND, OR, NOT.
- Il costrutto iterazione.
- Iterazione con controllo in testa.
- Iterazione con controllo in coda.
- Iterazione a conteggio.

TRASMISSIONE DEI DATI E RETI DI COMUNICAZIONE

- Comprendere cosa significa comunicare e quali sono le forme di comunicazione.
- Comprendere cosa sono le reti di computer.
- Comprendere le differenze fra i vari tipi di rete.
- Riconoscere i dispositivi che servono per realizzare una rete.
- La comunicazione.
- Gli elementi della comunicazione.
- La telematica.
- Reti di computer
- Tipi di rete.
- Segnali analogici e digitali.
- Mezzi trasmissivi
- L'architettura client/server.
- Le topologie di rete.
- Estensione di una rete.
- Reti analogiche e reti digitali
- Le apparecchiature per creare una semplice LAN
- La componente software della trasmissione:
- i protocolli.
- Il protocollo TCP/IP

INTERNET : UNA RETE UNIVERSALE

- Comprendere la ‘rivoluzione digitale’.
- Cosa è Internet, come funziona e cosa serve per connettersi.
- Come si naviga in Internet.
- Le origini di Internet.
- Internet e WWW.
- Il protocollo applicativo HTTP.
- La sintassi URL.
- Le funzioni della posta elettronica.
- Comprendere e utilizzare i principali servizi di
- Internet.
- Gli indirizzi IP
- I browser.
- La connessione a Internet.
- Modalità di collegamento.
- Internet computing mobile.
- La ricerca delle informazioni.
- La posta elettronica.
- Le applicazioni di Internet:
- E-commerce e home – banking.
- Il telelavoro.

Obiettivi minimi disciplinari da conseguire al termine del secondo anno :

- Conoscere le tecniche di analisi e risoluzione dei problemi indipendentemente
- dal linguaggio di programmazione.
- Conoscere la sintassi di un linguaggio di programmazione.
- Saper sviluppare semplici programmi che utilizzano dati elementari.
- Conoscere le funzioni base di una rete di computer.
- Conoscere i vari elementi che compongono una rete.
- Conoscere le funzioni base di Internet.
- Conoscere le applicazioni più importanti di Internet.

VALUTAZIONI E VERIFICHE - INFORMATICA

Per la valutazione complessiva di ciascun periodo è previsto un voto unico anche in conformità con la c.m. n° 89 del 18 /10/2012. Per elaborare il giudizio finale e quello intermedio si terrà conto dei risultati conseguiti nelle prove di verifica sia orali che scritte.

Il Dipartimento disciplinare delibera che il numero minimo di verifiche necessarie per elaborare la valutazione è di : **4**

due prove per ciascun periodo di cui almeno una per lo scritto Le verifiche scritte possono essere svolte su carta oppure al computer; in questo ultimo caso verrà predisposta la relativa stampa oppure le verifiche saranno memorizzate in formato digitale su supporto informatico.

Come verifiche orali, oltre alle interrogazioni, possono essere valutati anche lavori svolti al computer, esercitazioni da posto o alla lavagna, brevi interventi individuali.

In particolare, per il numero esiguo di ore settimanali, le verifiche potranno essere proposte in sola forma scritta, con la eventuale riserva di sottoporre a ulteriore verifica orale gli alunni che mostreranno evidenti difficoltà.

Per la valutazione si tiene conto :

- delle conoscenze possedute;
- delle abilità acquisite nelle applicazioni, anche in situazioni nuove;
- della qualità dell'esposizione;
- dell'impegno e della partecipazione al dialogo educativo
- del progresso rispetto alla situazione di partenza.

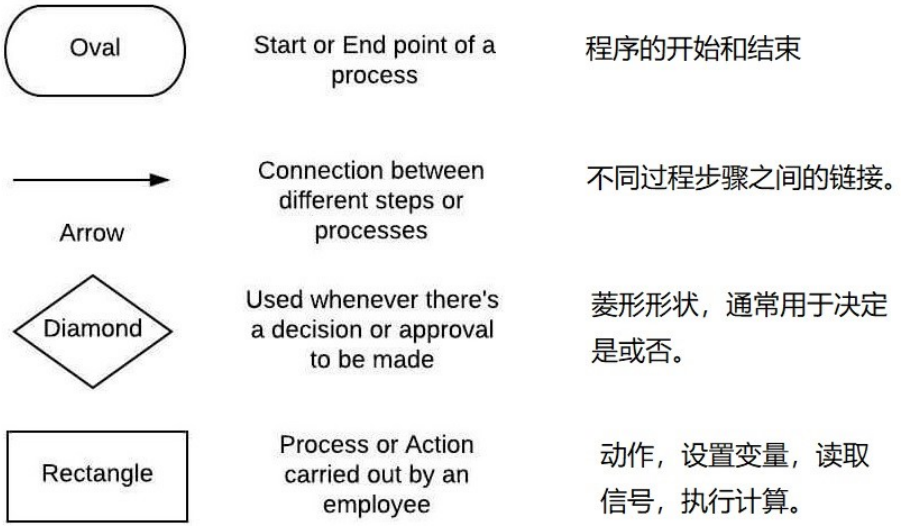
Seconda liceo 2021. Preparazione all'esame finale. Prof. Marco Gottardo.

CAPACITA' CONOSCENZE DAL PROBLEMA AL PROGRAMMA

1. Analizzare, risolvere problemi e codificarne la soluzione con il linguaggio degli algoritmi
2. Gli obiettivi della programmazione.
3. I linguaggi di programmazione.
4. Il problem solving applicato alla programmazione:
5. analisi di un problema
6. progettazione di un algoritmo
7. codifica di un algoritmo.
8. La progettazione di un algoritmo:flow chart

I flowchart

I flowchart, diagrammi di flusso, sono una tecnica grafica che permette di visualizzare le azioni che deve eseguire il computer durante l'elaborazione del programma. I simboli sono quattro.



9. pseudocodice Le notazioni in uso negli pseudolinguaggi

COSTRUIAMO ALGORITMI CON LA PROGRAMMAZIONE STRUTTURATA

10. Formalizzare la soluzione del problema con le regole della programmazione strutturata
11. Le istruzioni di un algoritmo.
12. Le strutture di controllo.
13. La sequenza.

La selezione

La maniera più semplice per implementare una selezione è quella di usare il costrutti “if”.

Nella bibliografia è sempre presentato con l’alternativa decisionale e quindi nella forma “if-then-else”, in cui, nel caso di fallimento del test dell’espressione booleana si esegue l’alternativa contenuta nel corpo dell’else.

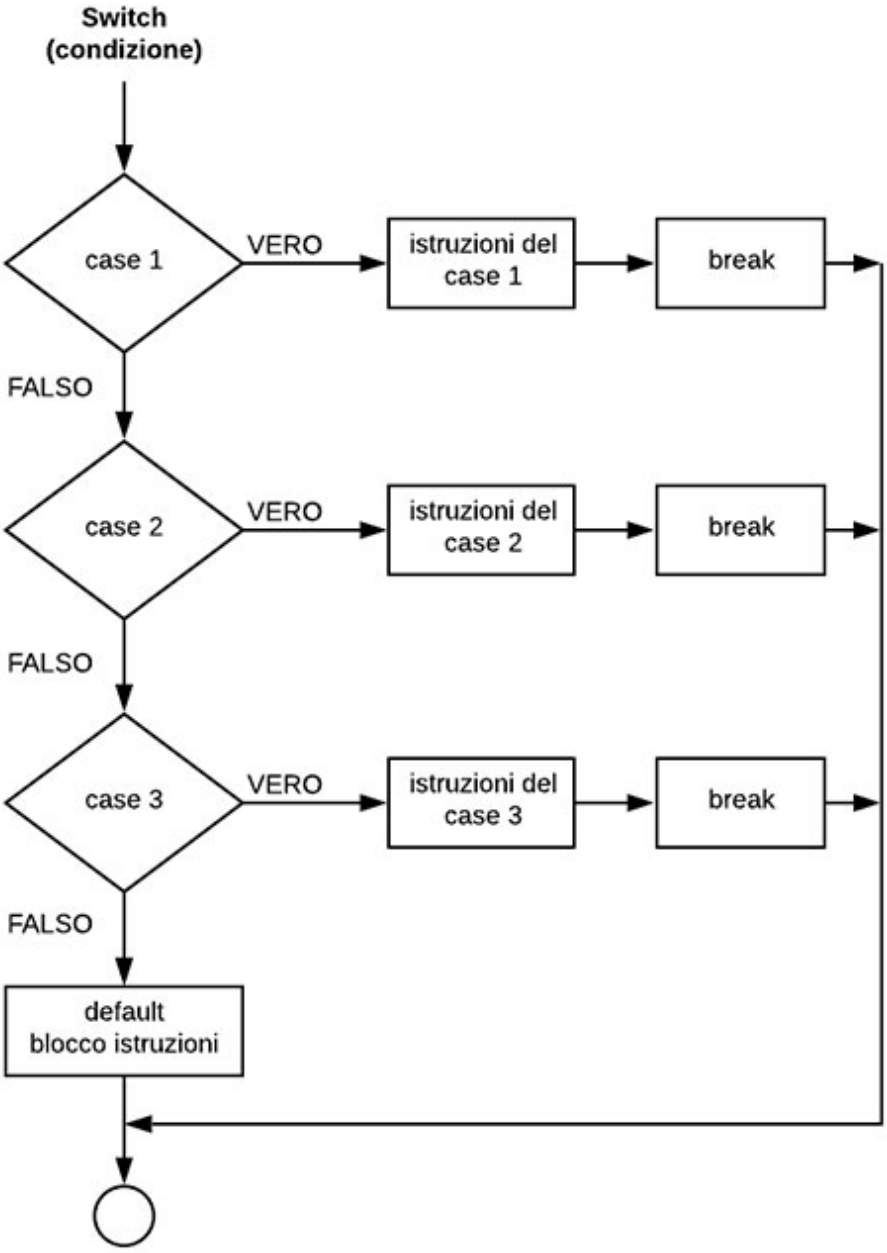
Spesso le decisioni potrebbero essere non binarie ma multi via, come ad esempio quando un carattere pervenuto dalla tastiera assuma i valori compresi tra ‘a’ e ‘z’.

La selezione multipla avviene quando in caso di fallimento del test che è l’espressione che controlla l’IF viene eseguita l’alternativa then.

Possono però esserci molti casi “validi” e non uno solo, come ad esempio quando un carattere ricevuto tramite trasmissione di qualunque genere da una periferica, arrivi in un buffer, e molte delle sue possibilità corrispondono ad una reazione concreta dell’automa, ad esempio ‘A’ = robot avanti, ‘I’ = robot indietro, e così via.

In questa situazione si opta per il costrutto multivia case switch.

Costrutto *switch case*



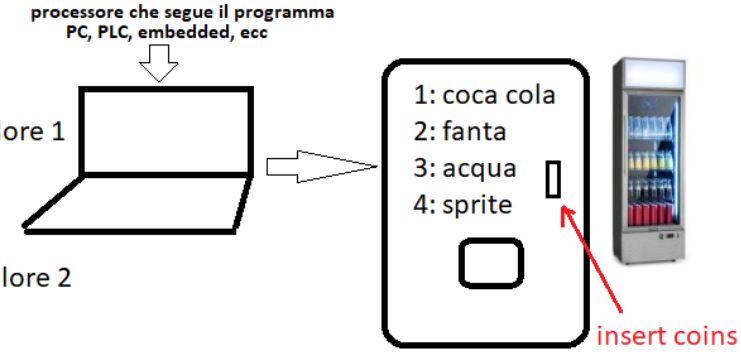
Applicazione del costrutto switch-case

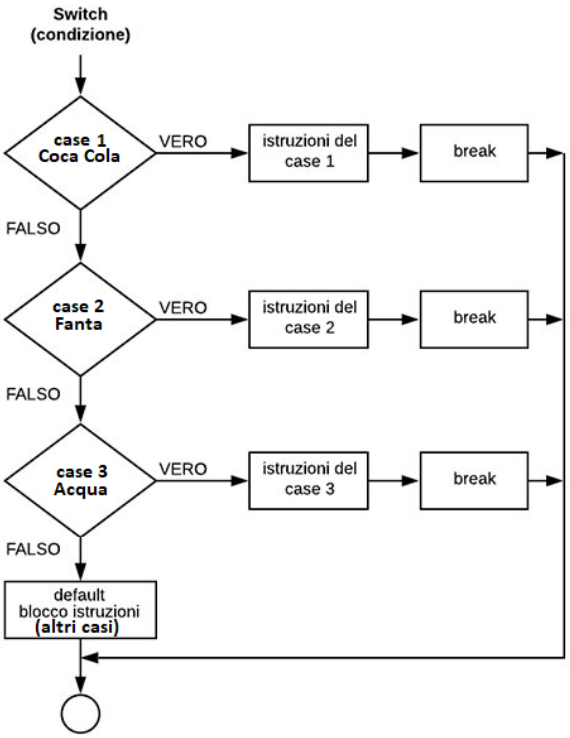
il costrutto switch case (un caso reale)

Lo switch - case è un if multiplo, o un selettore, che permette di decidere una reazione specifica a una richiesta particolare della stessa variabile.

Definita una variabile numerica, ad esempio 'a' quindi int a; questa può assumere valori diversi durante l'esecuzione del programma, ad esempio tramite lettura da tastiera.

```
int a;
case (a){
 //...caso 1
   //azioni in risposta al valore 1
   break:

//...caso 2
    //azioni in risposta al valore 2
break:
_end case
```






```
#include <stdio.h> //permette l'uso dei tipi standard

int bibita = 1;
bit coin = 0;
void main( )
{ //inzio del programma principale
  if (coin){
   switch (bibita)
   {
   case 1:
     Console.WriteLine("Coca Cola");
     break;
   case 2:
     Console.WriteLine("Fanta");
     break;
   case 3:
     Console.WriteLine("Acqua");
     break;
   case 4:
     Console.WriteLine("Sprite");
     break;
   default:
     Console.WriteLine("Insert coin");
     break;
   }
  }
}
```

L’iterazione

I programmi spesso necessitano di ripetere più volte uno stesso gruppo di istruzioni. le istruzioni che permettono di controllare la ripetizione vengono dette strutture cicliche o ripetitive. Le strutture cicliche permettono di ripetere una sequenza di operazioni, terminando quando si verifica una specifica condizione.

Un’iterazione è quindi un loop controllato che può iterare:

1. Zero volte, se la condizione di controllo è falsa fin dal primo incontro nel flusso di esecuzione
2. n (enne) volte, a priori non necessari di valore noto, in quanto se lo fosse si potrebbe semplicemente usare il costrutto for to do.
3. Infinite volte, se la condizione di test è sempre vera, ad esempio implementando un while(1) { } come avviene regolarmente in ambito embedded ovvero quando non si deve ripetere le impostazioni di registri e variabili ad ogni ripetizione del codice ma solo all’inizio, ad esempio nel settaggio di motori e sensori di un robot semovente (quelli con le ruote).

La realizzazione di algoritmi ripetitivi

Per realizzare un algoritmo ciclico si devono determinare:

- le inizializzazioni da effettuare prima del ciclo;
- le operazioni che devono essere ripetute all'interno del ciclo;
- la condizione di terminazione del ciclo;
- le istruzioni da eseguire alla fine del ciclo.

Il ciclo può terminare:

- quando è stato eseguito un certo numero di ripetizioni noto;
- quando l'utente lo richiede esplicitamente;
- quando viene inserito un valore particolare;
- più in generale quando si verifica una condizione particolare dipendente dal problema.

Una scelta più tecnica riguarda il fatto di utilizzare:

- la ripetizione con controllo della condizione in testa, cioè all'inizio del ciclo(precondizionale);
- la ripetizione con controllo della condizione in coda, cioè alla fine del ciclo(postcondizionale).

La differenza sostanzialmente riguarda il fatto di valutare la condizione di terminazione prima di eseguire il ciclo o dopo averlo eseguito almeno una volta. Un'altra differenza più marginale riguarda il fatto se si deve uscire dal ciclo quando la condizione è vera o quando la condizione è falsa.

L’algebra booleana e il suo ruolo nella programmazione strutturata

La migliore risposta alla domanda “L’algebra booleana e il suo ruolo nella programmazione strutturata” si può dare definendo la varabile di tipo binario e utilizzarla come elemento decisionale nelle espressioni testate da costrutti fondamentali come “if” “then” “else” e assimilabili.

In genere i costrutti di selezione, come quelli che generano dei loop controllati, eseguono il “corpo tra parentesi graffe se l’espressione è nel complesso pari a bool di valore TRUE.

TRUE= 1 logico, (in italiano vero).

in C++ come in tutti gli altri linguaggi di programmazione strutturata esiste la variabile binaria normalmente definita con il costrutto BIT oppure bool.

La sintassi in C++ è:

bool falg; //definisce una variabile per rilevare un evento

Esistono versioni del C e del C++ orientate alla programmazione dei dispositivi embedded, ovvero le schede elettroniche con un microprocessore a bordo come ad esempio

Arduino o le Micro-GT.

In questo caso la variabile booleana viene utilizzata spesso per definire lo stato di moviemento dei motori del robot. Ad esempio:

RB0 = 1; //il motore collegato al pin RB0 del processore è ON.

```
#include <iostream>
using namespace std;
void caso1(){
 cout<<"il valore inserito è maggiore della soglia";
}
void caso2(){
 cout<<"il valore inserito è minore della soglia";
}
Int main(){
 int valore;
 int soglia;
 soglia=10;
 cout<<" inserire dato numerico intero -> ";
 cin>>valore;
          if (valore>soglia){ //il test avviene sul bool true
      caso1();
          }
          else caso2();
//l'uso delle variabili boolena consente di convertire
//confronti numerici in decisioni dicotomiche, si o no,
// fare o non fare, chiamare o non chiamare.
```

//quindi una variabile bool puo deviare il flusso di
//esecuzione del programma grazie al costrutto if
return 0;
}

La variabile booleana può essere manipolata in test con i costrutti <,>, ==, ! (questo significa not). Oppure impostata con costrutti come =, ~ (inverti lo stato), ecc

LISTA DEI LINGUAGGI DI PROGRAMMAZIONE

Esistono centinaia di linguaggi di programmazione, ma solo 10 rientrano nella lista dei linguaggi di programmazione più richiesti e usati nel 2021 in Italia e nel mondo.

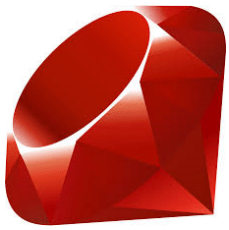
Imparare i linguaggi di programmazione più richiesti in Italia ti permette di avere una strada spianata verso un lavoro sicuro.

Molti di questi linguaggi di programmazione sono anche più pagati dalle aziende sempre alla ricerca di programmatori abili e rapidi nella ricerca di soluzioni algoritmiche e stilistiche.

Ogni linguaggio di programmazione ha delle caratteristiche specifiche, che lo rendono perfetto per risolvere determinate situazioni, ma inadatto per altre.

E' quindi il problema che definisce quale linguaggio sia meglio usare.

#10 – Ruby



Ruby è un linguaggio di programmazione open-source, focalizzato su semplicità e produttività. La sintassi di Ruby è semplice ed elegante: questo lo rende un linguaggio facile da scrivere e con una lettura naturale.

La popolarità di Ruby è dovuta anche al suo framework più diffuso, Ruby on Rails. Grazie a questo framework, sono stati realizzati web app e servizi web molto diffusi, come Twich, Zendesk, GitHub, Square, SoundCloud.

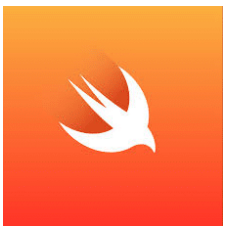
#9 – TypeScript

TypeScript è un linguaggio di programmazione open source sviluppato da Microsoft, e si tratta sostanzialmente di una versione estesa di JavaScript.

TypeScript è un superset di JavaScript: la sua forza è che qualsiasi programma scritto in JS è anche compatibile con la sintassi e la semantica TypeScript, senza alcuna modifica.

Con TypeScript si sviluppano applicazioni di grandi dimensioni, che vengono poi ricompilate in JavaScript per essere interpretate da browser o app.

#8 – Swift



Swift è un linguaggio di programmazione orientato agli oggetti. Il linguaggio Swift è stato sviluppato da Apple, ed è diretto ai programmatori dei sistemi Apple, nelle sue diverse versioni (macOS, iOS, watchOS…).

Versatile e potente, Swift è pensato per coesistere con il linguaggio Objective-C, altro linguaggio usato precendentemente in casa Apple. Uno dei maggiori punti di forza di Swift è la grande ottimizzazione, che permette di creare software estremamente veloci.

#7 – Go

Go è un linguaggio di programmazione sviluppato da Google, supportato sia da Google che dalla comunità di sviluppatori indipendenti visto che si tratta di un progetto open-source. È un linguaggio semplice da scrivere (semplice come Python) e al tempo stesso molto efficiente (efficiente come C++).

Alcune delle caratteristiche principali di Go sono il grande supporto dato alla programmazione concorrente, l'ottimizzazione dei tempi di compilazione anche per hardware modesti, e la presenza di un buon numero di strumenti di sviluppo integrati.

