Fondamenti di Informatica e Programmazione

A cura di **Marianna Amendola**

Edizioni Manna

Edizioni Manna s.r.l.

Sito web: www.edizionimanna.com o www.edizionimanna.it

E-mail: info@edizionimanna.com edizionimanna@pec.it

Fax: 081/522.75.00

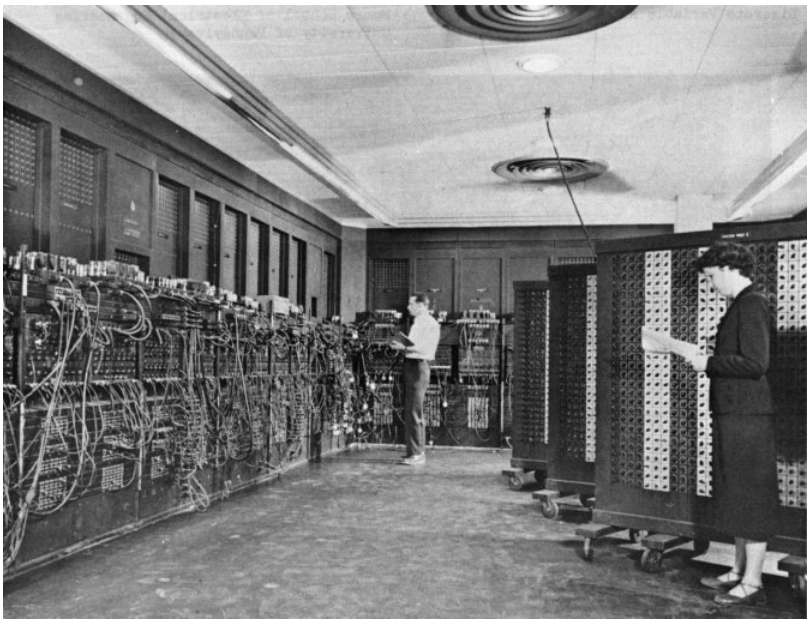
SOMMARIO

SOMMARIO

1.1 Cenni di storia del computer e dei sistemi operativi

1.1.1 Le cinque generazioni del computer

Il primo computer elettronico digitale non sperimentale entrò in funzione nell'Università della Pennsylvania nel 1946, dopo che erano stati necessari tre anni per progettarlo e realizzarlo. Si trattava dell'**ENIAC** (pr. èniac; dalle iniziali di "Electronical Numerical Integrator And Calculator", che significa "integratore e calcolatore numerico elettronico"): 30 tonnellate di peso, 180 metri quadrati di superficie! (vedi figura in basso).



ENIAC inaugurò la **prima generazione** dei computer (allora chiamati ancora "calcolatori"), caratterizzata dall'uso delle valvole termoioniche (più spesso chiamate semplicemente "valvole") che imponevano enormi dimensioni degli elaboratori, spropositati consumi energetici ed emanazione di calore misurabile in centinaia di gradi centigradi. In cambio di tutto questo ENIAC e i suoi parenti erano in grado di eseguire una moltiplicazione a sei cifre in un secondo, mentre i computer attuali in un secondo ne eseguono parecchi milioni.

L'utilizzo delle valvole venne abbandonato nel 1960, quando furono sostituite dai transistor. Nacque così la **seconda generazione** di computer, sostituita a soli quattro anni di distanza dalla **terza generazione**, caratterizzata dall'utilizzo del "chip" (pr. *cìp*), vale a dire un circuito integrato, costituito da una piccola piastra di silicio sulla quale sono impressi con le tecniche della microelettronica diodi, circuiti e transistor: attualmente su un solo chip si possono trovare diversi milioni di questi componenti!

All'inizio degli anni Settanta, con la comparsa del microprocessore, si fa risalire l'inizio della **quarta generazione** dei computer: fu la Intel (pr. *intèl*; è, ancora oggi, il più importante produttore di microprocessori) a creare nel 1970 il primo microprocessore, che fu chiamato 4004.

Il 12 agosto del 1981 la ditta IBM (andrebbe pronunciata *ai-bi-emme,* ma è accettata anche *i-bi-emme*) presentò il suo primo personal computer, espressamente destinato ai privati. Il prezzo era compreso, a seconda delle versioni, tra 1.565 e 2.100 dollari. La IBM sperava di vendere 250.000 personal computer in cinque anni, ma la realtà superò ogni più rosea previsione e l'IBM in dieci anni vendette 15.000.000 di personal computer. Ma il vero successo di quel modello fu un altro: era un modello che poteva essere copiato (o, come si dice in gergo, "clonato") e quindi riprodotto da altre ditte. Nacquero così i computer "IBM compatibili", una categoria alla quale appartiene la grande maggioranza dei PC che usiamo oggi.