#6 – C/C++

C e C++ sono linguaggi di programmazione storici: sviluppati negli anni 70, sono tra i linguaggi più utilizzati della storia dell'informatica.

Proprio la loro diffusione li rende imprescindibili. C e C++ sono ancora oggi i linguaggi alla base di molti sistemi operativi, browser e videogiochi.

Non solo:

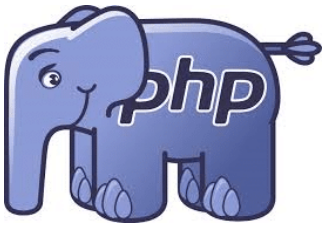
La richiesta di programmatori C e C++ è in continuo aumento, soprattutto grazie allo sviluppo dell'IoT. Buona parte dei chip presenti dentro gli smart devices, infatti, è programmata in C o in C++, che forniscono prestazioni ottimali.

#5 – C#

C# (pronunciato C Sharp, suona come “vedere nitidamente” – see sharp) è un linguaggio di programmazione orientato agli oggetti. Sviluppato da Microsoft, si presenta come un linguaggio concorrente di Java.

Il linguaggio C# è versatile: usato per programmare app e software Enterprise, web app e applicazioni per mobile, è il linguaggio di programmazione per eccellenza del Framework .NET.

#4 – PHP



PHP è un linguaggio di scripting interpretato, con una sintassi semplice e di uso molto diffuso. PHP supporta un approccio sia di tipo imperativo, che orientato agli oggetti.
Sviluppato per la programmazione di pagine web interattive e dinamiche, negli anni è stato spesso criticato e sottovalutato da parte di molti sviluppatori. Questo non ne ha impedito l’uso in moltissimi progetti di diffusione mondiale: giusto per fare qualche esempio, Wikipedia, Facebook e Yahoo sono stati programmati in PHP.
Ad oggi, PHP si usa per programmare applicazioni web server-side, script a riga di comando e applicazioni stand-alone con interfaccia grafica. Le ultime implementazioni hanno reso PHP ancora più veloce e affidabile, dando nuova popolarità a questo linguaggio di programmazione.

#3 – Python

Sul gradino più basso del podio si posiziona Python:
Python è un linguaggio di programmazione semplice da imparare, ha un codice facilmente leggibile ed è molto versatile. Si tratta infatti di un linguaggio di alto livello multi-paradigma, adatto alla programmazione orientata agli oggetti, alla programmazione strutturale e a quella funzionale.
Per queste sue caratteristiche, Python viene considerato da molti sviluppatori uno dei migliori linguaggi di programmazione con cui iniziare a programmare.
Il campo di applicazione più diffuso di Python è lo sviluppo di applicazioni web (Instagram ad esempio usa Python insieme al framework Django), ma sta avendo grandi riscontri anche in ambiti di *machine learning* e analisi di dati.

#2 – Java

Java si assicura la seconda piazza nel podio dei migliori linguaggi di programmazione 2021.
Non che sia una sorpresa:
Java è incredibilmente diffuso, grazie alle caratteristiche che lo rendono uno dei linguaggi più stabili, completi ed affidabili per costruire sistemi complessi – LinkedIn è scritto in Java, ad esempio.
Java inoltre è stato concepito seguendo il motto *Write Once, Run Everywhere*. Una volta scritto il codice, può girare senza necessità di ricompilazione. Tradotto, significa che Java è un linguaggio indipendente dalla piattaforma su cui gira. In più, gli innumerevoli frameworks Java consolidano la capacità dell'ecosistema Java (JSE, J2EE, JVM…) di offrire ai programmatori uno sviluppo rapido e solido di applicazioni di ogni tipo.
Inoltre Java è il linguaggio di programmazione che sostanzialmente sta dietro al sistema operativo Android, per cui è alla base anche dello sviluppo di app e giochi per mobile.
Come dire, il futuro di Java è assicurato!

#1 – JavaScript

Il primo posto dei linguaggi di programmazione più richiesti va a JavaScript.
JavaScript is King.
Le ragioni dietro a questo primato sono presto dette: JavaScript è un elemento essenziale per lo sviluppo di siti internet con funzioni dinamiche, interattive o animate. Per questo, JavaScript è presente in gran parte del web.
Nato per girare su client e codificare front-end, adesso JavaScript si sta diffondendo anche per l'uso in back-end, grazie a NodeJS che consente di eseguire codice JavaScript lato server. Inoltre, si usa anche per sviluppare giochi e applicazioni desktop.
Completo, versatile, diffuso, sviluppato: con queste caratteristiche, la permanenza di Javascript sul podio è assicurata fino alla prossima evoluzione informatica!

Comprendere le differenze di funzione e di stile inerenti i vari linguaggi di programmazione

La prima sostanziale differenza tra lo stile di un linguaggio di programmazione ed un altro sta nella tipologia linguaggio compilato o linguaggio interpretato.

I linguaggi compilati generano un file eseguibile che può essere trasportato in altri PC in cui sia in esecuzione un sistema operativo uguale o affine, mentre il secondo viene eseguito runtime e direttamente nella RAM della macchina in uso.

Consideriamo ad esempio il linguaggio Python, a cui riga per riga vengano fornite i dati che costituiscono l'input.

Ad ogni inserimento il dato si trova in esecuzione in un contesto algoritmico di calcolo quindi all'invio si ha l'esecuzione e la produzione di un risultato alla riga successiva.

Questo non avvien nei linguaggi compilati che invece richiedono la completezza dell'algoritmo che si intende sviluppare prima di mandarlo in esecuzione allo scopo di ottenere il suo output.

Risolvere problemi con l'ausilio delle strutture astratte dei dati
Una struttura dati, o struct, è una tecnica software che permette di descrivere un oggetto con l'ausilio di più variabili, anche di tipo diverso, in esso contenute.

In alcuni linguaggi di programmazione, oggi da considerarsi un po' arcaici, le struct erano chiamate record e le variabili che la compongono erano dette campi del record.

Le strutture sono più consone dei linguaggi che non permettono l'uso della programmazione orientata agli oggetti, infatti, quando questo è possibile come nel C++, le struct, tra i loro campi, contengono anche funzioni oltre che singole variabili.

Queste funzioni assumono il nome di metodo della classe e donano all'oggetto le tre caratteristiche principale della OOP ovvero ereditarietà, incapsulamento e polimorfisfo.

Questa cosa non è possibile nelle semplici strutture.

L'esempio più comune per l'applicazione di una struttura è una scheda anagrafica che descrive una persona.

Una persona ha, ad esempio, il nome, il cognome come stringhe, il peso e l'altezza, come numeri reali ovvero float, l'età e il numeri scarpe come numeri interi INT, e così via.

L'uso delle strutture possono, grazie a procedimenti di memorizzazione, ordinamento, e ricerca dare origini a quelle che si chiamano le strutture di dati.

Le famose Query, non sono altro che funzioni di filtro e ricerche applicate su una struttura dati di ti tipo relazionale e/o gerarchico, allo scopo di identificare un record su nolti migliaia, utilizzando una specifica chiave di ricerca da identificarsi in uno dei campi del record. Ad esempi la variabile nome.

Codificare algoritmi in un linguaggio di programmazione

Per definizione l'Algoritmo è una sequenza di passi che, se intrapresa da un esecutore, permette di ottenere i risultati attesi a partire dai dati forniti.

Cosa fondamentale e che di certo va detta per prima è che l'algoritmo è un procedimento e quindi di natura indipendente sia dal linguaggio di programmazione utilizzato che dall'hardware in cui questo verrà eseguito.

Quando invece il linguaggio è un vincolo per il programmatore, il generico algoritmo deve essere implementato con la sitassi di destinazione.

E' necessaria la lista delle librerie e dei costrutti che lo specifico linguaggio possono riconoscere ed eseguire.

L'algoritmo essendo di per se stesso eseguibile in maniera universale deve essere scritto in un linguaggio universale detto pseudo linguaggio.

Si po' ad esempio definire un algoritmo di entrata in una abitazione da parte di inquilino, ad esempio, passo 1: portarsi di fronte alla porta. Passo 2: infilare la mano nella tasca destra. Passo 3: se non trovo le chiavi infilare la mano nella tasca sinistra. Passo 4: se non trovo la chiave provare a suonare il capannello. Passo 5: se trovo la chiave infilarla nella serratura e aprire. Passo 6: entrare fino ad avere la porta alle spalle. Passo 7: chiudere la porta. Passo 8: togliere le scarpe. Fine algoritmo.

Eventualmente, i risultati ottenuti come esecuzione dell'algoritmo "entrare in casa" potranno essere passati come set di dati in input dell'algoritmo che descrive cosa fare quando si è in casa.

E così via con successivi algoritmi. I primi giochi Platform, ad esempio il famoso Zac MacCraken della Lucas Art, era stato sviluppato in questa maniera considerando il personaggio come un automa da accompagnare all'interno di un mondo virtuale alla ricerca di alcuni misteri marziani in America.

Programmiamo in C

Prima di entrare nel merito della programma zione C è necessario procurarsi un IDE ovvero una piattaforma che contenga un compilatore in grado di interpretare e assemblare il codice sorgente per poi mandarlo in esecuzione.

Esistono molte piattaforme gratuite, sia istallabili che utilizzabili direttamente online,

- DevC++ (versione vecchia: Bloodshed / versione nuova: Orwell) funzionante su Windows
- Code::Blocks, per sistemi operativi Windows, Linux, Mac OSX
- Xcode che funziona sui sistemi operativi Mac OSX

Un compilatore utilizzabile direttamente online è disponibile a questo indirizzo.

https://www.onlinegdb.com/online_c++_compiler

Il lavoro inizia con la creazione di un progetto con il quale si intende la definizione della struttura di percorsi (path) e cartelle contenenti i files (detti moduli) che saranno impiegati, inoltre vengono decisi gli strumenti, ad esempio la versione del compilatore, e le librerie.

Normalmente, negli IDE che usano lo stile Eclipse, il project tree è il pannello di sinistra e mostra tutte queste cose.

I moduli sono di due tipi, quelli che contengono le definizioni, le descrizioni e i prototipo di classi e funzioni, detti moduli di intestazione, con estensione .h e quelli che contengono la logica, il flusso del codice ovvero le chiamate delle funzioni con i relativi passaggi di parametri ecc, detti moduli sorgenti, con estensione .c

Per usare Dev-C++ occorre creare un progetto che include tutti i file necessari nel processo di sviluppo, questi sono non solo .c, .h, .obj, .exe ma anche tutti i file necessari a Dev-C++ per gestire l'intero processo, ad esempio il makefile, che si cura dell'assemblaggio dei moduli obj prodotti nella prima fase della compilazione.

DevC++ rende disponibili diversi "tipi" di progetti, alcuni dei quali già includono frammenti di codice e librerie, specie per l'interazione con il sistema operativo.

Questi sono:

- console application: finestra DOS (quello più diretto e utile per le applicazioni didattiche)
- Windows Application che crea la finestra windows da popolare con il nostro programma.
- Windows DLL
- Windows Static Library

E' possibile creare nuovi files oppure aggiungere files esistenti all'attuale progetto, che siano essi di tipo sorgente (.c) oppure di intestazione (header files .h).

• Per creare un nuovo .c: Menu File/New/Source File oppure Menu Project/Source File (viene creato un file senza titolo, comunque il programma ne chiede il nome prima della compilazione o salvataggio)

• Per aggiungere un file esistente: Menu Project / Add to Project

• Per modificare un file (ad esempio per scrivere il programma!): - selezionare il file da modificare con la finestra di sinistra - il contenuto si modifica (come visto) attraverso la finestra centrale - il nome può essere modificato dalla finestra di sinistra con tasto destro del mouse + Rename file

Dato un problema, questo va prima formalizzato, ovvero si devono identificare di nomi e dei tipi per le variabili che ne rappresentano il problema.

Ad esempio per fare la media di 4 valori definire 5 variabili di tipo float.

float voto1,voto2,voto3,voto4, media;

poi si esegue il calcolo scaricando il risultato della somma divisa per 4 sulla variabile media.

media=(voto1+voto2+voto3+voto4)/4;

Per creare un buon programma in C o C++ bisogna prendere il problema iniziale e suddividerlo in sotto problemi più semplici secondo la tecnica detta top down.

Ogni sotto problema verrà poi risolto con uno specifico gruppo di codice che realizza la funzione.

Una funzione C o C++ rimane definita da un tipo, un nome e una lista di parametri, che costituiscono la **firma** della funzione, oltre che al corpo della funzione compreso tra due parentesi graffe.

Tra chi chiama e chi viene chiamato esistono due tipologie di passaggio dei parametri ovvero i dati su cui operare, il primo metodo comporta il passaggio di una copia dei valori, (by value) e il secondo fornendo l'indirizzo della locazione di memoria che contiene il dato (by reference).

Nel passaggio per valore si opera in copie delle funzioni, quindi gli originali non subiscono modifiche, così alla fine dell'esecuzione della specifica funzione ritrovo i valori originali.

Nel passaggio by reference invece all'uscita dalla funzione trovo il valore modificato in maniera permanente.

Prima di cominciare a programmare è importante procurarci la lista dei costrutti disponibili e delle librerie per le funzioni particolari, ad esempio le librerie matematiche.

Tipi di dato e operatori del C++

Parola chiave	Tipo	Dimensioni	Valori
`char`	Carattere singolo	1	1 codice ASCII (da 0 a 255)
`float`	Reale singola precisione	4	±1.2E-38 ÷ ±3.4E38
`double`	Reale doppia precisione	8	±2.2E-308 ÷ ±1.8E308
`short int`	Intero	1	-128 ÷ 127
`int`	Intero	2	-32768 ÷ 32768
`long int`	Intero	4	-2147483648 ÷ 2147483647
`bool`	Booleano	1	True (1) / False (0)
`string`	Stringa	n	lunghezza variabile, ASCII

ARITMETICI	
+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
%	modulo
++	incremento
--	decremento
>>	shift di bit a destra
<<	shift di bit a sinistra

RELAZIONALI	
==	uguale
>	maggiore
>=	maggiore uguale
<	minore
<=	minore uguale
!=	diverso

CARATTERE	
+	concatenazione

CONDIZIONALE	
?	condizione

LOGICI	
&&	And
||	Or
!	Not

LOGICI sui BIT	
&	And
|	Or
^	Xor
!	Not
~	Complemento a 1

Formattazione stringhe e caratteri di escape

Inserendo la sequenza barra rovesciata seguita da uno dei caratteri in tabella si ottine quanto indicato nella colanno do descrizione, ad esempio un a capo riga o un segnale acustico.

\0	carattere vuoto, NUL (0)
\a	allarme, beep
\b	backspace
\f	salto pagina, FF (12)
\n	salto riga, LF (10)
\t	tab.orizzontale, HT (9)
\v	tab.verticale, VT (11)
\\	carattre \
\?	carattre ?
\'	carattre '
\"	carattre \"

%d	int – numero intero
%i	int – numero intero
%c	int – codice ASCII
%s	char* - stringa
%f	double – numero con 6 decimali
%x.yf	double – numero con x cifre di cui y decimali
%p	void* - indirizzo in memoria
%x	int – numero intero esadecimale
%e	double – numero decimale esponenziale
%%	carattere %
...	etc. etc. (altri tipi di formattazione)

Usare il costrutto system per eseguire i comandi DOS

```
system("cls");  // pulisce lo schermo
```

```
possiamo anche attivare programmi di windows ad esempio:
system("pause");     // pausa dell'esecuzione
system("calc");      // lancia la calcolatrice
system("notepad");   // lancia notepad
system("dir");       // mostra i files in cartella
system("data");      // cambia la data da tastiera
system("print file.txt");  //stampa il contenuto
```

Le librerie del C++

La direttiva #include <nome.h> permette di inserire tutto il codice sorgente contenuto nel file nome.h al posto della riga.

All'interno del file sono contenute le definizioni delle funzioni che utilizzeremo nei nostri programmi.

Segue la lista delle funzioni C++ contenute nelle rispettive librerie.

#include <stdlib.h>

atoi()	converte una stringa in numero intero	n = atoi(s);
itoa	converte un numero intero in una stringa	s = itoa(n);
atof()	converte una stringa in numero reale float	x = atoi(s);
srand()	inizializza il generatore di numeri casuali	srand(time(0));
rand()	restituisce un numero casuale intero	n = rand() %10; [n = 0÷9]
system()	esegue um comando MS-DOS	system("dir");
malloc()	alloca memoria centrale	int *a; a = malloc(2);
free()	libera la memória centrale allocata	free(a);
qsort()	ordinamento (QuickSort)	
bsearch()	ricerca binaria	

#include <stdio.h>

printf()	stampa una stringa (formattata con costanti e variabili)	prinf("totale %d\n",x);
scanf()	richiede valori passando i puntatori alle variabili	scanf("%d %d ",&x,&y);
getchar()	legge carattere da tastiera attendendo il tasto	ch = getchar(); / getchar();
putchar()	visualizza un carattere	putchar('m');
gets()	legge una stringa da tastiera	gets(nomevar);
puts()	visualizza una stringa	puts(" buongiorno! ");
flushall()	svuota il buffer di input	flushall();
FILE	tipo di dati "file", per descrittori di file dati	FILE *fp
fopen()	apre un file di dati	fp = fopen("file.txt","r");
fclose()	chiude un file di dati aperto	fclose(fp);
fread()	legge da un file di dati	
fwrite()	scrive su un file di dati	
fseek()	si posiziona su un record del file di dati	
fputs()	memorizza una stringa sul file	
fputc()	memorizza un carattere sul file	
fgets()	legge una stringa (che finisce con '\n' o '0') dal file	
fgetc()	legge un carattere dal file	

#include <iostream.h>

cin	input da tastiera	cin >> a;
cout	output a video	cout << "risultato: " << x;

#include <conio.h>

getch()	legge carattere da tastiera attende il tasto non visualizza	ch=getch(); / getch();
getche()	legge carattere da tastiera attende il tasto visualizza	ch=getche(); / getche();
clrscr()	pulisce il video	clrscr();
gotoxy()	posiziona il cursore a colonna e riga specificate	gotoxy(40,10);
wherex()	restituisce la posizione orizzontale del cursore	x=wherex();
wherey()	restituisce la posizione verticale del cursore	y=wherey();
kbhit()	controlla se è stato premuto un tasto da tastiera	printf(" premi un tasto"); while (!kbhit());

#include <fstream.h>

ifstreamf()	definisce (ed apre) un file di input (nell'esempio 'fin')	ifstream fil("dati.txt");
ofstreamf()	definisce (ed apre) un file di output (nell'esempio 'fou')	ofstream fou("risult.txt");

#include <string.h>

strlen()	restituisce la lunghezza della stringa	n=strlen(" buonasera");
strcpy()	copia la seconda stringa sulla prima	strip(s1,s2);
strcat()	concatena la seconda stringa alla prima	strcat(s1, "ciao");
strstr()	cerca s2 in s1 e se c'è da la sua posizione, se no NULL	p = strstr(s1, "pippo");
strchr()	cerca un carattere in una stringa e se c'è da la sua posizione, se no NULL	p = strstr(s1, "p");
strcmp()	confronta due stringhe e restituisce 0 se uguali, < 0 se s1 < s2, > 0 se s1 > s2	if (strcmp(s1,s2)= =0).

#include <math.h>

abs()	valore assoluto di un numero intero	a = abs(a);
sqrt()	radice quadrata di un numero double	x = sqrt(y);
pow()	elevamento a potenza (primo base, secondo esponente)	z = pow(x,y);
sin()	seno, coseno, tangente di un angolo	y = sin(x); y = cos(x);
cos()		y = log(x);
tan()		y = log10(x);
log()	logaritmo naturale	
log10()	logaritmo in base 10	

#include <time.h>

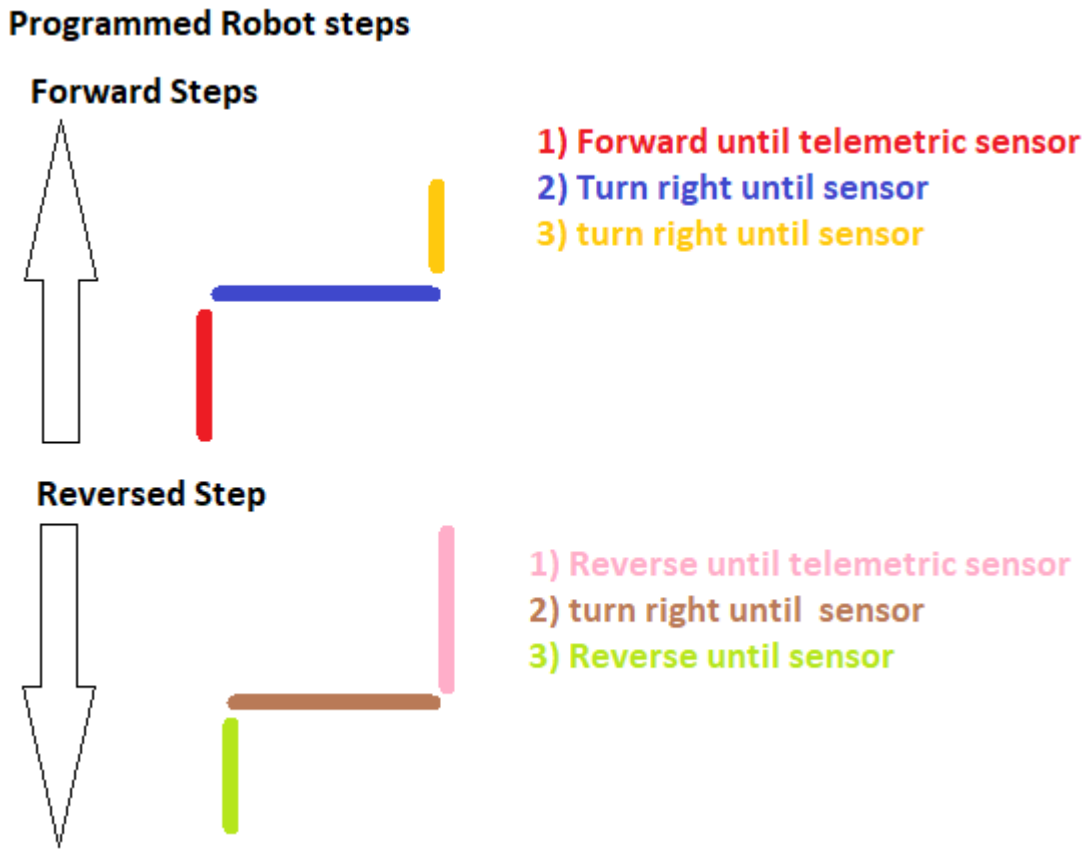
time()	da il numero di secondi associato al timer	srand(time(0));
difftime()	da la differenza in secondi tra due tempi/date	
clock()	da il numero di 'tick' della CPU dall'inizio del processo	

Nota: le librerie mostrate non sono tutte ma le principali e più usate, per approfondimenti si rimanda al manuale del compilatore.

Per includere librerie non standard o file header usare #include ‘.h’

Il costrutto sequenza

Sfrutta i cicli di esecuzione del main program, quindi non si tratta di un vero comando ma una sequenza di comandi o azioni elementari.
Prendiamo come esempio un robot che deve svolgere delle mansioni ripetitive cicliche
Supponiamo che le azioni del robot siano numerate 1,2,3,4.
Una volta arrivato all’ultima posizione, il robot deve tornare indietro per la stessa strada.
Vogliamo quindi eseguire le azioni ogni volta che un sensore ci restituisce il consenso.
I movimenti del Robot sono dati dal seguente diagramma di navigazione:




```
/*******************************************
Prof. Gottardo Marco Ph.D. Scuola SIIC 2020/2021
Classe seconda Liceo indirizzo tecnologico
Lezione di implementazione delle sequenze in C++
Il programma controlla un robot semovente.
I movimenti sono identificati con dei numeri
che attiveranno delle funzioni.
Le funzioni di base sono:
RobotAvanti();
RobotIndietro();
```

```
RobotDestra();
RobotoSinistra();
Vogliamo impostare delle sequenze fisse di movimento
con avanzamenti e rotazioni, sia in avanti che
all'indietro.
*******************************************/

#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;
void RobotAvanti(){
        cout<<" Mi sposto in avanti \n";
}

void RobotIndietro(){
        cout<<" Mi sposto all\'indietro \n";
}

void RobotDestra(){
        cout<<" Giro a detra e mi sposto in avanti \n";
}

void RobotoSinistra(){
        cout<<" Giro a sinistra e mi sposto in avanti \n";
}

int main()
{
   char step;
   int value=0;
   cout<<"press F for Robot sequnce Forward or R for Reverse -> ";
   cin>>step;
   if (step=='F') {
      value++; //incrementa il valore
      cout<<"primo passo Robot Avanti "<<value<<"\n";
      RobotAvanti();
   }
   if (step=='R') {
      value--; //decrementa il valore
      cout<<"primo passo Robot indietro "<<value<<"\n";
      RobotIndietro();
                  }
   // esegue il secondo passo delle sequenza
      if (step=='F') {
      value++; //incrementa il valore
```

```
    cout<<"secondo passo Robot Avanti "<<value<<"\n";
                RobotDestra();
  }
  if (step=='R') {
    value--; //decrementa il valore
    cout<<"secondo passo Robot indietro "<<value<<"\n";
                RobotoSinistra();
  }
  // esegue il terzo passo
      if (step=='F') {
    value++; //incrementa il valore
    cout<<"terzo passo Robot Avanti "<<value<<"\n";
                RobotAvanti();
  }
  if (step=='R') {
    value--; //decrementa il valore
    cout<<"terzo passo Robot indietro "<<value<<"\n";
                RobotIndietro();
  }
  cout<<"Fine della sequenza: Bye bye";
  return 0;
}
```

Scriviamo un programma

Per scrivere un programma in C, come in altri linguaggi è necessario avere un editor, ovvero un foglio per l'elaborazione dei testi.

L'editor fornisce poi il suo contenuto a uno strumento detto linker che esegue la costruzione di un unico codice sorgente quando questo sia composto da più parti in più file separati.

Il terzo strumento è il compiler in italiano compilatore, che prende il file lincato e ne crea un codice macchina esadecimale che può essere eseguito dal processore di destinazione.

Oggi, editor, linker, e compiler sono quasi sempre integrati in un'unica piattaforma detta IDE (integrated development enviroment) o sistema di sviluppo integrato, che permette di lavorare in un unico ambiente cosi che la compilazione possa avvenire con un unico comando o click in un tasto.

Esistono anche compilatori online, ovvero disponibili in internet e utilizzabili con un browser qualunque, ad esempio google crome.

Questa soluzione è molto utile specialmente in ambito didattico così che la scuola non deve dotare le postazioni di compilatori.

Una soluzione molto buona per la didattica è l'uso del compilatore DevC++, gratuito, scaricabile da internet con delle librerie molto complete.

Su questo compilatore è basato il libro del prof. Gottardo adottato nella scuola SIIC.

Un programma in linguaggio C o C++ o ANSI C è strutturalmente formato da moduli (file esterni) e funzioni, il cui nome può essere sovracaricato, quindi avere lo stesso nome ma con diversa lista di parametri, quindi la funzione int somma(a,b) è diversa dalla funzione int somma(a,b,c).

È diversa anche se il nome della funzione contiene lettere maiuscole o minuscole situazione che in inglese si chiama case sensitive.

Somma (a,b){} è diversa della funzione somma(a,b){} perché nella seconda la s è minuscola.

Linguaggio modulare e funzionale

Il linguaggio C è **modulare e funzionale**, ovvero è diviso in più file di cui il primo ha estensione .h ovvero header che contiene i prototipi delle funzioni e il .c che invece contiene le logiche e le chiamate alle funzioni, sono di fatto il codice sorgente.

L' inserimento di un modulo esterno avviene con il costrutto #include che è una direttiva al preprocessore.

Viene infatti lincato il contenuto del file di destinazione al posto di questa riga.

L'include può avere come destinazione un file indicato tra parentesi angolari <math.h>, che indica una cartella, di solito la include, del sistema operativo, oppure gli apici "settaggi" che indica la cartella in cui il programma è stato creato dal programmatore.

Il programma è poi diviso in funzioni la quali possono restituire il calcolo eseguito nel loro nome e tipo, ad esempio la funzione int somma( ) alla fine dell'esecuzione restituisce un valore intero contenuto nella variabile somma.

Diventa quindi possibile la sintassi: cout<<somma(a,b);

Le variabili dichiarate dentro a una funzione si chiamano locali mentre quelle dichiarate all'esterno di ogni funzione si chiamano globali.

È possibile passare le variabili da una funzione a un'altra funzione fornendo una copia, cosi che le elaborazioni non rovinano il volare originale, oppure per indirizzo perché siamo interessati che le modifiche siano permanenti.

La compilazione ed esecuzione

La compilazione è quel processo che dopo la stesura di un programma sorgente in un qualsiasi editor di testo traduce le istruzioni in esso contenute in linguaggio macchina, ovvero un file genericamente parlando prima esadecimale e poi binario che risulti eseguibile dal computer. La compilazione viene eseguita da uno strumento software chiamato compilatore.

I linguaggi interpretati, saltano la costruzione "bulding" del codice esadecimale o eseguibile traducendo ed eseguendo direttamente, tramite la memoria RAM, ogni singola istruzione del programma. E' un esempio di programmazione interpretata il Java e il Python, come molti altri quali il mathlab, utilizzati nella ricerca.

Nella tipologia compilata Il programmatore scrive il codice sorgente (in C o in C++) con un editor e lo memorizza in un file.

Il compilatore genera il codice macchina e lo salva in un file eseguibile, questo potrà essere lanciato in esecuzione nelle successive volte senza dovere ricompilare.

In sostanza, il sistema di sviluppo **Integrated Development Environment** (Ambiente di sviluppo integrato), ad esempio il DevC++, usato da molti istituti scolastici in quanto gratuito e sufficientemente performante per ogni scopo didattico, comprende una varietà di strumenti coordinati per supportare il processo di sviluppo dei programmi (creazione, traduzione,esecuzione, test, ecc), tra cui:

- editor
- compilatore
- linker
- debugger
- strumenti per la gestione delle configurazioni
- analizzatori statici, strumenti per il test, ecc

In sostanza, un buon IDE, riesce a supportare e automatizzare (parte del) processo di sviluppo di un progetto software, fornendo in output un file eseguibile trasportabile ina altri calcolatori.

Qualora il sistema non richieda l'istallazione sulla macchina in cui va eseguito questo si dirà "portable".

Se il sistema potrà essere eseguito su più sistemi operativi diverso si dirà "cross platform" a volte indicato con Xplatform. Questa tipologia spesso crea una cartella in cui è sviluppato un mini sistema operativo virtuale che permette al programma di vivere sempre nel suo ambiente.

Dichiarazione di variabili

In ogni linguaggio di programmazione esistono delle metodologie analoghe per la dichiarazione delle variabili.

A seconda di dove la variabile viene dichiarata assume un ambito (luogo in cui) essa è visibile e utilizzabile.

L'area della dichiarazione della variabile dipende da come vogliamo isolarla dal resto del problema.

La variabile dichiarata fuori da ogni funzione, compreso il main, si dice **globale**.

Gli esperti informatici cercano di **non** dichiarare variabili globali ma locali all'interno delle funzioni.

Tra le variabili globali e quelle locali esistono due metodi di passaggio del valore.

Questi sono:

1. **Metodo per copia**, detto per valore, in inglese by value. Si crea una copia della variabile e la si trasferisce all'interno della funzione. In questo luogo circoscritto la copia subisce delle modifiche e restituisce dei risultati locali. Alla fine dell'esecuzione della funzione, non rimane traccia delle modifiche eseguite perché si torna ai valori originali che la variabile aveva prima di entrare nella funzione. In pratica ogni modifica è stata fatta su una copia e non sull'originale.
2. **Metodo per indirizzo**, detto anche by referece, che richiede l'uso di puntatori. Si fornisce alla funzione la locazione di memoria in cui la variabile è locata. Ogni modifica è permanente in quanto non esistono copie di quella variabile. Alla fine dell'esecuzione della funzione le variazioni eseguite rimangono.

Quando di crea una funzione, ad esempio:

```
int somma( ){

        int a,b, somma; //sono variabili locali non visibili nelle altre funzioni.

        Somma = a + b;  //esecuzione locale della somma

        cout<<"la somma vale "<<somma;  //il risultato è visibile solo all'interno della funzione

        return somma; //copia il valore ottenuto nella variabile che si chiama somma, ovvero il nome della
        //funzione.

}
```

È possibile passare alla funzione delle variabili dichiarate all'esterno di essa.

Bisogna conoscere i costrutti (i costrutti sono i comandi del linguaggio o parole chiave, in inglese si chiamano statement) reference * e dereference &.

- Il costrutto ***** estrae il valore da una locazione di memoria
- Il costrutto **&** ricava l'indirizzo della variabile in cui è contenuto il valore.

Vediamo un esempio:

```
#include <iostream>
using namespace std;
int somma( int *a, int *b){  //i parametri tra parentesi sono le variabili in ingresso tramite l'indirizzo
int somma;  //la variabile intera somma è locale alla funzione somma
somma = a + b;  //si esegue in locale la somma sui valoro arrivati dal main
return somma;  //si restituisce il valore della somma nel nome e nel tipo della funzione
}

int main( ){
int a,b; //dichiarazione delle variabili addendi locali al main
cout<<"inserire il primo addendo";  //verbose acquisizione del primo addendo
cin>>a; //acquisizione del primo addendo
cout<<"inserire il secondo addendo"; //verbose acquisizione del secondo addendo
cin>>b; //acquisizione del secondo addendo
cout<<"la somma vale "<<somma (&a,&b); // stampa e passaggio dei parametri per puntatore alla somma
return 0; //chiusura della funzione main
}
```

Tipi di dati

I tipi di dati, in C come in C++, come in altri molti linguaggi, sono di tipo predefinito oppure di tipo definito dall'utente.

Tutti i linguaggi devo poter accettare i tipi base, ovvero:

bool -> spesso ha sintassi "bit", e esprime una variabile binaria che può assumere solo due stati, true o false, sinonimo di 1 o 0.

Le variabile binarie spesso si definiscono FLAG, ed hanno lo scopo di rilevare situazioni e permettere l'avanzamento dello routine.

Ad esempio:

```
bool detect;

int addendo1, addendo2, somma; // valore possibile  -32768 fino a +32767
```

Rilevare tramite un flag booleano se la somma inserita è maggiore di una soglia.

```
void calcola(){
    bool detect;
    int addendo1, addendo2, somma; // valore possibile  -32768 fino a +32767
    cout<<"Inserisci prmo addendo -> ";
    cin>>addendo1;
```

```
        cout<<"Inserisci secondo addendo -> ";
        cin>>addendo2;
        cout<<" il valore massimo intero e\' -> +32767\n";
        cout<<" il valore minimo intero e\' -> -32768\n";
        somma = addendo1 + addendo2;
        cout<<"la somma vale -> "<<somma<<"\n";
        if (somma>10) detect= true;
        else detect=false;
                detect = getch();
        if (detect == 1) cout<<" la somma e\' maggiore della soglia\n";
        if (detect == 0) cout<<" la somma e\' minore della soglia\n";
}
```

Esistono inoltre il tipo **float** (virgola mobile) per la rappresentazione dei numeri con la virgola a 32 bit e i tipi di dato composti come gli array (insiemi di dati semplici indicizzati) e i tipi definiti dall'utente.

Il range delle variabili dichiarate Floating-Point è nella tabellina, con valori espressi nella notazione esponenziale scientifica.

Type	Minimum value	Maximum value
float	1.175494351 E - 38	3.402823466 E + 38
double	2.2250738585072014 E - 308	1.7976931348623158 E + 308

Nei 32 bit disponibili per ogni variabile float dichiarata, il prima a sinistra è il segno, i successivi 8 sono l'esponente espresso in eccesso 127, e gli ultimi 23 la mantissa normalizzata con la prima cifra a destra dello zero e virgola diversa da zero.

Questo modo di scrivere i numeri Float, detti anche reali, si chiama formato IEEE 754 32-bit base-2.

Assegnazione di valori alle variabili

Per assegnare dei valori alle variabili si usa il comando = diverso dal comando == perché il primo associa il valore di quello che c'è a destra del segno a quello che c'è a sinistra del segno, mentre il secondo costrutto == esegue una verifica dei valori e li confronta invece che assegnarli.

Ad esempio, quando voglio mettere il numero 3 nella variabile a scriverò:

```
int a=3;  //associa il valore 3 alla variabile a
```

quando invece voglio verificare se una variabile di controllo ha raggiunto un certo valore allora useràò il costrutto di confronto == in questa maniera:

```
if (a==0){
cout<< inserire zero o uno;
cin>> a;
}
```

In questo esempio viene sfruttato il ciclico del main per continuare a chiedere l'inserimento di un valore fino a che non viene digitato un numero diverso da zero.

Il costrutto selezione

Il costrutto di selezione in quasi tutti i linguaggi strutturati ad alto livello è il comando
if (selettore) { // Corpo del if
// programma di reazione alla selezione
} // fine dell'azione se la variabile selettore risultava TRUE

La variabile selettore, in questo caso di tipo BOOL, può essere un pulsante di comando di una macchina operatrice.
Quando il pulsante è premuto il programma esegue il corpo dell'IF.
Se il pulsante NON viene premuto allora il programma ignora (salta) il corpo dell'IF e passa e passa all'istruzione successiva.
Questo comando serve per permettere all'esecuzione del programma di selezionare strade diverse, in funzione dello stato delle variabili.
Nelle parentesi rotonde può esserci qualsiasi espressione a patto che il risultato finale sia Vero o Falso.
Esempio: La variabile intera, conteggio, definita con l'istruzione:
int conteggio = 0; //definisce la variabile e la inizializza a zero
Incrementa il suo valore a ogni ciclo di esecuzione del programma.
vogliamo identificare quando conteggio risulta maggiore di 100.
esempio:

```
void ciclica( ){
        int conteggio = 0;
        while (conteggio < 200){
                if (conteggio <= 100){
                motore=0;
                conteggio ++;
                }
                if (conteggio>100){
                motore = 1;
                conteggio++;
                }
                if (conteggio >= 200){
```

```
            conteggio=0;
            motore =0;
            }
    }
```

Vediamo un secondo semplice esempio dove il costrutto IF viene usato per determinare se il primo valore inserito è maggiore o minore del secondo.

```
#include <iostream>
using namespace std;
void select() {
    int primo, secondo;
    cin>>primo;
    cin>>secondo;
            if(primo>secondo) cout<<"il primo e\' maggiore";
            else cout<<"il secondo e\' maggiore";
}

int main(int argc, char**argv){
select();
return 0;
}
```

Esercizio complementare: Usando l'ambiente DevC++, creare il progetto, in modalità console, con nome "costrutto selettore esempio".

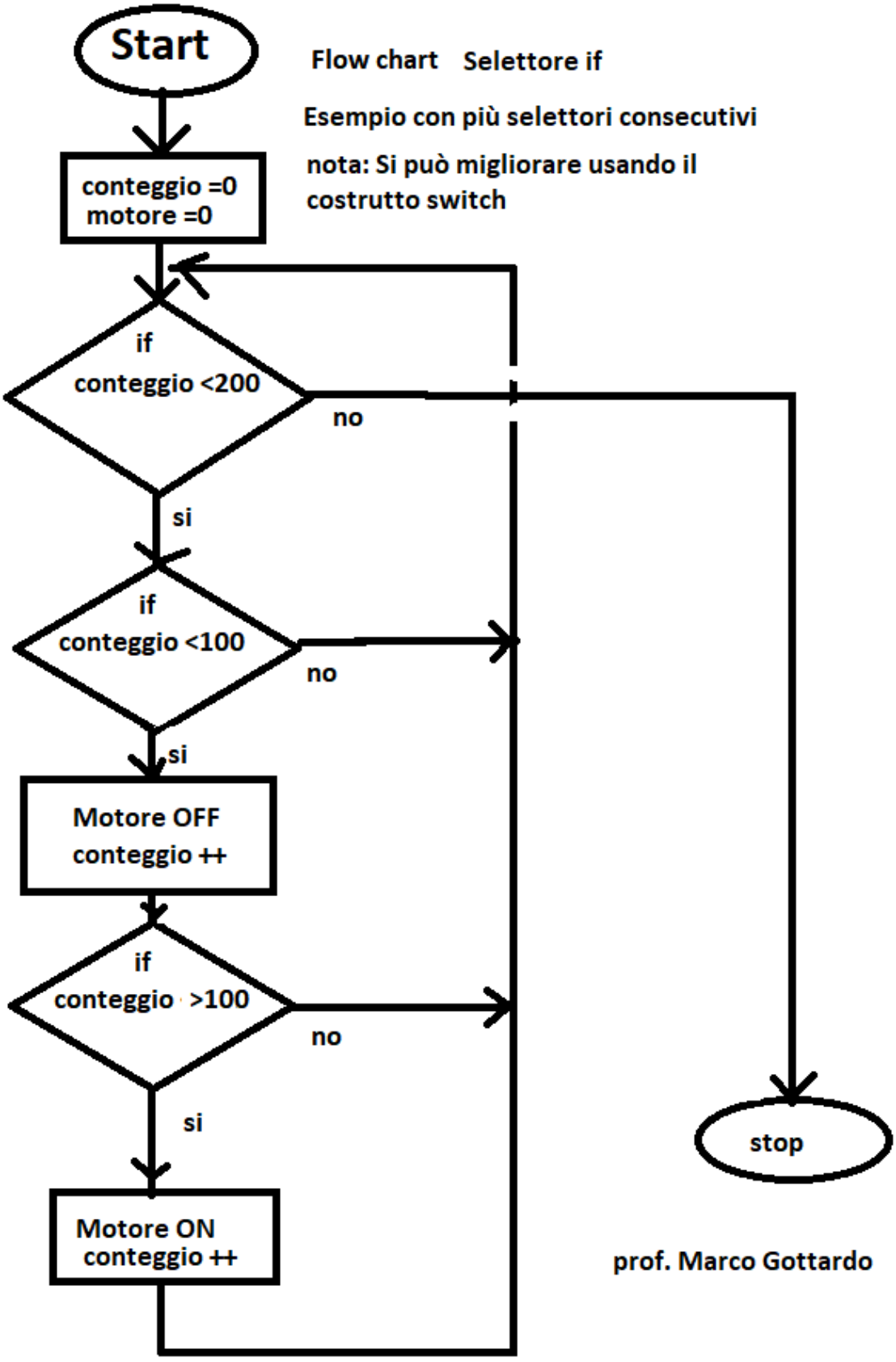
Eseguire la compilazione e il test iterativo.