Oggi, i computer non sono più riservati a poche persone, sono venduti negli ipermercati, così come i televisori o le lavatrici. Gli ingombranti armadi grigi sono divenuti oggetti delle dimensioni di poco superiori a una agenda da portare con sé nella borsa, con loro non si comunica più introducendovi cartoncini sforacchiati con le domande e leggendo le risposte stampate in modo pessimo su lunghi tabulati, ma si comunica con il mouse e la tastiera e le risposte si possono leggere da ottimi e sottili schermi a colori, da stampanti silenziose, veloci e in grado di riprodurre anche fotografie, oppure si possono ascoltare da casse audio che per resa acustica non hanno nulla da invidiare a quelle degli impianti stereo.

Siamo così giunti, secondo molti studiosi, alla **quinta generazione** dei computer, che adesso sono affiancati da altri dispositivi elettronici estremamente leggeri e trasportabili, ma nello stesso tempo più potenti dei computer da scrivania della precedente generazione, più semplici da usare, al punto che è possibile impartire dei comandi a voce, e in genere sempre collegati a Internet. Si tratta, principalmente, dei **computer portatili o notebook** (pr. *nòtebuk*), costituiti da un unico apparecchio nel quale si trovano tutte le componenti elettroniche, la tastiera e lo schermo del computer; dei **tablet** (pr. *tàblet*), che permettono di utilizzare applicazioni e digitare toccando direttamente sullo schermo, che è sensibile al tocco del dito o di un pennino ed è perciò detto "touchscreen" (pr. *tàuc-scrìn*, con la "c" pronunciata come nella parola "cena"); degli **smartphone** (pr. *smart-fón*), che sono telefoni cellulari che permettono di collegarsi a Internet e di eseguire applicazioni come agenda, rubrica, calendario, posta elettronica, giochi, ecc.

1.1.2 Dialogare con il computer

Il problema più complicato che i programmatori hanno dovuto affrontare sin dagli inizi dell'era del computer, circa settant'anni fa, è stato quello di **mettere in comunicazione la macchina con l'utente**. Il computer ha, infatti, un metodo completamente diverso dall'uomo per immagazzinare, catalogare ed elaborare informazioni: utilizza il sistema matematico del calcolo binario. Ciò significa che ogni informazione, di qualsiasi tipo (numerica, alfabetica, grafica), viene memorizzata dal PC sotto forma di lunghe file di zeri e di uno.

La scelta del **sistema binario** è stata determinata dal fatto che l'utilizzo di due sole cifre (lo 0 e l'1) si adatta perfettamente al funzionamento dei dispositivi elettronici, perché la trasmissione dei dati all'interno del computer avviene tramite un circuito elettrico: quando il circuito è percorso dalla corrente elettrica viene detto "chiuso" e questa condizione comunica al computer la cifra 1, quando non c'è tensione elettrica il circuito è "aperto" e il computer interpreta questo stato con la cifra 0.

Per il computer, ad esempio, la parola “Italia” si scrive così:

01001001 01110100 01100001 01101100 01101001 01100001

Allo stesso modo, il titolo di questo libro (ICDL *più*. Corso di Informatica per il primo biennio delle Superiori), tradotto in linguaggio binario, diventa:

```
01001001 01000011 01000100 01001100 00100000 01110000 01101001 11111001
00101110 00100000 01000011 01101111 01110010 01110011 01101111 00100000
01100100 01101001 00100000 01001001 01101110 01100110 01101111 01110010
01101101 01100001 01110100 01101001 01100011 01100001 00100000 01110000
01100101 01110010 00100000 01101001 01101100 00100000 01110000 01110010
01101001 01101101 01101111 00100000 01100010 01101001 01100101 01101110
01101110 01101001 01101111 00100000 01100100 01100101 01101100 01101100
01100101 00100000 01010011 01110101 01110000 01100101 01110010 01101001
                  01101111 01110010 01101001 00001010
```

Ogni cifra – *0* oppure *1* – si chiama **bit** (deriva da “BInary digiT”, che significa. “cifra binaria”) e una sequenza di otto bit serve al computer per identificare un qualsiasi carattere: lettera, numero o simbolo. Perciò, l’unità di misura fondamentale della memoria di un computer è formata da una sequenza di otto bit, che equivale a un carattere ed è chiamata **byte**.

Capirete che questo metodo – utilissimo per il computer che gestisce il fluire delle informazioni nelle sue memorie e circuiti alla velocità della luce (quasi 300.000 chilometri al secondo) – è per noi umani un linguaggio incomprensibile, abituati come siamo a scambiarci informazioni tramite suoni, immagini e segni grafici.

Quindi bisognava trovare un interprete, un terreno comune dove poter dialogare con un linguaggio comprensibile a entrambi: questo, in sintesi, è il motivo per cui sono nati i **linguaggi di programmazione**.

All’inizio, i programmatori fornivano le istruzioni al computer digitando loro stessi le lunghe file di zeri e di uno che la macchina era in grado di comprendere: a parte il tempo che richiedevano quei “dialoghi”, essi erano patrimonio di ristrette cerchie di persone.

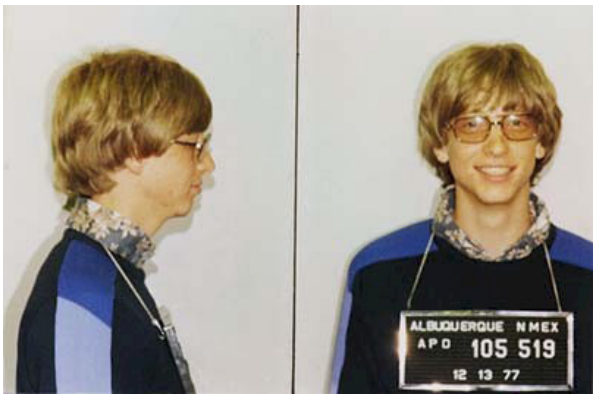
Il primo passo, allora, fu quello di creare delle *interfacce*; si doveva cioè far capire al computer (e all’utente) che un certo numero binario, ad esempio il 65 (che in linguaggio binario corrisponde a questi otto numeri: “01000001”), doveva corrispondere a un segno grafico che aveva questa forma: **A**.

Una volta stabilito questo primo strumento di comunicazione (che prese il nome di codice *ASCII*, pr. **àsci**), il secondo passo fu la parola: infatti, ampliando questa convenzione comune a entrambi (uomo e macchina) si arrivò a stabilire che il comando “**PRINT**”, seguito da una A, significava per il computer far apparire sul monitor una A, e per l’uomo doversi attenere rigidamente a quella codifica; erano così nati una sintassi e un linguaggio: il **BASIC** (pr. *bèsik*).

Ben presto si svilupparono altri linguaggi di programmazione – vale a dire sistemi per comunicare fra l’uomo e la macchina – sempre più evoluti e complessi: Cobol, Pascal, C, C+ (pr. *còbol, pàscal, cì, cì plas*) e altri ancora; tutti questi linguaggi erano (e in parte sono ancora) usati dagli addetti ai lavori, i programmatori appunto; ma cosa si poteva inventare per i comuni utilizzatori dei computer?

1.1.3 Il sistema operativo MS-DOS

Nel 1980, un giovanotto di 25 anni (nella foto sotto ne ha 22 ed è stato appena arrestato dalla polizia per eccesso di velocità) mise in vendita un linguaggio di programmazione basato sul BASIC: il nome di quel giovanotto era **Bill Gates** (pr. *Bill Ghéits*), è stato sino al 2008 proprietario della società informatica Microsoft (pr. màicrosoft) ed è dal 1996 uno degli uomini più ricchi del mondo.



Il nuovo linguaggio di programmazione venne chiamato **MS-DOS** (pr. *emme-esse-dòs;* dalle iniziali di "MicroSoft Disk Operating System"). Nel DOS alcuni concetti erano completamente rivoluzionati: non più linguaggi di programmazione composti da 0 e 1, ma un sistema (tecnicamente si chiama **interfaccia testuale**) che, tramite appropriati comandi, gestiva per l'utente una serie di programmi già pronti, facendoli interagire fra loro.

Dal sistema MS-DOS in poi, questi programmi prendono il nome di **file** (pr. *fàil*) e il computer li mette in comunicazione fra loro per ottenere svariati risultati: far partire un programma di videoscrittura, un gioco, oppure un foglio di calcolo, copiare informazioni da un computer a una penna USB, ecc. ecc.

Dal 1980 sono state prodotte numerose versioni dell'MS-DOS, ognuna contraddistinta da un numero progressivo (ad esempio 2.0, 3.0. 3.3, 4.0, 6.0), in quanto è stato necessario aggiornare il sistema operativo per permettergli di utilizzare le potenzialità dei computer sempre più recenti. Era semplice riconoscere un computer che lavorava in DOS, perché, dopo l'accensione, compariva sul monitor uno schermo nero, vuoto, con la seguente scritta:

```
C:\>_
```

nella quale lampeggiava il trattino finale.

Questo era il cosiddetto *prompt* (pr. *pròmpt*) del DOS.

Il principale limite del DOS era costituito dal fatto che, per svolgere la maggior parte delle operazioni, bisognava conoscere i comandi del sistema operativo, perché ognuno di esso andava digitato sulla tastiera dopo di che doveva essere premuto il tasto *Invio* per eseguire il comando. Il sistema rimaneva quindi complesso e difficile da ricordare, soprattutto per chi non lo usava frequentemente.