Un programma si dice iterativo quando Runtime (durante l'esecuzione) vi è un verbose tra la macchina e l'utente, ad esempio richiede l'inserimento dei dati da elaborare,

In sostanza è richiesto l'utilizzo dei costrutti cout<< e cin>> con la relativa abilitazione dell'ambiente e librerie.

using namespace std; //consente l'utilizzo di cin e cout con il compilatore C++

Il diagramma di flusso che descrive il programma sopra mostrato è mostrato nell'immagine successiva.



Flow chart Selettore if

Esempio con più selettori consecutivi

nota: Si può migliorare usando il costrutto switch

prof. Marco Gottardo

Selezione semplice e doppia

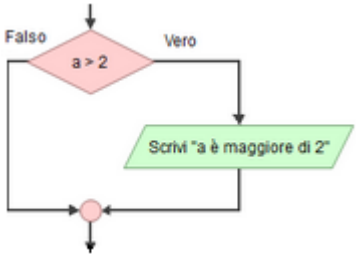
Esistono due tipi di costrutti utili a distinguere cosa eseguire all'interno di un gruppo di selezione, questi sono la selezione semplice e la selezione doppia.

L'istruzione di selezione semplice permette di far eseguire al calcolatore alcune istruzioni solamente quando il valore di una istruzione di test ha esito positivo nel caso contrario, in cui l'esito è negativo non esegue nessuna istruzione.

La selezione si dice doppia quando si prevede di compiere azioni sia nel ramo vero che nel ramo falso.

Vediamo due esempi:

Selezione semplice

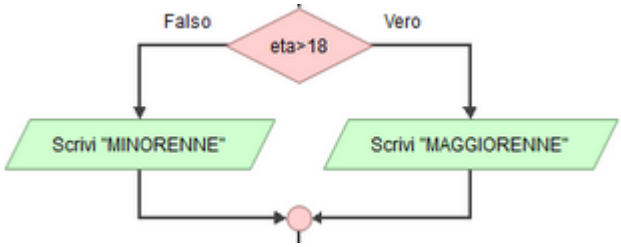


if (A>2) {

cout<<"A è maggiore di 2";

}

Selezione doppia



If (eta>18) cout<<"maggiorenne";

else cout<<"minorenne";

Selezione con blocchi di istruzioni

Risposta: Il costrutto IF può essere usato per lanciare in esecuzione una sequenza ovvero per chiamare una funzione che contiene questa sequenza.

Consideriamo un robot semovente, ovvero una piattaforma con le ruote come la sonda curiosity sulla superficie di Marte.

Vogliamo dare dei comandi a questa sonda che eseguano una specifica sequenza in base a ciò che inviamo.

```
#include <iostream>
#include <conio.h>

using namespace std;

//all'interno del corpo di ciascuna delle seguenti funzioni possiamo insrire una sequenza anche molto articolate di azioni, //ad esempio per l'accensione o spegnimento dei motori, il verso di rotazione, l'acquiszione dei sensori telemetrici, la //lettura degli encoder ecc.
void MoveForward(){cout<<"ok, i m moving forward";}
void MoveReverse(){cout<<"ok, i m moving Backward";}
void StopRobot(){cout<<"system is shutting down !";}

void selectMove(){
 char move;
 cout<<"Mars probe curiosity: command list\n";
 cout<<" F to move forward\n";
 cout<<" R to move Backward\n";
 cout<<" S to stop and shutdown the robot\n";
 move=getch();
 if (move=='F') MoveForward();
 if (move=='R') MoveReverse();
 if (move=='S') StopRobot();
}

int main (){
 selectMove();
 return 0;
}
```

Il costrutto selezione multipla

Il costrutto di selezione multipla è molto utile quando una variabile, ad esempio un carattere, proveniente da un dispositivo di trasmissione o comunicazione oppure dalla tastiera del PC può assumere svariati valori corrispondenti ad azioni diverse.

Prendiamo ad esempio un Robot semovente a cui vogliamo impartire i comandi di A->Marcia Avanti, I->Marcia Indietro, D-> Rotazione a destra, S-> Rotazione a sinistra.

Si presta bene alla risoluzione di questo problema il costrutto **switch – Case**.

In questo esempio lo abbiniamo alla funzione getch() che ha il prototipo in conio.h.

Questa acquisisce un carattere e permette di scaricarlo in una predefinita funzione char, in questo esempio abbiamo dichiarato char select;

Ogni caso si chiude con il comando break;

In ogni condizione testata si potranno mettere molte reazioni all'evento, come assegnazioni di variabili, calcoli, trasmissioni ecc, in questo caso faremo la chiamata a delle specifiche funzioni che azioneranno i motori nella corretta direzione e stamperanno a monitor la scritta che indica l'azione del robot.

```
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

void RobotAvanti(){
    cout<<"il robot si sposta in avanti \n";
}

void RobotIndietro(){
    cout<<"il robot si sposta all indietro \n";
}

void RobotDestra(){
    cout<<"il robot gira a destra \n";
}

void RobotSinistra(){
    cout<<"il robot gira a sinistra \n";
}

void select_move( ){
char select;
cout<<" digita A per avanti, I per indietro, D per destra, S per sinistra ";
```

```
select = getch(); //associamo un carattere che viene dalla tastiera a una variabile
switch (select){
case 'A':  //caso robot si sposta in avanti
RobotAvanti();
break;
case 'I': //caso robot si sposta all'indietro
RobotIndietro();
break;
case 'D':   //caso robot gira a destra
RobotDestra();
break;
case 'S':  //caso robot gira a sinistra
RobotSinistra();
break;
}
}

int main(int argc, char** argv) {
      select_move( );
      return 0;
}
```

L’algebra della logica proposizionale

L'algebra delle proposizioni in matematica e in informatica, nella forma booleana, consente di descrivere delle affermazioni, da utilizzare all’interno di un algoritmo, ad esempio come funzione di test per entrata o uscita da un loop, sotto forma di proposizioni logiche. Una proposizione logica può essere vera o falsa. La logica binaria (TRUE o FALSE) consente di utilizzare l'algebra delle proposizioni come strumento fondamentale dell'informatica come linguaggio dei computer. Nell'informatica le proposizioni logiche sono utilizzate per la progettazione degli algoritmi, sono rappresentate dalle variabili logiche ed elaborate dagli operatori logico-matematici in una procedura o in un programma informatico (linguaggio di programmazione). In un computer lo stato vero o falso della proposizione logica è realizzato con la presenza o l'assenza del segnale elettrico, questi due stati sono rappresentati dai simboli binari "uno" (1) e "zero" (0). Avendo due soli stati, la variabile logica è detta anche variabile di commutazione (algebra di commutazione).

Le tabelle di verità sono fondamentali per l’uso dell’algebra combinatoria, e queste sono principalmente la funzione NOT con un unico operando, la OR e la AND, con 2 o più operandi, in cui la OR è la somma logica, almeno un termine a 1 porta il risultato globale a 1, mentre la AND è il prodotto logico in cui è necessario che tutti i termini siano a 1 per avere l’uscita a 1.

Esiste la funzione OR-esclusivo, detto Ex-Or, che implementa la somma algebrica che tiene conto degli eventuali riporti. Secondo questo concetto per gli operatori A e B, entrambi a 1 il risultato è zero “0” in quanti si ha il riporto sulla cifra successiva.

In C e in C++ gli operatori logici OR sono | (la barretta verticale) e il & per la funzione AND.

I connettivi logici: AND, OR, NOT

In tutti i linguaggi di programmazione sono necessarie decisioni di tipo vero o falso che derivano da elaborazioni di stati messi tra loro in serie (costrutto AND), in parallelo (costrutto OR) oppure in negazione dello stato logico, (costrutto NOT).

La gerarchia degli operatori booleani corrisponde all'ordine di priorità con cui gli operatori di una espressione logica vengono eseguiti ad esempio all'interno di una espressione di test per l'uscita o l'ingresso in loop. A parità di condizioni, in una espressione logica gli operatori logici con priorità superiore saranno elaborati prima degli altri. L'operatore logico con priorità più alta è l'operatore NOT, quindi AND e infine OR. Gli operatori logici con priorità inferiore sono OR, XOR e NOR.

La sintassi di molti linguaggi di programmazione usano i simboli & per AND, | per OR e ! per la negazione.

In alcuni linguaggi, orientati all'embedded, è necessario distinguere se l'OR, AND, è relativo alla singola variabile booleana oppure a l'intero registro di 8 o più bits, come avviene ad esempio quando si vuole distinguere lo stato dei bit che contengono un numero intero.

Questa situazione è detta bitwise e si abilita con il simbolo ripetuto due volte, ad esempio "&&" oppure "||". Questa evenienza si incontra con il compilatori XC rilasciati da Microchip per la programmazione dei Microcontrollori PIC.

Nel caso, ad esempio, in cui il programma realizzi il controllo di un robot, gli stati logici in input possono essere creati da un insieme di sensori telemetrici (distanza dagli ostacoli), estensimetri (sforzo sugli organi di presa, ad esempio le mani) ecc.

supponiamo che la marcia in avanti del robot sia concessa se i due sensori frontali risultano liberi da ostacoli.

Creiamo un progetto C++ che chiamiamo Robot semovente.

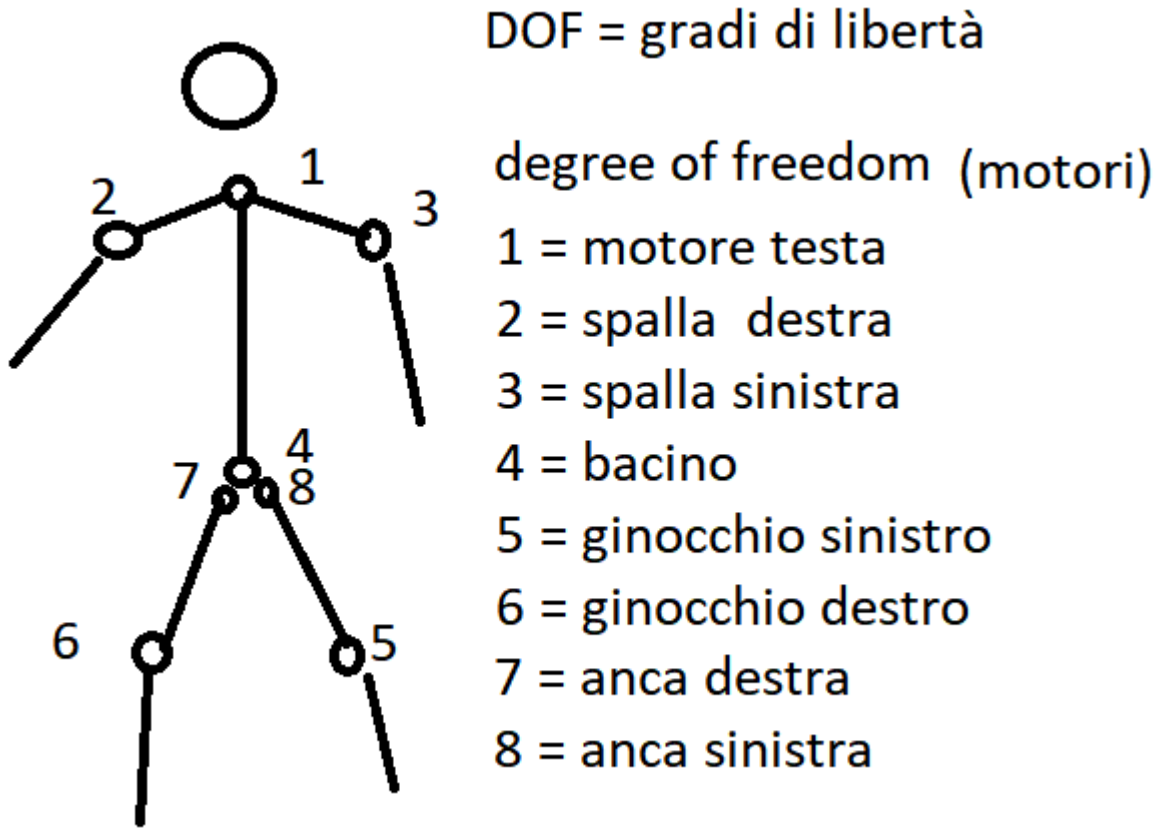
```
#include <iostream>
using namespace std;
void movimentoRobot(){
    // FW,RW,Sx,Dx telemetric sensors
    bool FW;
    bool RW;
    bool Sx;
    bool Dx;
    //Move_FW,Move_RW,Move_Sx,Move_Rx, movements command
    bool Move_FW,Move_RW,Move_Sx,Move_Dx;
    // Sensors acquire, actually forced by programmer
    // Sensor operative if TRUE
    // Sensor detect obstacle if turn to FALSE
    // All sensors false means unfunctions
```

```
FW=1;
RW=1;
Sx=0;
Dx=1;

//test costrutto AND dei sensori
//se tutti operativi danno segnale TRUE
//si testa con AND logico, se il risultato
//logico combinatorio dell AND "&" vale 1
//allora il Robot può muoversi.
// se il risultato è FALSE si tiene fermo il
//robot per questioni di sicurezza.

if (FW & Sx & Dx & RW) cout<<"Sensori operativi\n";
if (!FW & !Sx & !Dx & !RW) cout<<"Sensori tutti rotti\n";

// Eseguiamo il test dei comandi di rotazione a Dx o SX
//Utilizzando il comando OR che si realizza con il simbolo
// barretta verticale "|" tasto sotto esc

if (Sx | Dx & FW & RW) {  //esempio di uso del costrutto logico OR di segnali bool
cout<<"Ricevuto un comando di rotazione\n";  //non discrimina da che parte ruotare
        if (!Sx & Dx) {  //verifica se segnale da sensore sinistro
        cout<<"Il robot ruota verso sinistra\n";
        Move_Sx=1;
        Move_Dx=0;
        }
        if (Sx & !Dx){ //verifica se segnale da sensore di destra
        cout<<"Il robot ruota verso destra \n";
        Move_Sx=0;
        Move_Dx=1;
        }
}

cout<<"Muovi il Meccanoid F-> avanti, R -> indietro";
cin>>Move_FW;
if (Move_FW){
        if (FW & Sx & Dx) {          //use of AND condition
        cout<<"comanda Meccanoid avanti\n";
        }
}

cin>>Move_RW;
if (Move_RW){
        if (FW & Sx & Dx) {          //use of AND condition
        cout<<"comanda Meccanoid indietro\n";
        }
```

```
    }

    cin>>Move_Sx;
    if (Move_Sx){
            if (FW & Sx & Dx) {         //use of AND condition
            cout<<"comanda Meccanoid sinistra\n";
            }
    }

    cin>>Move_Dx;
    if (Move_Dx){
            if (FW & Sx & Dx) {         //use of AND condition
            cout<<"comanda Meccanoid destra\n";
            }
    }


}
int main(int argc, char** argv) {
    movimentoRobot();
    return 0;
}
```

Vediamo altri interessanti esempi di applicazione degli operatori logici applicati alla robotica.

Consideriamo il robot umanoide schematizzato e semplificato mostrato nella figura seguente:



Ogni DOF, grado di libertà, è un servomotore, il cui movimento viene impartito dal processore elaborando valori numerici che costituiscono angoli.

Al momento, il programma sarà semplificato, ovvero non avrà i sensori di ostacolo ma solo eseguirà i comandi di aumento e diminuzione dell'angolo del segmento (ad esempio braccio destro).

Vediamo due semplici esempi di soluzione per il controllo dei segmenti.

Il primo esempio mostra come definire delle strutture di dati "struct" o "record" e come accedere ai sui campi.

```
#include <iostream>

using namespace std;


      struct Robot{  //dichiarazione del nome della struct, nuovo tipo di dato
      int speed; //definizione dei campi della struct come singole variabili
      double number;
      bool collo;
      bool spalla_destra;
      bool spalla_sinistra;
      bool bacino;
      bool ginocchio_sinistro;
      bool ginocchio_destro;
            )//inizializza i campi della struct
```

```
            Robot() : speed(42), number(4.1), collo(true    {
    }
}miorobot; //crea una variabile del tipo definito dalla struct


void RobotMove(){  //funzione per muovere il robot
      char op;
      cout << "Premere O per ON dei motori ->\t";
      cin >> op;
      if (op == 'O'){
            miorobot.bacino = 1;
            miorobot.collo = 1;
            miorobot.ginocchio_destro = 1;
            miorobot.ginocchio_sinistro = 1;
            miorobot.spalla_destra = 1;
            miorobot.spalla_sinistra = 1;
      }
}

int main(int argc, char** argv) {
      int numero;
numero = Robot().speed;
cout<<numero;
cout << "\nStato dei motori:" << endl;
cout << Robot().bacino << endl;
cout << Robot().collo << endl;
cout << Robot().ginocchio_destro << endl;
cout << Robot().ginocchio_sinistro << endl;
cout << Robot().spalla_destra << endl;
cout << Robot().spalla_sinistra << endl;
      return 0;
}
```

Vediamo una versione del comando del Robot in cui si predispone all'acquisizione dei sensori angolari "encoder" e si realizzano le funzioni di accensione e spegnimento del sistema.

```
#include <iostream>
using namespace std;

void splash(){
    cout << "**********************************" << endl;
    cout << "** prof Marco Gottardo PhD      **" << endl;
    cout << "**    09 Novembre 2020          **" << endl;
    cout << "**   Controllo robot  3 DOF     **" << endl;
    cout << "**   SIIC prima   Liceo 2020    **" << endl;
    cout << "**********************************" << endl;
    cout << endl;
}

encoderAcquire(){ //lettura impulsi dagli encoder
//gli encoder li mettiamo su 3 variabili intere (integer) nel range -32768 fino a +32767
int alfa_quota = 0;
int beta_quota = 0;
```

```
int gamma_quota = 0;
int dofA_plus,dofA_minus;
int dofB_plus,dofB_minus;
int dofC_plus,dofC_minus;
char on_off;
int onetime = 0;
int oneBye = 0;
      cout << " Do you want to switch ON the Robot y or n  -> ";
      cin>>on_off;
      while (on_off== 'y'){
      if (on_off== 'y' & onetime <1) {
      cout << " Robot engage and ready ";
      if(dofA_plus==1){
                  cout << " Alza la spalla\n ";
      }
      if(dofA_minus==1){
      }
         cout << " Abbassa la spalla\n ";
      if(dofB_plus==1){
                  cout << " Alza il braccio\n ";
      }
      if(dofB_minus==1){
                  cout << " Abbassa il braccio\n ";
      }
      if(dofC_plus==1){
                  cout << " Alza il polso\n ";
      }
      if(dofC_minus==1){
                  cout << " Abbassa il polso\n ";
      }
      }
       else {
                  if(oneBye<1){
                    cout << " Robot standby " <<endl;
           oneBye++;
                     }

        }
        cout << " Do you want to switch OFF the Robot q to quit  -> ";
        cin>>on_off;
        onetime++;  //incremento automatico di una variabile del valore 1
      }
}

motor_move(){
}
```

```
void byebye(){
    cout << "*****************************" << endl;
    cout << "**      Robot shutting down         **" << endl;
    cout << "** Controllo robot  3 DOF  OFF   **" << endl;
    cout << "**   SIIC prima   Liceo 2020         **" << endl;
    cout << "*****************************" << endl;
    cout << endl;
}

int main(int argc, char** argv) {
    splash();
    encoderAcquire();
    encoderAcquire();
    motor_move();
    byebye();
    return 0;
}
```

Il costrutto iterazione

Il **costrutto iterativo** detto anche ciclo viene utilizzato quando un istruzione, o più istruzioni devono essere eseguite più di una volta. Ne è un esempio una variante del programma risolutivo per la torre di Hanoi, il quale richiamo più volte luna delle tre funzioni principali, SxToSwap, SwapToDx, SwapToSx, e poche altre. Per risolvere problemi, come ad esempio la torre di Hanoi, in cui un'azione o un gruppo di azioni devono essere ripetute in un certo periodo di tempo conviene usare questo tipo di programmazione.

L'obbiettivo dell'iterazione è quello di convergere verso una soluzione utilizzando un numero, anche non predefinito, di ripetizioni di una funzione o di un'istruzione.

Nella programmazione strutturata vengono utilizzati due costrutti.

- costrutto iterativo pre-condizionale o costrutto iterativo con controllo in testa
- costrutto iterativo post-condizionato

Affinché si possa realizzare una iterazione senza porre l'elaboratore in un loop è necessario che siano soddisfatte le tre precondizioni.

1. Il test di verifica della condizione di entrata\uscita dal ciclo
2. Il gruppo di istruzioni e la singola istruzione da iterare queste vengono eseguite a ogni iterazione.
3. Tra le istruzioni da iterare deve essercene una che possa modificare l'esito del test di che proietti l'esecuzione fuori dal loop.
4. Le variabili devono avere dei valori che pur variando nelle iterazioni presentino alla funzione di test valori sensati per il range di utilizzo nello specifico algoritmo.

In bibliografia è comunque possibile trovare anche la seguente definizione per l'iterazione, divisa in due possibili tipi di realizzazione:

- Iterazione definita
- Iterazione condizionata

Iterazione definita: si conosce a priori quante volte l'azione elementare (o il blocco di istruzioni) verrà ripetuta. L'iterazione termina quando si è raggiunto il numero prefissato di ripetizioni.

Iterazione condizionata: Non si conosce a priori quante volte l'azione elementare (o il blocco di istruzioni) verrà ripetuta. L'iterazione termina quando si verifica una certa condizione.

- pre-condizionata: La condizione viene controllata prima delle istruzioni che quindi, potrebbero non essere mai eseguite.
- post-condizionata: La condizione viene controllata dopo le istruzioni, che quindi vengono eseguite almeno una volta.

In linguaggio C e C++, come in molti altri linguaggi, l'iterazione a dimensione predefinita si può realizzare utilizzando il costrutto "for to do", mentre quella a dimensione non predefinita si può realizzare con il costrutto "While do" oppure "do while" a seconda che il test della variabile di controllo avvenga in testa "While do" oppure in coda "do while".

Nel primo caso, il "while do" l'esecuzione del test prima di entrare nel blocco da eseguire potrebbe risultare subito falso e di conseguenza non si esegue neppure una iterazione, mentre nel secondo caso si esegue sempre almeno la prima iterazione dato che il test viene seguito alla fine di questa.

Nel caso di implementazione tramite il ciclo for bisogna tenere presente che nella versione del compilatore detto C99 e quelli successivi la variabile di indice può essere dichiarata direttamente all'interno dei parametri dello stesso for, ma in questo caso non sarà riconosciuta al di fuori del corpo del for, quindi solo tra le prentesi graffe aperta e chiusa.

Ne consegue che per la variabile index è locale in questo caso:

for (int index=0;index<20;index++){ funzioni da iterare}

mentre in questo caso è globale ed utilizzabile anche nella funzione in ci è contenuta.

int index;

for (index=0;index<20;index++){ funzioni da iterare}

Rimane possibile l'uso del iteratore for anche per la costruzione di loop infiniti, molto utili ad esempio nella programmazione enbedded in cui l'automa rimane in esecuzione del programma dopo che questo sia stato avviato, ma senza ripetere il blocco di inizializzazione delle variabili.

La costruzioni di un loop infinito può avvenire ad esempio con la sintassi for(;;) che risulta formalmente equivalente a while (1){ istruzioni in loop;}

Esempio di programma per la stampa di caratteri estratti dalla tabella ASCII compresi tra un minimo e un massimo:

```
#include <iostream>
#include <conio.h>

using namespace std;

void stampaChar(){
        char a;
        cout<<"Letters in range from A to Z are: \n";
        for (a='A'; a<='Z'; a++)
   cout<<a;
}


int main(int argc, char** argv) {
        char answer;
        cout<<"prof. Marco Gottardo Ph.D.  14 marzo 2021 \n";
        cout<<"vuoi stampare i caratteri tra A e Z ?\n";
        cout<<"press y for yes or n for skip -> \n";
        answer=getch();
        if(answer=='y')
        stampaChar();
        else cout<<"OK -> Bye Bye !!!";
        return 0;
}
```

Iterazione con controllo in testa

```
#include <iostream>
#include <conio.h>
using namespace std;

void trial (){
    int a;
    cout << "Inserire un valore ->\t";
    cin >> a;
    while (a == 2){
            system ("cls");
            cout << "Il valore inserito equivale a 2" << endl;
            cout << "\n\nInserire un valore ->\t";
            cin >> a;
            if (a != 2){
                    break;
            }
    }
}
```

```
int main (){
    trial();
    return 0;
}
```

Iterazione con controllo in coda

```
#include <iostream>
using namespace std;

void trial(){
    int a =2;
    while ( a == 2){
            cout << "Inserire un valore ->\t";
            cin >> a;
            if (a != 2){
                    break;
            }
    }
}

int main (){
    trial();
    return 0;
}
```

Iterazione a conteggio

iterazione a conteggio (usando un ciclo while)

```
#include <iostream>
using namespace std;


void trial (){
    int a, variabileConteggio;
    while (variabileConteggio < 3){
            cout << "Inserire un valore ->\t";
            cin >> a;
            variabileConteggio ++;
            if (variabileConteggio >= 3){
                    break;
            }
    }
}

int main (){
```

```
	trial();
	return 0;
}
```

OPPURE (usando un ciclo for)

```
#include <iostream>
#include <conio.h>
using namespace std;


void trial (){
	for (int variabileConteggio; variabileConteggio < 3; variabileConteggio ++){
		int a;
		cout << "Inserire un valore ->\t";
		cin >> a;
	}
}

int main (){
	trial();
	return 0;
}
```

Appendice: Programmi estesi dalla prima alla quinta liceo

Linee generali e competenze

L'insegnamento dell'informatica deve contemperare diversi obiettivi: comprendere i principali fondamenti teorici delle scienze dell'informazione, acquisire la padronanza di strumenti dell'informatica, utilizzare tali strumenti per la soluzione di problemi significativi in generale, ma in particolare connessi allo studio della altre discipline, acquisire la consapevolezza dei vantaggi e dei limiti dell'uso degli strumenti e dei metodi informatici e delle conseguenze sociali e culturali di tale uso. Al termine del percorso liceale lo studente dovrà acquisire la conoscenza e la padronanza dei più comuni strumenti di software per il calcolo, la ricerca e la comunicazione in rete, la comunicazione multimediale , l'acquisizione e l'organizzazione dei dati applicandoli in una vasta gamma di situazioni, ma soprattutto nell'indagine scientifica, scegliendo di volta in volta lo strumento più adatto. Verranno proposti problemi significativi che consentano un collegamento tra l'informatica e le altre discipline STEM allo scopo di far acquisire al discente un ulteriore strumento di lavoro. Il discente dovrà essere consapevole delle ragioni che hanno prodotto lo sviluppo scientifico e tecnologico nel tempo in relazione ai bisogni e alle domande di conoscenza dei diversi contesti, ponendo attenzione alle conquiste scientifiche, in particolare quelle più recenti. L'allievo dovrà anche comprendere il ruolo della tecnologia, come mediazione tra scienza e vita quotidiana e saper utilizzare gli strumenti informatici in relazione all'analisi dei dati e alla modellizzazione di specifici problemi scientifici. Gli obiettivi didattici ed educativi possono essere pertanto espressi genericamente come segue:

- ✓ Acquisire un metodo di studio autonomo e flessibile, che consenta di condurre ricerche e approfondimenti personali;
- ✓ Essere consapevoli della diversità dei metodi utilizzati ed essere in grado valutare i criteri di affidabilità dei risultati in essi raggiunti;
- ✓ Saper compiere le necessarie interconnessioni tra i vari metodi ed i vari contenuti;
- ✓ Acquisire l'abitudine a ragionare con rigore logico, ad identificare i problemi e a individuare possibili soluzioni;
- ✓ Saper collocare storicamente lo sviluppo delle varie invenzioni tecnologiche;
- ✓ Essere in grado di utilizzare criticamente strumenti informatici e telematici nelle attività di studio e di approfondimento;
- ✓ Comprendere la valenza metodologica dell'informatica nella formalizzazione e modellizzazione dei processi complessi e nell'individuazione di procedimenti risolutivi;
- ✓ Acquisire padronanza del linguaggio tecnico, logico e formale della disciplina - Saper utilizzare strumenti di calcolo e di rappresentazione per la modellizzazione e la risoluzione di problemi;
- ✓ Saper cogliere la potenzialità delle applicazioni e delle invenzioni tecnologiche nella vita quotidiana;

- ✓ Comprendere il ruolo della tecnologia come mediazione fra scienza e vita quotidiana;
- ✓ Saper utilizzare gli strumenti informatici in relazione all'analisi dei dati e alla modellizzazione di specifici problemi scientifici ed individuare la funzione dell'informatica nello sviluppo scientifico;
- ✓ Acquisire la consapevolezza dei vantaggi e dei limiti dell'uso degli strumenti e dei metodi informatici e delle conseguenze sociali e culturali di tale uso - Comprendere la struttura logico-funzionale della struttura fisica e del software di un computer e di reti locali, tale da consentirgli la scelta dei componenti più adatti alle diverse situazioni e le loro configurazioni, la valutazione delle prestazioni, il mantenimento dell'efficienza;
- ✓ Saper collegare in modo sistematico l'uso di strumenti e la creazione di applicazioni ai concetti teorici ad essi sottostanti.

verifiche e valutazioni

Nell'ambito della didattica per competenze la valutazione non è intesa come un momento isolato, bensì diventa un processo continuo, nel corso del periodo didattico, dell'anno scolastico e di tutto il percorso dei 5 anni della secondaria superiore con dei momenti particolarmente significativi in occasione della fine del primo e secondo biennio e della conclusione del quinto anno

Le modalità di valutazione adottate sono basate soprattutto sulla verifica della esistenza o meno nello studente della connessione tra il possesso delle conoscenze e la effettiva capacità di selezionarle, elaborarle, interpretarle criticamente e sistemarle. Particolare importanza sarà data all'impegno individuale che, a prescindere dalla propensione dello studente alla disciplina, verrò inteso come disponibilità al confronto ed assunzione di responsabilità nella conduzione del lavoro scolastico.

Pertanto, in relazione agli obiettivi enunciati per i singoli nuclei, si osserverà la capacità dell'allievo di:

- ✓ conoscere i contenuti dei diversi nuclei tematici;
- ✓ analizzare un quesito e rispondere in forma sintetica;
- ✓ prospettare soluzioni, verificarle e formalizzarle.

Si osserverà anche l'aderenza ad alcuni obiettivi trasversali, fra i quali:

- ✓ leggere e interpretare un testo di carattere scientifico;
- ✓ comunicare e formalizzare procedure;
- ✓ rappresentare e convertire oggetti matematici;
- ✓ rielaborare in modo personale e originale i contenuti;
- ✓ partecipare in modo costruttivo e critico alle lezioni.

In ogni verifica scritta verranno indicati i criteri di attribuzione del punteggio i quali si rifanno alla griglia di valutazione dipartimentale. Il punteggio verrà poi trasferito in un voto in decimi in base ad una tabella corrispondente, non perfettamente proporzionale ma ponderata sulle competenze. Il momento della consegna e attribuzione del voto sarà sempre un occasione di discussione degli argomenti proposti, particolarmente formativo per il miglioramento dell'apprendimento e del risultato finale.

Verranno effettuate prove tra le seguenti tipologie a seconda delle esigenze e delle strategie necessarie per il consolidamento degli argomenti proposti:

Prove strutturate su una o più unità didattiche

Prove semi strutturate su una o più unità didattiche.

Ci si riserva la possibilità di effettuare dei colloqui orali. Tali colloqui verranno valutati non solo per quanto riguarda la conoscenza e la comprensione degli argomenti, ma anche per la chiarezza dell'esposizione e la proprietà di linguaggio.

Per la valutazione finale di ciascun alunno si terrà conto della valutazione dei compiti scritti e dei colloqui orali, delle competenze informatiche acquisite, del comportamento globale, della costanza nel lavoro pomeridiano e laboratoriale e della puntualità nelle consegne.

Particolare rilievo sarà dato alle capacità in termini di autonomia nello svolgimento dei compiti a carattere laboratoriale nonché ai metodi di risoluzione dei problemi posti.

Attività di recupero

Verranno effettuati, se necessari, da parte dei docenti recuperi in itinere per consentire agli allievi un più agevole approccio con la disciplina.

Modalità e strumenti

La disciplina sarà trattata con varie metodologie didattiche:

Lezioni frontali: il docente descrive con l'aiuto degli strumenti disponibili (lavagna, computer, videoproiettore, dispense, ecc) gli aspetti importanti dell'argomento trattato, non limitandosi alla semplice esposizione, ma stimolando la partecipazione costruttiva della classe e privilegiando il metodo deduttivo.

Discussione in classe: si creano situazioni di confronto su tematiche inerenti gli argomenti trattati al fine di far emergere problemi, dubbi e congetture utili al rafforzamento dell'azione formativa.

Esercitazioni pratiche e in laboratorio: dopo aver illustrato gli aspetti teorici dell'argomento, vien e assegnato agli allievi un lavoro di progettazione e realizzazione. Grazie all'attività di laboratorio vengono messe alla prova le abilità progettuali e organizzative acquisite.

Lavoro di gruppo: al fine di stimolare la cooperazione ed il confronto Valutazione (tipologia di prove, criteri di valutazione e griglie) - Per quanto riguarda i compiti scritti (almeno 6 in totale) si terrà conto della completezza dell'elaborato, della strategia risolutiva, del calcolo e dell'esposizione formale.

Problem Posing e Problem Solving: Particolare importanza al lavoro di gruppo verrà data nell'ambito del secondo biennio e del quinto anno con la pratica del Problem Posing e svolgimento del lavoro in gruppo mediante la tecnica del Problem Solving. Data la complessità dei problemi da affrontare i lavori verranno proposti in classe dal docente mediante la presentazione del "caso di studio" e verranno risolti dagli studenti divisi in gruppi anche attraverso dei lavori di ricerca della soluzione.

Programmazione modulare

Il dipartimento di Matematica e Scienza ritiene che la programmazione modulare sia lo strumento efficace per conseguire le finalità formative precedentemente illustrate e per costruire i percorsi formativi disciplinari, che traducano nella successione dei moduli i nuclei fondanti precedentemente individuati e stabiliscano le competenze da accertare.

L'organizzazione modulare flessibile della didattica è una strategia formativa altamente strutturata che prevede l'impiego di segmenti unitari chiamati moduli. Il *modulo* è una parte significativa, omogenea ed unitaria di un più esteso percorso formativo, disciplinare, o pluri/multi/inter disciplinare (con la distinzione nominale nel caso di una sola disciplina di "modulo debole", nel caso di più discipline di "modulo forte") la cui finalità è il raggiungimento di obiettivi.

Il modulo può essere disinserito facilmente, modificato nei contenuti e nella durata, sostituito, mutato di posto nella struttura curriculare sequenziale iniziale. I motivi che hanno portato alla scelta dei moduli nella programmazione sono:

- ✓ *L'individualizzazione dell'insegnamento*: l'assemblaggio di moduli consente di operare una didattica vicina alle esigenze di ciascun allievo;
- ✓ *La quantificazione della competenze acquisite*: i moduli possono rappresentare l'unità di misura delle competenze acquisite;
- ✓ *L'organizzazione razionale delle attività*: i moduli, e ancor più le unità didattiche di cui essi sono costituiti, consentono di operare su segmenti curricolari brevi in modo da ridurre gli insuccessi e i fallimenti formativi.

Nelle pagine seguenti è riportata la scansione in moduli delle attività di programmazione relative al curricolo di Informatica.

Programmazione primo biennio

Obiettivi Specifici di Apprendimento

Nel primo biennio verranno usati gli strumenti di lavoro più comuni del computer insieme ai concetti di base ad essi connessi. Verranno introdotte le caratteristiche architetturali di un computer: i concetti di hardware e software, una introduzione alla codifica binaria presenta i codici ASCII e Unicode, gli elementi funzionali della macchina di Von Neumann: CPU, memoria, dischi, bus e le principali periferiche. Verrà spiegato il concetto di sistema operativo, le sue funzionalità di base e le caratteristiche dei sistemi operativi più comuni. Verrà introdotto il concetto di processo come programma in esecuzione, illustrato il meccanismo base della gestione della memoria e le principali funzionalità dei file system.

Verranno introdotti gli elementi costitutivi di un documento elettronico e i principali strumenti di produzione. Occorre partire da quanto gli studenti hanno già acquisito nella scuola di base per far loro raggiungere la padronanza di tali strumenti, con particolare attenzione al foglio elettronico.

Verranno introdotti la struttura e i servizi di Internet. Insieme alle altre discipline si condurranno gli studenti a un uso efficace della comunicazione e della ricerca di informazioni, e alla consapevolezza delle problematiche e delle regole di tale uso.

Verranno introdotti i principi alla base dei linguaggi di programmazione, illustrate le principali tipologie di linguaggi e il concetto di algoritmo. Verrà sviluppata la capacità di implementare un algoritmo in pseudo-codice o in un particolare linguaggio di programmazione, di cui si introdurrà la sintassi.

Classe prima

Modulo 1: Il Sistema Computer
Obiettivi:

Far conoscere agli studenti l'architettura di un elaboratore ed il suo principio di funzionamento.
Competenze:

- Riconoscere le caratteristiche logico funzionali di un computer;
- Conoscere il ruolo strumentale svolto nei vari ambiti(calcolo, elaborazione, comunicazione);
- Comprendere i principi di funzionamento di uno smart device.

Contenuti:

- Architettura del computer;
- Hardware e software;
- Il case e la scheda madre;
- L'unità centrale di elaborazione;
- La memoria centrale;
- Come si misura la memoria: bit e byte ;
- La memoria di massa;
- Le periferiche: input, output e input/output ;
- Tipi di computer.

Modulo 2: Il Sistema Operativo
Obiettivi:

Far comprendere le funzionalità e l'importanza del sistema operativo nel funzionamento del pc e di tutti gli smart device.
Competenze:

- Riconoscere e utilizzare le funzioni di base di un sistema operativo;
- Interagire con gli elementi del Sistema Operativo e personalizzarli;
- Operare su file e cartelle per strutturare e organizzare l'archivio

Contenuti:

- Elementi e caratteristiche dell'interfaccia grafica del Sistema Operativo;
- L'avvio del computer: il bootstrap;
- Il desktop e le icone;
- Gli elementi delle finestre;
- File e cartelle;

- Estensione dei file;
- Il sistema di archiviazione;
- Il pannello di controllo;
- I sistemi operativi per i vari dispositivi.

Modulo 3: La codifica dell'informazione

Competenze:

- Come si rappresentano i dati all'interno del computer;
- Convertire un numero da un sistema di numerazione ad un altro;
- Rappresentare i dati alfanumerici.

Contenuti:

- La rappresentazione delle informazioni;
- Dato e informazione;
- Il codice;
- Codifica e decodifica dell'informazione;
- Codifica di informazioni e dati nel PC;
- I sistemi di numerazione;
- Il sistema di numerazione binario;
- Il sistema di numerazione esadecimale;
- Conversione da binario a decimale e viceversa;
- Conversione da decimale a esadecimale e viceversa;
- Conversione da esadecimale a binario e viceversa;
- Rappresentazione delle informazioni alfanumeriche.

Modulo 4: Elaboratore Testi

Competenze:

- Creare, salvare, aprire, modificare, correggere, stampare e chiudere un file;
- Applicare le procedure operative per la formattazione di base del testo;
- Formattare i documenti con elenchi, bordi e sfondi;
- Inserire e gestire : oggetti grafici, immagini, forme e caselle di testo.

Contenuti:

- Il word processor;
- Caratteristiche, funzionalità del word processor;
- Procedure per creare, archiviare, aprire, controllare un documento e stamparlo;
- Modalità operative per la formattazione base: margini, carattere e paragrafo;
- Tecniche per applicare elenchi, bordi e sfondo;
- Modi di inserire e gestire oggetti grafici e immagini;
- Distribuire il testo in colonne;
- Inserire interruzioni di colonna e di sezione;
- Fissare le tabulazioni;
- Inserire le tabelle e intervenire su righe e colonne.

Modulo 5: il Foglio Elettronico

Competenze:

- Creare, salvare, aprire, modificare e chiudere una cartella di lavoro;
- Eseguire semplici calcoli e espressioni con gli operatori matematici;
- Assegnare diversi formati numerici e dimensionare righe e colonne;
- Allineare i dati e applicare bordi, sfondi e stili di cella;
- Gestire le opzioni per impostare la pagina e i parametri di stampa del foglio di lavoro;
- Eseguire calcoli con le funzioni .

Contenuti:

- Il foglio elettronico;
- Caratteristiche e funzionalità del foglio elettronico;
- Definizione di cella, zona, etichetta, valore e formula;
- Struttura di una formula e i simboli degli operatori matematici;
- I diversi formati numerici e le loro proprietà;
- Tecniche per formattare il foglio di lavoro;
- Visualizzazioni- modalità stampa del foglio di lavoro;
- Eseguire calcoli con il foglio elettronico;
- Creare grafici e operare con fogli e riferimenti;
- Funzioni matematiche;
- Funzioni logiche.

Classe seconda

Modulo 1: L'informatica e il Problem Solving

Competenze:

- Analizzare,un problema e trovare la strategia risolutiva;

Contenuti:

- L'informatica e il trattamento delle informazioni;
- I problemi e il problem solving:la strategia risolutiva;
- Il problem solving ;
- L'analisi della formulazione dei problemi;
- La modellizzazione del problema;
- I metodi per trovare la strategia risolutiva;
- Risolutore ed esecutore.

Modulo 2: Dal Problema all'Algoritmo

Competenze:

- Analizzare, risolvere problemi e codificarne la soluzione con il linguaggio degli algoritmi

Contenuti:

- Costruire strategie risolutive non ambigue.
- Azioni e istruzioni.
- Il concetto di algoritmo.
- Rappresentazione degli algoritmi.
- I diagrammi a blocchi.
- Lo pseudolinguaggio.
- Rappresentazione di variabili e costanti.
- I linguaggi di programmazione.
- Il problem solving applicato alla programmazione
- analisi di un problema
- progettazione di un algoritmo
- codifica di un algoritmo

Modulo 3: Algoritmi con la programmazione strutturata

Competenze:

Formalizzare la soluzione del problema con le regole della programmazione strutturata.

Contenuti:

- Le istruzioni di un algoritmo;
- Le strutture di controllo;
- La sequenza;
- La selezione;
- L'algebra booleana e il suo ruolo nella programmazione strutturata.

Modulo 4: Fondamenti di Teoria dei Linguaggi

Competenze:

Riconoscere le differenze fra linguaggi naturali e linguaggi formali.

Riconoscere le caratteristiche di un linguaggio di programmazione.

Contenuti:

- Cosa è un paradigma di programmazione;
- Come lavorano i compilatori e gli interpreti;
- Il software;
- Linguaggi naturali e linguaggi formali;
- Linguaggi di programmazione a basso livello;
- Linguaggi di programmazione ad alto livello;
- I paradigmi di programmazione;
- I programmi traduttori: compilatori e interpreti.

Modulo 5: Studio dei linguaggi di programmazione

Studio delle istruzioni di base di uno o più linguaggi di programmazione (C++, Java, Visual Basic)

Competenze:

Saper realizzare il progetto software di un semplice algoritmo.

Seguire la logica del problem solving avviata con la progettazione.

Contenuti:

- Le variabili e le costanti;

- La dichiarazione;
- Il programma principale (main);
- Il costrutto di selezione;
- La selezione multipla;
- Il ciclo iterativo: incondizionato, pre-condizionato e post-condizionato
- Esercizi vari a contenuto interdisciplinare.

Programmazione secondo biennio

Obiettivi Specifici di Apprendimento

Nel secondo biennio si procede ad un allargamento della padronanza di alcuni strumenti e un approfondimento dei loro fondamenti concettuali. La scelta dei temi dipende dal contesto e dai rapporti che si stabiliscono fra l'informatica e le altre discipline. Sarà possibile disegnare un percorso all'interno delle seguenti tematiche:

- ✓ Strumenti avanzati di produzione dei documenti elettronici, linguaggi di markup (XML etc), formati non testuali (bitmap, vettoriale, formati di compressione), font tipografici, progettazione web.
- ✓ Introduzione al modello relazionale dei dati, ai linguaggi di interrogazione e manipolazione dei dati.
- ✓ Implementazione di un linguaggio di programmazione, metodologie di programmazione, sintassi di un linguaggio orientato agli oggetti.

Classe terza

Modulo 0: Studio dei linguaggi di programmazione
Studio delle istruzioni di base di uno o più linguaggi di programmazione (C++, Java, Visual Basic)

Competenze:
Saper realizzare il progetto software di un semplice algoritmo.

Seguire la logica del problem solving avviata con la progettazione.

Contenuti:

- Le variabili e le costanti;
- La dichiarazione;
- Il programma principale (main);
- Il costrutto di selezione;
- La selezione multipla;
- Il ciclo iterativo: incondizionato, pre-condizionato e post-condizionato
- Esercizi vari a contenuto interdisciplinare.

Modulo 1: Strutture dati e algoritmi per la loro gestione
Obiettivi:

Fornire agli studenti i concetti per consentire lo sviluppo di applicazioni che fanno uso di strutture di dati complesse (vettori, matrici, record, vettori di record) e gli strumenti per la loro elaborazione.

Competenze:

- Conoscere la differenza tra variabili semplici e variabili strutturate;
- Saper rappresentare i dati nelle idonee strutture;
- Conoscere gli algoritmi fondamentali per la gestione delle strutture dati;

Contenuti:

- Organizzare dati in vettori, matrici, record e vettori di record
- Applicare gli algoritmi di ordinamento;
- Applicare gli algoritmi di ricerca;
- Costruire programmi strutturati di una certa complessità.

Modulo 2: File
Obiettivi:

Fornire agli studenti la capacità di utilizzare all'interno di un programma dati memorizzati su memoria permanente

Competenze:

- Saper scrivere e leggere informazioni in un file

Contenuti:

- Memorizzare i dati su memoria permanente;
- Leggere i dati da memoria permanente;
- Effettuare semplici operazioni sui file

Classe quarta

2. Nozioni avanzate della programmazione ad oggetti

Competenze:

Contenuti

3. Modello dei dati

Competenze:

Contenuti:

4. Manipolazione di un database

Modulo 1: Concetti di base sulla OOP
Obiettivi:

Fornire agli studenti le conoscenze per lo sviluppo di applicazioni secondo il paradigma della programmazione ad oggetti.

Competenze:

- Conoscere gli ambiti di applicazione di un oggetto;
- Conoscere ed utilizzare correttamente i termini tecnici relativi alla OOP;
- Saper distinguere tra classe e oggetto;
- Definire le classi con attributi e metodi;
- Creare gli oggetti come istanze delle classi

Contenuti:

- Ereditarietà
- Sovrapposizione/ridefinizione di metodi
- Polimorfismo

Modulo 2: Nozioni avanzate della programmazione ad oggetti
Obiettivi:

Applicare i principi della programmazione ad oggetti: incapsulamento dei dati, ereditarietà, polimorfismo.

Competenze:

- Sapere definire lo stesso metodo con diverse signature;
- Saper derivare una nuova classe da una classe esistente;
- Sapere ridefinire i metodi ereditati da una superclasse;

- Sapere utilizzare istanze di una sottoclasse al posto di istanze della superclasse.

Contenuti:

- Memorizzare i dati su memoria permanente;
- Leggere i dati da memoria permanente;

Modulo 3: Base di Dati
Obiettivi:

Fornire agli studenti una visione di insieme sui diversi tipi di dati del sistema informatico e la

capacità di rappresentare situazioni reali attraverso modelli.

Competenze:

- Acquisire i concetti fondamentali sulle basi di dati;
- Saper rappresentare situazioni reali attraverso modelli.

Contenuti:

- Introduzione ai sistemi informativi aziendali. Differenza tra archivi e basi di dati;
- Teoria del modello Entità/Relazioni (progettazione concettuale e logica).

Modulo 4: Creazione e Gestione di un Data Base
Obiettivi:

Fornire agli studente gli strumenti per creare, modificare e manipolare una base di dati.

Competenze:

- Saper definire, creare ed aprire un nuovo database
- Saper creare una nuova tabella. Saper definire le caratteristiche dei campi nella struttura della tabella.
- Saper caricare i dati nella tabella
- Saper definire le relazioni tra le tabelle
- Saper creare le maschere per facilitare l'inserimento dei dati
- Saper creare i report per la stampa dei dati
- Saper importare ed esportare dati dal database

Contenuti:

- Il software DBMS
- La creazione delle tabelle. Le proprietà dei campi delle tabelle
- Le relazioni tra tabelle
- Le maschere
- I report

Modulo 5: Interrogazione di un database
Fornire agli studente gli strumenti per interrogare una base di dati**.**

Competenze:

- Definire ed eseguire una query;
- Ordinare/ricercare/manipolare i dati in una tabella o in una query

Contenuti:

- Le istruzioni fondamentali del linguaggio SQL;
- DDL, DML, DCL, QL;
- Le operazioni relazionali in SQL;
- L'istruzione Select

Programmazione quinto anno

Obiettivi Specifici di Apprendimento

Verranno studiati i principali algoritmi del calcolo numerico, introdotti i principi teorici della computazione e affrontate le tematiche relative alle reti di computer, ai protocolli di rete, alla struttura di internet e dei servizi di rete . Con l'ausilio degli strumenti acquisiti nel corso dei bienni precedenti, saranno inoltre sviluppate semplici simulazioni come supporto alla ricerca scientifica (studio quantitativo di una teoria, confronto di un modello con i dati…) in alcuni esempi, possibilmente connessi agli argomenti studiati in fisica o in scienze.

Modulo 1: Teoria delle reti
Fornire agli studenti il concetto di rete di elaboratori, di condivisione delle risorse e delle

problematiche relative alla comunicazione tra sistemi di comunicazione. Approfondire la

comunicazione attraverso la rete Internet.

Competenze:

- Classificare una rete in base alla sua estensione e alla sua tipologia;
- Conoscere la struttura della rete Internet;
- Sfruttare i principali servizi offerti dalla rete Internet in maniera consapevole

Contenuti:

- Dall'informatica centralizzata all'informatica distribuita. Dalla rete di terminali alla rete di elaboratori;
- I mezzi trasmissivi;
- Canali trasmissivi: punto-punto, broadcast. Commutazione. Commutazione di circuito. Commutazione di pacchetto a circuito virtuale;
- Classificazione delle reti: LAN, MAN e WAN;
- Reti Client/Server, peer to peer, ibride;
- Modello ISO/OSI. I livelli del modello OSI. Funzioni dei livelli;
- Architettura TCP/IP;
- Panoramica su Internet: indirizzi numerici e indirizzi mnemonici, i DNS, i server di Internet;
- Intranet ed Extranet;
- Il Cloud Computing;
- La sicurezza della rete;
- La crittografia. La firma digitale;
- Gli strumenti e le tecnologie per l’Amministrazione digitale.

Modulo 2: Calcolo numerico

Fornire agli studenti gli strumenti necessari per implementare gli algoritmi per la risoluzione di equazioni e sistemi matematici.

Competenze:

- Comprendere le basi del calcolo numerico;
- Acquisire il concetto dei numeri pseudocasuali;
- Saper utilizzare le funzioni di libreria del C++

Contenuti:

- Scomposizione in fattori primi;
- Calcolo approssimato della radice quadrata;
- Generazione di numeri pseudocasuali;
- Calcolo approssimato della radice di una equazione: metodo di bisezione;
- Calcolo approssimato del seno di un angolo con la serie di Taylor.
- Programmazione e calcolo matriciale in Octave: Prodotto matriciale, Inversa di una matrice, trasposta di una matrice.
- Rappresentazione di una funzione in Octave: la funzione sinusoidale, la polinomiale, le funzioni irrazionali.

Python: un linguaggio di programmazione interpretato

Python è un linguaggio di programmazione di più "alto livello" rispetto alla maggior parte degli altri linguaggi, orientato a oggetti (ma non in maniera ferrea come ad esempio Java), adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

Ideato da Guido van Rossum all'inizio degli anni novanta, il nome fu scelto per la passione dello stesso inventore verso i "Monty Python" ed è spesso paragonato a Ruby, Tcl, Perl, JavaScript, Visual Basic o Scheme.
Spesso viene anche studiato tra i primi linguaggi per la sua somiglianza a uno pseudo-codice e di frequente viene usato per simulare la creazione di software grazie alla flessibilità di sperimentazione consentita da Python, che permette al programmatore di organizzare le idee durante lo sviluppo, come per esempio il creare un gioco tramite Pygame oppure il back-end di un sito web tramite Flask o Django.

Python è un linguaggio di programmazione moderno object-oriented, dalla sintassi semplice e potente che ne facilita l'apprendimento.
Python è un linguaggio **pseudocompilato**: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo.
In Python, non esiste una fase di compilazione separata (come avviene in C, per esempio) che generi un file eseguibile partendo dal sorgente.
Una volta scritto un sorgente, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme attualmente utilizzate, siano esse di casa Apple (Mac) che PC (Microsoft Windows e GNU/Linux), semplicemente, basta la presenza della versione corretta dell'interprete.
Python rappresenta una delle tecnologie principali del core business di colossi come Google (YouTube ad esempio è fondalmentalmente basato su questo linguaggio).
Chiunque nell'arco di un paio di giornate può imparare ad usarlo e a scrivere le sue prime applicazioni.
In questo ambito gioca un ruolo fondamentale la *struttura aperta* del linguaggio, priva di dichiarazioni ridondanti e estremamente simile ad un linguaggio *parlato*.
L'indentazione perde il suo ruolo inteso come stile di buona programmazione per facilitare la lettura del codice, per diventare parte integrante della programmazione che consente di suddividere il codice in blocchi logici.
Ecco alcune notevoli caratteristiche di Python:

- Utilizza un'elegante sintassi, permettendo ai programmi scritti di essere facilmente letti;
- È un linguaggio facile da usare che rende semplice il funzionamento dei vostri programmi. Questo rende Python ideale per lo sviluppo di progetti e di altre attività correlate di programmazione, senza rinunciare alla manutenibilità;
- Viene fornito con un vasta libreria che supportano le più comuni attività di programmazione, come la connessione ai web server, cercare testo con le regex, leggere e modificare files, etc...;
- La modalità interattiva di Python permette di testare brevi frammenti di codice. C'è anche un ambiente di sviluppo in bundle chiamato IDLE;
- È un linguaggio facile da estendere aggiungendo nuovi moduli implementati in un linguaggio compilato come C o C++;
- Può essere incorporato in un'applicazione per fornire un'interfaccia programmabile;
- Può essere eseguito su molto piattaforme come Linux, Windows, Mac OSX e Unix;
- È un software libero. Non esistono costi per scaricare ed usare Python o per includerlo in un'applicazione.

Alcune funzioni del linguaggio sono:

Una varietà dei tipi di dati disponibili: numeri (a virgola mobile, complessi, interi a lunghezza illimitata), stringhe (sia ASCII che Unicode), liste, tuple e dizionari.
Python supporta la programmazione ad oggetti con classi ed ereditarietà multipla;

Il codice può essere raggruppato in moduli e pacchetti;

Il linguaggio supporta un variegato sistema delle eccezioni, risultando chiaro nella gestione degli errori;

I tipi di dati sono fortemente e dinamicamente tipicizzati. Mischiare tipi incompatibili (es. cercare di sommare una stringa ed un numero), scatena un'eccezione:

Python contiene interessanti caratteristiche avanzate come i generatori e le list comprehensions;

Python gestisce automaticamente lo spazio in memoria, liberando dal bisogno di allocare o di liberare manualmente la memoria stessa.

Prima di iniziare avrete bisogno di installare l'interprete Python sul vostro computer e probabilmente lo dovrete scaricare. Se usate una distribuzione Linux o Mac OSX, prima di cercare di installare Python controllate nel vostro sistema, perché molto probabilmente lo avete già.

Per la scrittura di programmi elementari in python e sufficiente il blocco note , ovviamente ci sono IDE dedicate, in Windows ad esempio si può utilizzare notepad++ o VisualStudio.

Per verificare se è installata una verisone di python sul vostro PC, basta lanciare un terminale ed inserire la seguente istruzione al prompt:

C:\Users\Utente>pyhton –version

se vi è installata una versione del software disponibile, verrà identificata dalla risposta nel terminale:

Python x.y.z

dove “x.y.z” identifica il numero di release

Python è un linguaggio interpretato e quindi è immediatamente fruibile anche da terminale per esempio, se si vuole inserire direttamente dei comandi in python da terminale, basta digiare al prompt il comando python ed il terminale si predisporrà per accettare direttamente i comandi scrivendo alcune informazioni a riguardo della release instalata:

Python x.y.z. (…) [MSC…]

Type “help”, “copyright”, “credits” or “license” for more information.

>>>

da questa configurazione si possono adesso inserire i comandi in python che verranno direttamente eseguiti:

>>> 1+1

2

>>>print (“Hello World”)

Hello World

>>>

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

Alcuni esempi di comandi da terminale su operazioni con numeri:

```
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>>
>>> 17 // 3  # floor division discards the fractional part
5
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # result * divisor + remainder
17
```

elevazione a potenza

```
>>> 5 ** 2  # 5 squared
25
>>> 2 ** 7  # 2 to the power of 7
128
```

Pieno supporto per la virgola mobile; operatori con operandi di tipo misto convertono l'operando intero in virgola mobile:

```
>>> 4 * 3.75 - 1
14.0
```

variabili:

```
>>> width = 20
>>> height = 5 * 9
>>> width * height
900
```

Oltre ai numeri, Python può anche manipolare stringhe, che possono essere espresse in diversi modi. Possono essere racchiusi tra virgolette singole ('...') o doppie ("...") con lo stesso risultato '\' può essere utilizzato per sfuggire alle virgolette:

```
>>> 'spam eggs'  # single quotes
'spam eggs'
>>> 'doesn\'t'  # use \' to escape the single quote...
"doesn't"
>>> "doesn't"  # ...or use double quotes instead
"doesn't"
>>> '"Yes," they said.'
'"Yes," they said.'
>>> "\"Yes,\" they said."
'"Yes," they said.'
>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'
```

Nell'interprete interattivo, la stringa di output è racchiusa tra virgolette e i caratteri speciali sono preceduti da barre rovesciate.
Anche se a volte questo potrebbe sembrare diverso dall'input (le virgolette che lo racchiudono potrebbero cambiare), le due stringhe sono equivalenti.
La stringa è racchiusa tra virgolette doppie se la stringa contiene una virgoletta singola e nessuna virgoletta doppia, altrimenti è racchiusa tra virgolette singole.
La funzione print () produce un output più leggibile, omettendo le virgolette racchiuse e stampando caratteri speciali e di escape:

```
>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'
>>> print('"Isn\'t," they said.')
"Isn't," they said.
>>> s = 'First line.\nSecond line.'  # \n means newline
>>> s  # without print(), \n is included in the output
'First line.\nSecond line.'
>>> print(s)  # with print(), \n produces a new line
First line.
Second line.
```

Le stringhe possono essere concatenate (incollate insieme) con l'operatore + e ripetute con *:

```
>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
'unununium'
```

Due o più stringhe letterali (cioè quelle racchiuse tra virgolette) l'una accanto all'altra vengono automaticamente concatenate.
>>> 'Py' 'thon'

'Python'

>>>

Le stringhe possono essere indicizzate (con indice), con il primo carattere che ha indice 0. Non esiste un tipo di carattere separato; un carattere è semplicemente una stringa di dimensione uno:

```
>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n'
```

Gli indici possono anche essere numeri negativi, per iniziare a contare da destra:

```
>>> word[-1]  # last character
'n'
>>> word[-2]  # second-last character
'o'
>>> word[-6]
'P'
```

Oltre all'indicizzazione, è supportata anche la suddivisione in sezioni. Mentre l'indicizzazione viene utilizzata per ottenere singoli caratteri, “slicing” consente di ottenere una sottostringa:

```
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]  # characters from position 2 (included) to 5 (excluded)
'tho'
```

Un modo per ricordare come funzionano le sezioni è pensare agli indici come punti tra i caratteri, con il bordo sinistro del primo carattere numerato 0. Quindi il bordo destro dell'ultimo carattere di una stringa di n caratteri ha indice n, ad esempio:

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

La funzione incorporata len () restituisce la lunghezza di una stringa:

```
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

Python conosce una serie di tipi di dati composti, usati per raggruppare altri valori. Il più versatile è l'elenco, (lista) che può essere scritto come un elenco di valori (elementi) separati da virgole tra parentesi quadre. Gli elenchi possono contenere elementi di tipi diversi, ma in genere gli elementi hanno tutti lo stesso tipo.

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
```

Come le stringhe (e tutti gli altri tipi di sequenza incorporati), gli elenchi possono essere indicizzati e suddivisi:

```
>>> squares[0]  # indexing returns the item
1
>>> squares[-1]
25
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]
```

Gli elenchi supportano anche operazioni come la concatenazione:

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

A differenza delle stringhe, che sono immutabili, le liste sono di tipo mutabile, cioè è possibile cambiarne il contenuto:

```
>>> cubes = [1, 8, 27, 65, 125]  # something's wrong here
>>> 4 ** 3  # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64  # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

Si può anche aggiungere nuovi elementi alla fine della lista, usando il metodo append ():

```
>>> cubes.append(216)  # add the cube of 6
>>> cubes.append(7 ** 3)  # and the cube of 7
>>> cubes
[1, 8, 27, 64, 125, 216, 343]
```

È anche possibile l'assegnazione a sezioni e ciò può persino modificare la dimensione dell'elenco o cancellarlo completamente:

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> letters
[]
```

È possibile nidificare elenchi (creare elenchi contenenti altri elenchi), ad esempio:

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]
'b'
```

I metodi della lista rendono molto facile usare una lista come una pila, dove l'ultimo elemento aggiunto è il primo elemento recuperato ("last-in, first-out").

Per aggiungere un elemento in cima alla pila, usa append (). Per recuperare un elemento dalla cima della pila, usa pop () senza un indice esplicito, per esempio:

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
>>> stack.pop()
6
>>> stack.pop()
5
>>> stack
[3, 4]
```

Python ha una sintassi molto potente, ad esempio, possiamo scrivere una sotto-sequenza iniziale della serie di Fibonacci con poche istruzioni:

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while a < 10:
...     print(a)
...     a, b = b, a+b
...
0
1
1
2
3
5
8
```

Altra versione di stampa, l'argomento della parola chiave end può essere utilizzato per evitare il ritorno a capo dopo l'output o terminare l'output con una stringa diversa:

```
>>> a, b = 0, 1
>>> while a < 1000:
...     print(a, end=',')
...     a, b = b, a+b
...
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

Oltre all'istruzione “while” appena introdotta, Python utilizza le solite istruzioni di controllo del flusso conosciute da altri linguaggi, con alcune modifiche, per esempio, l’istruzione “if”:

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

nota: possono esserci zero o più parti “elif “ e l'altra parte è facoltativa. La parola chiave "elif" è l'abbreviazione di "else if" ed è utile per evitare un rientro eccessivo. Una sequenza if ... elif ... elif ... è un sostituto delle istruzioni switch o case presenti in altre lingue.

L'istruzione “for“ in Python differisce un po 'da ciò a cui potresti essere abituato in C o Pascal. Piuttosto che iterare sempre su una progressione aritmetica di numeri (come in Pascal), o dare all'utente la possibilità di definire sia il passo di iterazione che la condizione di arresto (come C), l'istruzione “for” di Python itera sugli elementi di qualsiasi sequenza (una lista o una stringa), nell'ordine in cui appaiono nella sequenza, ad esempio:

```
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

Il codice che modifica una “lista” durante l'iterazione su quella stessa “lista” può essere difficile da ottenere correttamente. Invece, di solito è più semplice eseguire il ciclo su una copia della lista o creare una nuova lista:

```
# Strategy:  Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy:  Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

Se è necessario iterare su una sequenza di numeri, la funzione incorporata range () è utile. Genera progressioni aritmetiche:

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

Il punto finale dato non fa mai parte della sequenza generata; range (10) genera 10 valori, gli indici legali per elementi di una sequenza di lunghezza 10. È possibile lasciare che l'intervallo inizi con un altro numero, o specificare un incremento diverso (anche negativo; a volte questo è chiamato il 'passo'):

```
range(5, 10)
   5, 6, 7, 8, 9

range(0, 10, 3)
   0, 3, 6, 9

range(-10, -100, -30)
  -10, -40, -70
```

Per scorrere gli indici di una sequenza, si può combinare range () e len () come segue:

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])
...
0 Mary
1 had
2 a
3 little
4 lamb
```

L'istruzione “break”, come in C, esce dal ciclo for o while che racchiude le istruzioni al suo interno.

Le istruzioni loop possono avere una clausola else; viene eseguito quando il ciclo termina per esaurimento dell'iterabile (con for) o quando la condizione diventa falsa (con while), ma non quando il ciclo viene terminato da un'istruzione break. Questo è esemplificato dal seguente ciclo, che cerca i numeri primi:

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print(n, 'equals', x, '*', n//x)
...             break
...     else:
...         # loop fell through without finding a factor
...         print(n, 'is a prime number')
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

L'istruzione “continue”, anch'essa presa in prestito da C, continua con la successiva iterazione del ciclo:

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print("Found an even number", num)
...         continue
...     print("Found an odd number", num)
Found an even number 2
Found an odd number 3
Found an even number 4
Found an odd number 5
Found an even number 6
Found an odd number 7
Found an even number 8
Found an odd number 9
```

L'istruzione pass non fa nulla. Può essere utilizzato quando un'istruzione è richiesta sintatticamente ma il programma non richiede alcuna azione., per esempio:

```
>>> while True:
...     pass  # Busy-wait for keyboard interrupt (Ctrl+C)
...
```

Le funzioni: possiamo creare una funzione che scrive la serie di Fibonacci su un confine arbitrario:

```
>>> def fib(n):    # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print(a, end=' ')
...         a, b = b, a+b
...     print()
...
>>> # Now call the function we just defined:
... fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

La parola chiave “def” introduce una definizione di funzione. Deve essere seguito dal nome della funzione e dall'elenco di parametri formali tra parentesi. Le istruzioni che formano il corpo della funzione iniziano dalla riga successiva e devono essere rientrate.

Indentazione e blocchi di codice

A differenza di altri linguaggi che delimitano blocchi di codice con parentesi grafe (come C, C++ e Java) o con parole riservate come *begin/end*, Python usa l’indentazione.
Indentare il codice è una pratica comune in tutti i linguaggi, perché semplifica la lettura del codice e la compresione della sua struttura.
Anzichè usare due meccanismi separati per compilatori/interpreti (parentesi o keyword), per Python si è scelto di usare l’indentazione per entrambi. Questa scelta ha diversi aspetti positivi, tra cui:

- il linguaggio risulta più chiaro e leggibile;
- la struttura del programma coincide sempre con quella dell’indentazione;
- lo stile di indentazione è necessariamente uniforme in qualsiasi listato.

Questo significa che, in Python, **l’indentazione è significativa**, e che indentare in modo incorretto può portare a comportamenti sbagliati del programma o a errori.

Argomenti predefiniti: la forma più utile è specificare un valore predefinito per uno o più argomenti. Questo crea una funzione che può essere chiamata con meno argomenti di quelli che è definito per consentire, per esempio:

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

Un altro esempio: tutti i parametri formali che si trovano dopo il parametro * args sono argomenti "solo parola chiave", il che significa che possono essere utilizzati solo come parole chiave anziché come argomenti posizionali.

```
>>> def concat(*args, sep="/"):
...     return sep.join(args)
...
>>> concat("earth", "mars", "venus")
'earth/mars/venus'
>>> concat("earth", "mars", "venus", sep=".")
'earth.mars.venus'
```

Esistono ulteriori opzioni per la gestione degli argomenti all'interno delle funzioni che in questo breve capitolo non vengono considerati e si rimanda al tutorial di Python per approfondimenti.

Piccole funzioni anonime possono essere create con la parola chiave lambda. Questa funzione restituisce la somma dei suoi due argomenti: lambda a, b: a + b. Le funzioni Lambda possono essere utilizzate ovunque siano richiesti oggetti funzione. Sono sintatticamente limitati a una singola espressione. Come le definizioni di funzioni annidate, le funzioni lambda possono fare riferimento a variabili dall'ambito contenitore:

```
>>> def make_incrementor(n):
...     return lambda x: x + n
...
>>> f = make_incrementor(42)
>>> f(0)
42
>>> f(1)
43
```

L'esempio precedente utilizza un'espressione lambda per restituire una funzione. Un altro utilizzo è passare una piccola funzione come argomento:

```
>>> pairs = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]
>>> pairs.sort(key=lambda pair: pair[1])
>>> pairs
[(4, 'four'), (1, 'one'), (3, 'three'), (2, 'two')]
```

Altri esempi di applicazione della funzione “lambda”, supponiamo di voler creare un elenco di quadrati, come:

```
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Notare che questo crea (o sovrascrive) una variabile denominata x che esiste ancora dopo il completamento del ciclo. Possiamo calcolare l'elenco dei quadrati senza effetti collaterali utilizzando:

```
squares = list(map(lambda x: x**2, range(10)))
```

o, equivalentemente:

```
squares = [x**2 for x in range(10)]
```

Matrici: consideriamo il seguente esempio di una matrice 3x4 implementata come una lista di 3 vettori di lunghezza 4:

```
>>> matrix = [
...     [1, 2, 3, 4],
...     [5, 6, 7, 8],
...     [9, 10, 11, 12],
... ]
```

La seguente comprensione dell'elenco trasporrà righe e colonne:

```
>>> [[row[i] for row in matrix] for i in range(4)]
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

Come abbiamo visto precedentemente, il listcomp annidato viene valutato nel contesto del for che lo segue, quindi questo esempio è equivalente a:

```
>>> transposed = []
>>> for i in range(4):
...     transposed.append([row[i] for row in matrix])
...
>>> transposed
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

che, a sua volta, è lo stesso di:

```
>>> transposed = []
>>> for i in range(4):
...     # the following 3 lines implement the nested listcomp
...     transposed_row = []
...     for row in matrix:
...         transposed_row.append(row[i])
...     transposed.append(transposed_row)
...
>>> transposed
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

Nel mondo reale, si potrebbe preferire le funzioni incorporate a istruzioni di flusso complesse.

La funzione zip () farebbe un ottimo lavoro per questo caso d'uso:

```
>>> list(zip(*matrix))
[(1, 5, 9), (2, 6, 10), (3, 7, 11), (4, 8, 12)]
```

Si osservi che in questo caso la lista o vettore, è composta da “tuple”.

Una tupla è composta da un numero di valori separati da virgole, ad esempio:

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> # Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> # but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

Le tuple sono immutabili e di solito contengono una sequenza eterogenea di elementi a cui si accede tramite spacchettamento.

Un altro tipo di dati utile incorporato in Python è il dizionario.

I dizionari si trovano talvolta in altre lingue come "memorie associative" o "array associativi".

A differenza delle sequenze, che sono indicizzate da un intervallo di numeri, i dizionari sono indicizzati da chiavi, che possono essere di qualsiasi tipo immutabile; stringhe e numeri possono sempre essere chiavi.

Le tuple possono essere usate come chiavi se contengono solo stringhe, numeri o tuple; se una tupla contiene un oggetto modificabile direttamente o indirettamente, non può essere utilizzata come chiave.

Non puoi utilizzare gli elenchi come chiavi, poiché gli elenchi possono essere modificati sul posto utilizzando assegnazioni di indice, assegnazioni di sezioni o metodi come append () ed extent ().

È meglio pensare a un dizionario come un insieme di coppie chiave: valore, con il requisito che le chiavi siano univoche (all'interno di un dizionario). Una coppia di parentesi graffe crea un dizionario vuoto: {}.

Inserendo un elenco separato da virgole di coppie chiave: valore tra parentesi graffe si aggiungono le coppie chiave: valore iniziali al dizionario; questo è anche il modo in cui i dizionari vengono scritti in output.

Le operazioni principali su un dizionario sono memorizzare un valore con qualche chiave ed estrarre il valore dato dalla chiave.

È anche possibile eliminare una coppia chiave: valore con “del”.

Se si memorizza utilizzando una chiave già in uso, il vecchio valore associato a quella chiave viene dimenticato.

È un errore estrarre un valore utilizzando una chiave inesistente.

L'esecuzione di list (d) su un dizionario restituisce un elenco di tutte le chiavi utilizzate nel dizionario, in ordine di inserzione (se lo si desidera ordinato, è sufficiente utilizzare invece ordinato (d)).

Per verificare se una singola chiave è nel dizionario, utilizzare la parola chiave in.

Ecco un piccolo esempio utilizzando un dizionario:

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>> list(tel)
['jack', 'guido', 'irv']
>>> sorted(tel)
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

Il costruttore dict () crea dizionari direttamente da sequenze di coppie chiave-valore:

```
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

Quando si scorre nei dizionari, la chiave e il valore corrispondente possono essere recuperati contemporaneamente utilizzando il metodo items ().

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
...     print(k, v)
...
gallahad the pure
robin the brave
```

Moduli: se si esce dall'interprete Python e lo inserisce di nuovo, le definizioni che si sono fatte (funzioni e variabili) vengono perse, pertanto, se si desidera scrivere un programma un po 'più lungo, è meglio utilizzare un editor di testo o una IDE dedicata, (integrated development environment) per preparare l'input per l'interprete e eseguirlo invece con quel file come input.

Questo è noto come creare uno script.

Man mano che il programma si allunga, si potrebbe volerlo dividere in più file per una più facile manutenzione.

Si potrebbe anche voler usare una comoda funzione che da utilizzare in diversi programmi senza copiarne la definizione in ogni programma.

Per supportare ciò, Python ha un modo per mettere le definizioni in un file e usarle in uno script o in un'istanza interattiva dell'interprete.

Tale file è chiamato modulo; le definizioni di un modulo possono essere importate in altri moduli o nel modulo principale (la raccolta di variabili a cui si ha accesso in uno script eseguito al livello superiore e in modalità calcolatrice).

Un modulo è un file contenente definizioni e istruzioni Python il nome del file è il nome del modulo con il suffisso .py aggiunto.

All'interno di un modulo, il nome del modulo (come stringa) è disponibile come valore della variabile globale __name__.

Ad esempio, usa il tuo editor di testo o IDE preferito per creare un file chiamato fibo.py nella directory corrente con i seguenti contenuti:

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

Ora attivando l'interprete Python nel terminale si può utilizzare direttamente questo modulo con il seguente comando:

```
>>> import fibo
```

Questo non inserisce i nomi delle funzioni definite in fibo direttamente nella tabella dei simboli corrente; inserisce solo il nome del modulo fibo. Utilizzando il nome del modulo è possibile accedere alle funzioni:

```
>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

Se si intende usare spesso una funzione puoi assegnarla a un nome locale:

```
>>> fib = fibo.fib
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Un modulo può contenere istruzioni eseguibili e definizioni di funzioni. Queste istruzioni hanno lo scopo di inizializzare il modulo. Vengono eseguiti solo la prima volta che il nome del modulo viene rilevato in un'istruzione import (vengono eseguiti anche se il file viene eseguito come script).

Ogni modulo ha la propria tabella dei simboli privata, che viene utilizzata come tabella dei simboli globale da tutte le funzioni definite nel modulo, pertanto, l'autore di un modulo può utilizzare le variabili globali nel modulo senza preoccuparsi di conflitti accidentali con le variabili globali di un utente.

D'altra parte, se si sa cosa si sta facendo si possono usare le variabili globali di un modulo con la stessa notazione usata per fare riferimento alle sue funzioni, “modname.itemname”.

I moduli possono importare altri moduli: è consuetudine, ma non obbligatorio, inserire tutte le istruzioni di importazione all'inizio di un modulo (o script, se è per questo). I nomi dei moduli importati vengono inseriti nella tabella dei simboli globale del modulo di importazione.

Esiste una variante dell'istruzione import che importa i nomi da un modulo direttamente nella tabella dei simboli del modulo di importazione. Per esempio:

```
>>> from fibo import fib, fib2
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

utilizzo di “alias”:

```
>>> from fibo import fib as fibonacci
>>> fibonacci(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

il codice nel modulo verrà eseguito, proprio come se lo avessi importato, ma con __name__ impostato su "__main__". Ciò significa che aggiungendo questo codice alla fine del modulo:

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

puoi rendere il file utilizzabile sia come script che come modulo importabile, perché il codice che analizza la riga di comando viene eseguito solo se il modulo viene eseguito come file "principale" direttamente nel terminale:

```
$ python fibo.py 50
0 1 1 2 3 5 8 13 21 34
```

In questo caso però se il modulo viene importato, il codice non viene eseguito:

```
>>> import fibo
>>>
```

Per velocizzare il caricamento dei moduli, Python memorizza nella cache la versione compilata di ogni modulo nella directory __pycache__ sotto il nome module.version.pyc, dove la versione codifica il formato del file compilato; generalmente contiene il numero di versione di Python.

Python controlla la data di modifica del codice sorgente rispetto alla versione compilata per vedere se non è aggiornata e deve essere ricompilata.

Questo è un processo completamente automatico.

Inoltre, i moduli compilati sono indipendenti dalla piattaforma, quindi la stessa libreria può essere condivisa tra sistemi con architetture differenti.

Esercizio Python: Area e Perimetro del rettangolo

```
# calcolo area e perimetro del rettangolo

# def rettangolo(a,b):

a= int (input("base= "))

b= int (input("altezza= "))

area=a*b # area

perimetro =2*a+2*b

print ("l'area del rettangolo si misura con la formula: Area=base x altezza")

print ("il rettangolo ha un'area pari a ", area)

print ("il perimetro del rettangolo si misura sommando le lunghezze dei lati, oppure P= 2*base+2*altezza")

print ("il perimetro del rettangolo ha un lunghezza pari a ", perimetro)

uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Area del triangolo

```
# calcolo area del triangolo

# def triangolo(a,h):

a= int (input("base= "))

h= int (input("altezza= "))

area=(a*h)/2 # area

#perimetro =a+b+c

print ("l'area del triangolo si misura con la formula: Area=*(base x altezza)/2")

print ("il triangolo ha un'area pari a ", area)

print ("il perimetro del triangolo si misura sommando le lunghezze dei lati")

# print ("il perimetro del triangolo ha un lunghezza pari a ", perimetro)

uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Area del cerchio

```
# calcolo area del cerchio


r= int (input("raggio= "))
d= 2*r
area=(r*r*3.14) # area
perimetro= 2*r*3.14
print ("il diametro del cerchio è lungo il doppio del raggio: ", d)
print ("l'area del cerchio si misura con la formula: Area=*(raggio*raggio*pi_greco)")
print ("il cerchio con raggio",r,  "ha un'area pari a ", area)
print ("il perimetro del cerchio si calcola con la formula P= 2*r*pi_greco")
print ("il cerchio con raggio",r,  "ha un perimetro pari a ", perimetro)
uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Parametri del cilindro

```
# calcolo parametri del cilindro


r= int (input("raggio del cilindro= "))
h= int (input("altezza del cilindro= "))
d= 2*r
area_cerchio=(r*r*3.14) # area
perimetro_cerchio= 2*r*3.14
volume= area_cerchio*h # volume
superficie_cilindrica= perimetro_cerchio*h
superficie_cilindro= (2*area_cerchio) + (perimetro_cerchio * h)
print ("il diametro della  base del cilindro è lungo il doppio del raggio: ", d)
```

```
print ("l'area della base del cilindro si misura con la formula: Area=*(raggio*raggio*pi_greco)")

print ("il cerchio con raggio",r,  "ha un'area pari a ", area_cerchio)

print ("il perimetro della base del cilindro si calcola con la formula P= 2*r*pi_greco")

print ("il cerchio con raggio",r,  "ha un perimetro pari a ", perimetro_cerchio)

print ("il volume del cilindro si misura moltiplicando la base per l'altezza del cilindro: ", volume)

print ("la superficie del cilindro si misura come la somma delle tre superfici:")

print ("le due superfici della base e della sommità: ",area_cerchio, "+", area_cerchio)

print ("e la superficie della parete cilindrica: ",superficie_cilindrica)

print ("la superficie del cilindro risulta essere: ", superficie_cilindro)

uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Parametri del cono

```
# calcolo parametri del cono


r= int (input("raggio della base del cono= "))
h= int (input("altezza del cono= "))
d= 2*r
area_cerchio=(r*r*3.14) # area base
perimetro_cerchio= 2*r*3.14
volume_cono= area_cerchio*h/3 # volume
#superficie_laterale = perimetro_cerchio*apotema/2
#superficie_cono= (area_cerchio) + S_laterale (serve conoscere l'apotema che si calcola -> a=√r2+h2 a = r 2 + h 2 )
print ("il diametro della  base del cono è lungo il doppio del raggio: ", d)
print ("l'area della base del cono si misura con la formula: Area=*(raggio*raggio*pi_greco)")
print ("il cerchio con raggio",r,  "ha un'area pari a ", area_cerchio)
print ("il perimetro della base del cono si calcola con la formula P= 2*r*pi_greco")
print ("il cerchio con raggio",r,  "ha un perimetro pari a ", perimetro_cerchio)
print ("il volume del cono si misura moltiplicando la base per l'altezza del cono e dividendo per 3: ", volume_cono)
print ("la superficie del cono si misura come la somma della superficie della base e quella laterale:")
#print ("la superficie del cono risulta essere: ", superficie_cono)
uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Anagramma di un nome inserito

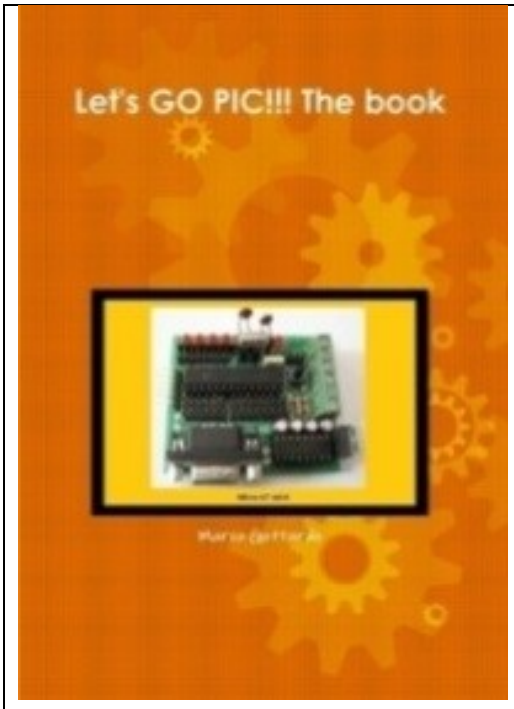
```
#programma anagramma


PAROLA= input ("scrivi il tuo nome: ")


def scambia(lista, p1,p2): # scambia posizioni
   temp=lista[p1]
   lista[p1]=lista[p2]
   lista[p2]=temp
i=0
def permuta(lista,pos,n):
   if(pos == n):
      global i
      print(str().join(lista),end="\t")
      i += 1
   else:
      for k in range(pos,n):
         scambia(lista,pos  ,k)
         permuta(lista,pos+1,n)
         scambia(lista,pos  ,k)


permuta(list(PAROLA),0,len(PAROLA))
#uscita
print ("\n")
print (" ci sono ",i, " parole anagrammate")
uscire= input ("premi un tasto per uscire per usire")
```

Esercizio Python: Il problema della torre di Hanoy

```
# torre di hanoy
def hanoi(n, sorg,temp,dest):
   if n == 1:
      print("sorgente",sorg, " >> ", "destinazione",dest)
   else:
      hanoi(n-1, sorg,dest,temp)
      hanoi(1  , sorg,temp,dest)
      hanoi(n-1, temp,sorg,dest)
#---------------------------------
n=int(input("Quanti dischi? "))
hanoi(n,1,2,3)
conta=0
#---------------------------------
def hanoi(n, sorg,temp,dest):
   global conta
   if(n == 1):
      conta += 1
      print(...)
   else:
      hanoi(n-1, sorg,dest,temp)
      hanoi(1  , sorg,temp,dest)
      hanoi(n-1, temp,sorg,dest)
#---------------------------------
n=int(input("Quanti dischi? "))
hanoi(n,1,2,3)
print("conta",conta)
```

```
from time import *
#---------------------------------
def hanoi(n, sorg,temp,dest):
   if(n == 1):
      pass  # non fare niente...
      # print(...)
   else:
      hanoi(n-1, sorg,dest,temp)
      hanoi(1  , sorg,temp,dest)
      hanoi(n-1, temp,sorg,dest)
#---------------------------------
n=25
clock()
hanoi(n,1,2,3)
print("clock",clock())
```

Altre publicazioni del prof. Marco Gottardo (available on www.lulu.com & Amazon Book).

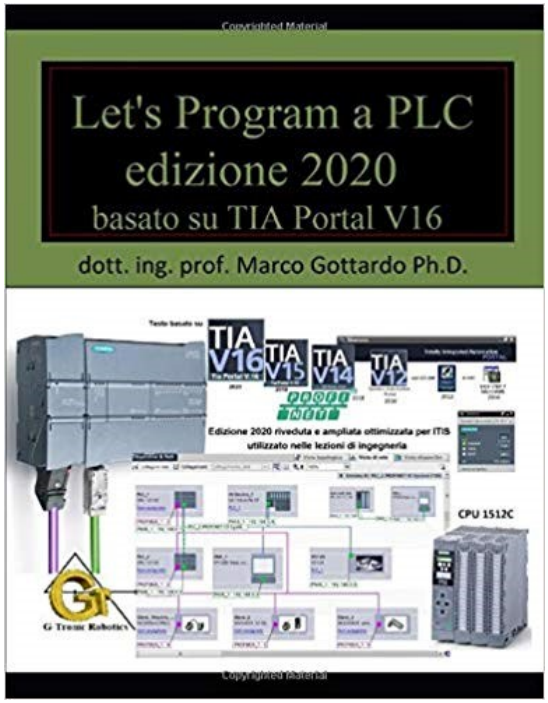


Let's GO PIC!!! The book

Marco Gottardo (UK)

ISBN: 978-1-291-06199-4

Complete course, although aimed at beginners, of programming the PIC microcontrollers, on the MpLab platform and Micro-GT hardware. It contains many exercises and examples in Hitech C and Ladder PIC. Examples of use of digital and analog I / O, LED and LCD displays, motor control and serial communication protocols. Great for schools. Three fully completed papers.



Let's program a PLC (Edizione 2018)

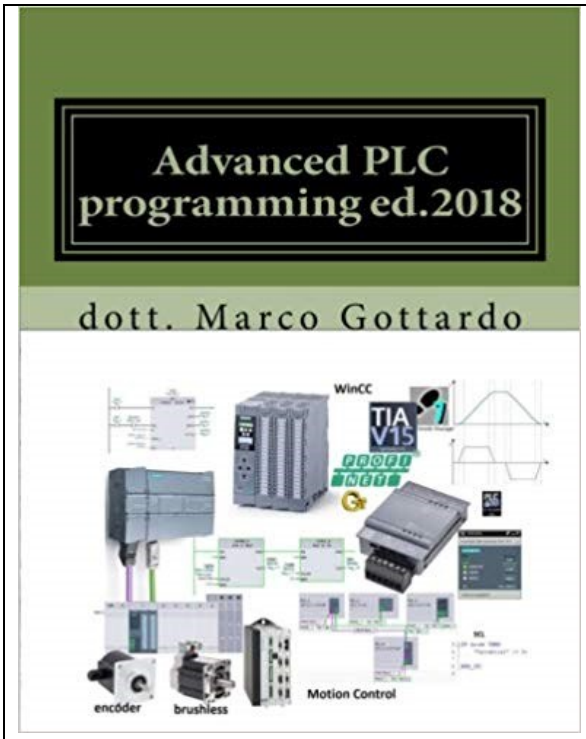
Marco Gottardo

ISBN-10: 1659304008

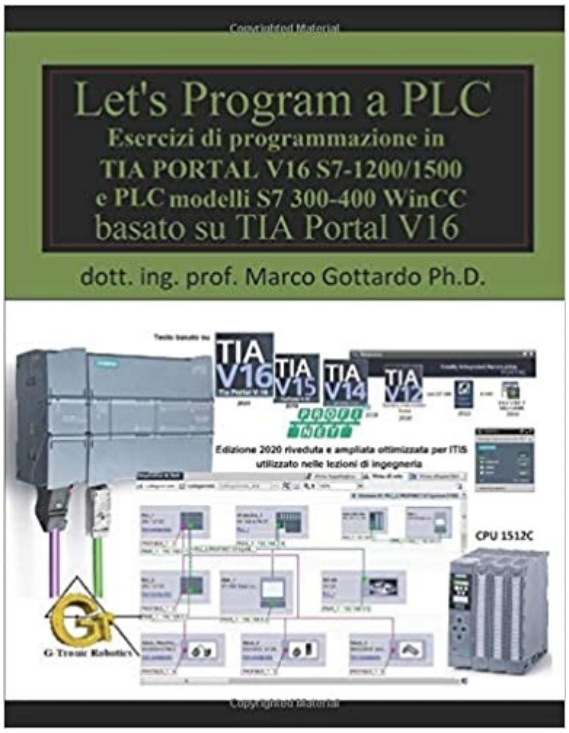
ISBN-13: 978-1659304008 (Italian)

ISBN-13: 978-1724470249 Advanced

This book, published in March 2018, was created to extend and update the previous edition with the same title published in 2016. It contains the evolution towards the new software platforms and new technologies of PLC and TIA Portal V14 networks. It concentrates the twenty-year experience in the field matured by the author and replaces the previous edition already well known and appreciated by the public. A new layout and the succession with which the topics are presented are optimal both for scholastic learning and for self-learning, bringing them knowledge at a professional level. The text is also suitable for university engineering courses. The use of HMI systems programmed via WinCC integrated in the TIA Portal, connected in Profinet and Profibus completes the preparation of the technician. Each topic is accompanied by numerous exercises. The advanced programming section interfaces a three-phase asynchronous motor to an inverter. The chapter on



the normalization of analog signals is fundamental.



Let's Program a PLC!!! Esercizi di programmazione in TIA Portal V16 S7-1200/1500 e PLC modelli S7300-400 WinCC flexible per HMI edizione 2020

Marco Gottardo.

ISBN-13: 979-8611490815 (Italian)

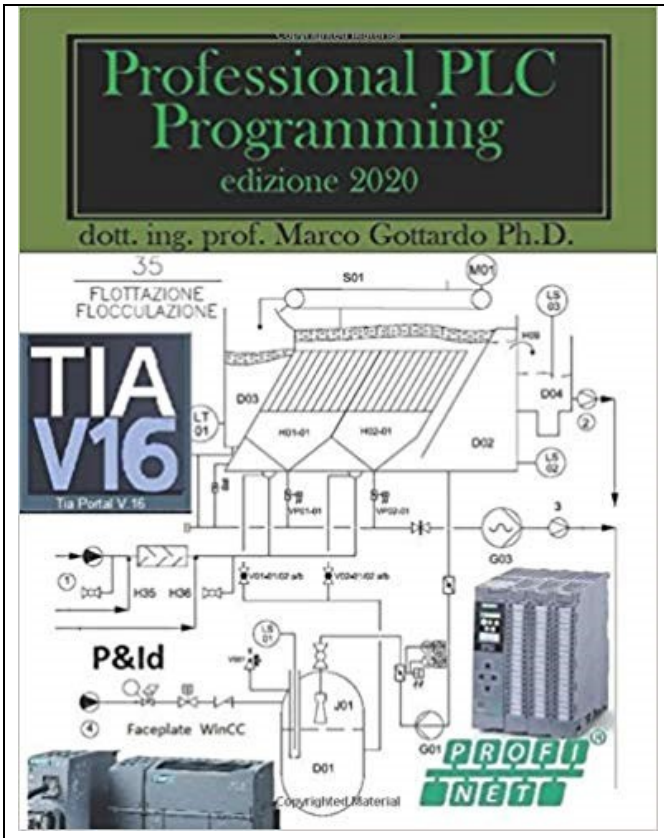
(English traslation in progress)

This book, published in 2018, is the third updated and expanded edition. Based on TIA Portal V16, it also mentions the previous Siemens versions.

Contains 34 exercises performed, with variants 40.

There are 23 proposed exercises but guided towards the solution.

In the "Professional Projects" section there are 4 professional jobs including one with implementation of the machine self-learning contained in an automatic ironer, the use of HMI systems programmed via WinCC connected in Profinet. Of extreme importance is the underground parking lot that can be converted into an automated warehouse. Equally interesting is a tracking solar panel which shows all the construction phases, mechanics, static energy conversion, PLC control. In the advanced programming section, a three-phase asynchronous motor interfaces with a static converter, inverter, paving the way for all real applications. Unique text of its kind that goes far beyond the normal teaching on the PLC. dr. ing. PhD Marco Gottardo.



Professinonal PLC Programming

Lingua: Italiano

Marco Gottardo

ISBN-10: 1711233358

ISBN-13: 978-1711233352

Available on Amazon Book, oriented to the study of industrial processes.

Italian language, imminent translation in English.

Fourth of the complete publication series for school and university. Unique of its kind in Italy.

Let's Build a stereo HiFi

Marco Gottardo

ed. www.lulu.com

(Italiano: English traslation in progress)

Costruire amplificatori stereo HiFi è un ambito dell'elettronica affascinante e grazie ai circuiti integrati sigle ended di approccio morbido anche per il principiante. Qui sono presentate moltissime soluzioni assemblate e collaudate con dettagliate spiegazioni costruttive. L'amplificatore principale è un 40+40Watt realizzato con TDA2051 della ST con mixer a 5 ingressi, selettore digitale dei canali, preampli con controllo di toni, volume, bilanciamento, filtro loudnees e VU-meter a 10+10 led. Imparerete a calcolare la potenza RMS, a distinguere la classe dell'amplificatore a progettare sistemi ibridi integrati/transistor finali di potenza. Tra i numerosi schemi un potente amplificatore da 350W, un sistema home theatre con filtri attivi e sub woofer da 200w.

[Robotica: basi applicative, edizione 2018]

Marco Gottardo

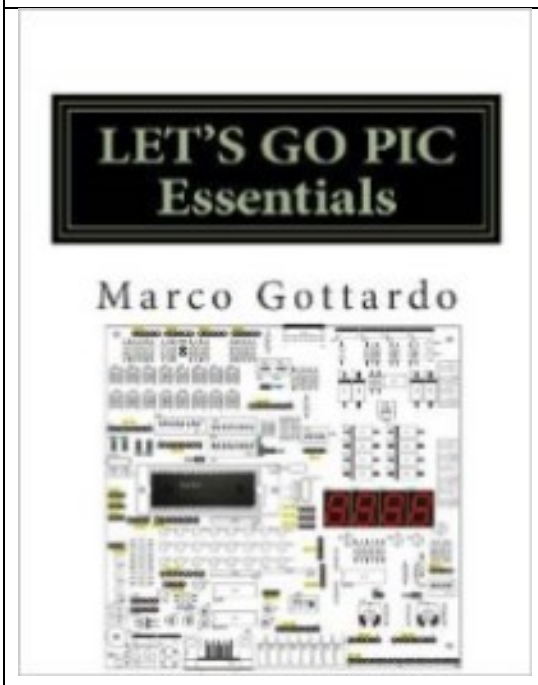
ISBN: 9780244027704

Lingua Italiano

(English traslation in progress)

Questo libro ci proietta nell'affascinante mondo della robotica creando il giusto equilibrio tra l'hobby e la scienza ingegneristica. Tutte le nozioni fondamentali sono impartite ovvero le basi elettroniche, meccaniche, elettrotecniche e delle logiche programmabili. Alcuni concetti più complessi richiedono nozioni ingegneristiche, ad esempio nella programmazione dei sistemi quindi vengono introdotti in maniera descrittiva ma comunque soddisfacente. Le relazioni matematiche avanzate, presenti a fine testo, potranno essere apprese in un secondo momento quando gli studi saranno più maturi.

Buona lettura e ben venuti nel fantastico mondo della robotica



[Let's Go Pic Essentials: Now Based on Micro-gt Ide and Mplab X]

[(UK)]

Disponibile solo su Amazon.it

Prezzo ridotto rispetto all'edizione standard, adattato agli studenti delle scuole superiori.

Blossura: 300 pages

Publisher: Marco Gottardo PhD Independent Publishing Platform; 1 edizione (ottobre 2013)
Lingua: English
ISBN-10: 1492851523
ISBN-13: 978-1492851523
Hevy: 862 g

Centro culturale ZIP corso assemblatore di PC

Lingua: Italiano

(English traslation in progress)

Testo che introduce alla moderna professione del tecnico impiegato nei negozi di computer, o installatore e manutentore d'impianti multimediali. In primo piano è messo l'hardware del PC con sconfinamento nei sistemi operativi, di cui si fa anche un'esaustiva cronistoria. Il lettore imparerà a risolvere i più problemi più comuni, quali la configurazione di una rete locale, l'accesso a internet e la configurazione della posta elettronica. Di grande importanza i capitoli dedicati alle impostazioni del WiFi e delle periferiche più comuni, stampante, il fax lo scanner ecc.

Amministratore, installatore, manutentore delle reti L.A.N

ISBN: 9781291357776 (694 pagine)

Lingua: Italiano

(English traslation in progress)

Excellent support for those technicians who have the role of administrator of the servers and systems. The theory is not neglected, as also the technical terminology is seen in the chapters concerning the TCP / IP and the glossary of a beautiful book. The book focuses on the Windows Server 2012 version, but does not neglect previous versions, Windows server 2003 and Windows server 2008, and Windows 8 clients. The chapter on Active Directory is well developed. Extensive exposure of the RAID system

Elettronica Analogica e Digitale con laboratorio e tecniche SMD. Edizione 2017

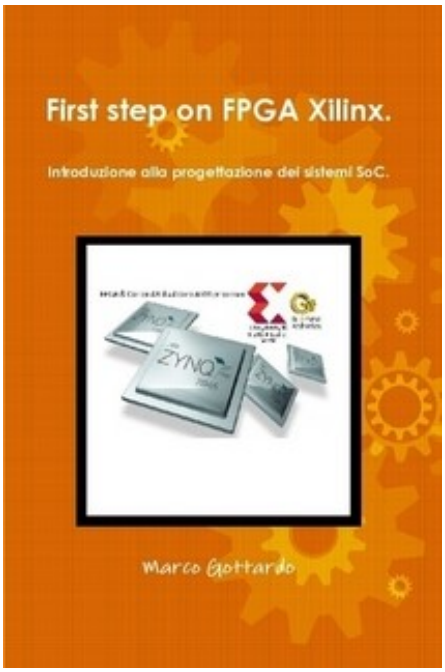
ISBN-10: 1724826638

ISBN-13: 978-1724826633

(lingua Italiano)

Amazon Book prime

Questo libro raccoglie i 25 anni di esperienza didattica dell'autore e propone un'ampia collezione di esperienze pratiche. Il lettore verrà guidato nella soluzioni di problemi che spaziano in praticamente tutti i campi dell'elettronica. Qui troverete anche le necessarie basi teoriche in merito alla tecnologia elettronica, lo sviluppo di circuiti stampati con Eagle, una chiara trattazione di elettrotecnica, forti basi per l'uso degli amplificatori operazionali, elettronica digitale anche programmabile, e unico in Italia un'introduzione allo sviluppo dei sistemi SoC, ovvero System on chip. Questi sono sistemi basati sui processori ZYNQ7000 che integrano una potente sezione ARM multicore con una estesa area FPGA della Xilinx. Un'interessante capitolo sulle trasmissioni radio amatoriali è stato sviluppato dal dott. Marco Barbisan, post doc presso gli istituti di ricerca del CNR di Padova, amico e collega dell'autore.



First step on FPGA Xilinx. Introduzione alla progettazione dei sistemi SoC.

ISBN: 9781326806064

Lingua Italiano

(English traslation in progress)

Con questa pubblicazione il lettore potrà acquisire le nozioni introduttive alle tecniche di programmazione delle FPGA attualmente impiegate nei reali prodotti tecnologici e multimediali commerciali ovvero di largo consumo come smartphone o stazioni di gioco, telecamere, strumenti biomedicali, oppure di nicchia, ad esempio per l'impiego nei sistemi di monitoraggio, acquisizione, controllo in real time in uso nell'ambito della ricerca scientifica.

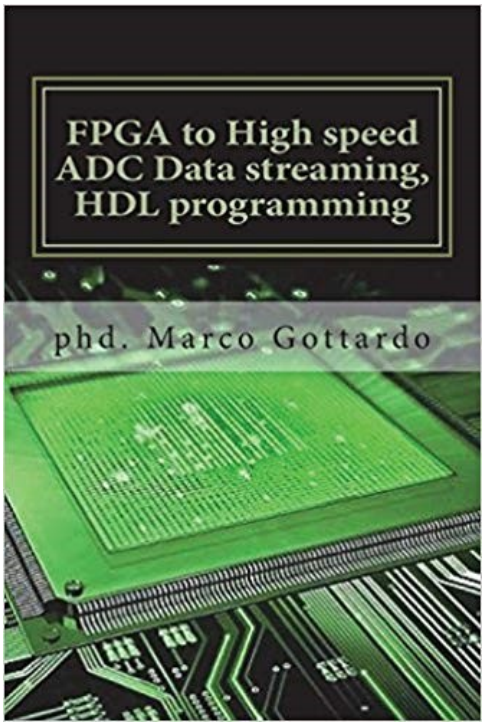
I dispositivi presentati sono estremamente performanti, integrando potenti processori della famiglia ARM multicore oltre alla sezione FPGA di ultima generazione, nello specifico gli Zynq7000 di Xilinx.

Operazionali 2018 Quarta edizione: Pubblicazioni di Elettronica: Volume 2

ISBN-10: 1724838113
ISBN-13: 978-1724838117

Lingua Italiano. Amazon Prime

This fourth edition, completely renewed in its graphic and revised content, was developed to meet the needs for clarity and synthesis requested by the students of the training courses and hobby courses in the area. The text is officially adopted at professional training courses held by Ing. Marco Gottardo at the classrooms of G-Tronic Robotics, in Padua. Compared to the previous edition it is enriched with new chapters for the design of sinusoidal oscillators, the analysis of the frequency response of higher-order active filters, waveform generators, noise analysis, hysteresis comparators, transistor interfacing for audio applications. An important chapter is dedicated to the introduction of the Eagle CAD for the design and development of printed circuits, with a clear guided example. Contains numerous exercises carried out SMD. Detailed chapter on analog filters. Great for self-taught and hobbyists



FPGA to High speed ADC Data streaming, HDL programming: Xilinx Zynq7000 family on Vivado IDE platform: Volume 1 (UK)
Lingua inglese.

ISBN-10: 1720843694
ISBN-13: 978-1720843696

The book set the objective to design and test a high-speed and high-density data acquisition system based on the latest generation FPGA technologies. Topic is from the author Phd thesis and show the latest products released by Xilinx to design a acquire stream system of signals from generic probes (specifically magnetic probes apply on a nucler fusion experiment located in Padova, Italy). The Zynq 7000 family is nowadays state of the art of sistemy SoC that integrating a powerful and extensive FPGA section with an ARM mullticore, with the architecture Cortex A9. Inside the book the basis of HDL programming on Vivado IDE.

Let's program FPGA !!! First guided experience: FPGA Tutorial: Volume 1 (UK)
ISBN-10: 1720843694
ISBN-13: 978-1720843696
Lingua inglese:
Amazon Prime

First chapter of a wide series of tutorials that introduce step by step to the last frontier of electronics. The SoC system with frontend in FPGA technology. Perfect for developing laboratory instruments including portable, robotics, civil and industrial controls, video processing systems and data streaming. To collect.

Origami 3D edizioni Gottardo 2020.

Di Luisa Wang
ISBN-13 : 978-1674878041

Questo libro è la versione a colori della raccolta di creazioni in carta, svolte da Wang HongYing, "Luisa", con spiegazioni guidate e chiare procedure. Gli oggetti sviluppati, e posti nelle mostra permanente di Padova sono davvero tanti quindi qui ne vengono riportati solo alcuni di più semplice realizzazione. Luisa fonda il circolo ricreativo e culturale le Valkirie, in settembre 2019, con sede nel centro culturale della G-Tronic