1.1.4 Il sistema operativo *Windows 3.x*

Nell'estate del 1983, Bill Gates annunciò di aver creato un nuovo sistema operativo, che aveva deciso di chiamare ***Windows*** (pr. *uìndovs*) perché basato su varie "finestre" (termine che è per l'appunto la traduzione italiana del termine "windows") che si aprivano sullo schermo e che contenevano le varie applicazioni e le diverse cartelle, finestre che potevano essere spostate, ingrandite o rimpicciolite. La vera commercializzazione del prodotto sarebbe però avvenuta solo due anni dopo, con il nome di *Windows 1.0.*

Windows introdusse la cosiddetta **interfaccia grafica**, il che significa che lo schermo nero del DOS venne sostituito da uno sfondo detto *desktop* (pr. *dèsk-tòp*) sul quale compaiono delle piccole immagini con delle brevi didascalie, dette *icone*, sulle quali basta cliccare per far partire i programmi. In inglese "desktop" significa "scrivania", perché, come su una scrivania si trovano di solito gli oggetti di uso più comune, così sul desktop troviamo le applicazioni o i documenti che si usano più spesso, rappresentati sotto forma di icone.

L'obiettivo di Gates era quello di facilitare l'utilizzo del computer eliminando la necessità di adoperare tutti i vecchi comandi testuali del DOS. Per ottenere questo risultato prese ispirazione anche da Mac OS (pr. *mèk-o-esse*), un sistema operativo concorrente creato sempre nel 1982 dalla Apple (pr. **àppol***,* anche se in genere si sente dire **éippol**; significa "mela", dal simbolo di una mela morsicata, che rappresenta l'azienda).

All'inizio il nuovo sistema operativo non suscitò molti entusiasmi. Ma con la terza versione (*Windows 3.0* e i suoi successivi aggiornamenti *Windows 3.1* e *Windows 3.11*, commercializzati nel 1992) il sistema operativo si diffuse rapidamente.

Un'altra novità di *Windows* consisteva nel fatto che, a fianco della tastiera, apparve un aggeggio dalla forma strana, al quale non si trovò di meglio che chiamarlo **mouse** (pr. *màus*) che significa "topo", perché la forma e la presenza di un lungo cavo ricordavano vagamente un topolino con una lunga coda. Il mouse aveva una funzione importante: per far partire un programma non era più necessario scriverne il nome e l'esatta collocazione, bastava qualche clic sulle icone che rappresentavano un programma, un file o una funzione.

1.1.5 I successivi sistemi operativi *Windows*

Nel 1995 avviene una nuova rivoluzione in casa Microsoft, con la commercializzazione del sistema operativo *Windows 95*. L'utente, da *Windows 95* in poi, è guidato passo passo nelle fasi più delicate dell'uso del computer, quelle di installazione di nuovi programmi e della loro rimozione, oppure quando è necessario aggiungere un nuovo componente hardware esterno od interno all'elaboratore stesso, come una stampante, uno scanner o una webcam.

Inoltre, a ogni file corrispondeva un'icona e ogni file dati creato da un determinato programma veniva raffigurato con la stessa icona del programma al quale era associato. Un altro passo in avanti fu costituito dal fatto che, quando non si conosceva bene la funzione di un pulsante o di una icona, bastava in genere puntarci su il puntatore del mouse e attendere un secondo per vedere apparire una brevissima descrizione dell'oggetto o altre informazioni su di esso.

Il salto in avanti fu, dunque, trasformare il "dialogo" fra l'utente e il computer, sostituendo il linguaggio con dei simboli grafici. Ad esempio, nel sistema DOS, per cancellare un file bisognava scrivere DELETE e far seguire il nome del file e l'indicazione precisa della *directory* (pr. *dairèctori*, **è l'equivalente delle odierne cartelle**) dove trovarlo; nel sistema Windows 95 e in quelli seguenti, basta trascinarlo col mouse fino al cestino.

In realtà, le operazioni che avvengono all'interno dei circuiti del computer sono esattamente le stesse, ma dal sistema *Windows 95* in poi (*Windows 98*, *ME*, XP, *Vista*, *7, 8, 10*, ecc.) ci sono presentate in una veste grafica più semplice da interpretare e dunque possiamo disinteressarci di cosa combina il PC al suo interno.

In verità, qualcosa di molto simile (e per diversi aspetti migliore) era già stato realizzato anni prima dalla Apple col sistema operativo Mac *OS* (pr. *mèk-o-esse*), ma questo sistema operativo ha avuto una diffusione inferiore rispetto a Windows, così come altri sistemi operativi tra cui ricordiamo almeno *Linux* (pr. *lìnux*).

Una notevole diffusione ha, invece, il sistema operativo *Android* (pr. *andròid*) nato proprio dall'esperienza di *Linux* e creato per i dispositivi mobili dalla società americana Google (pr. gùgol, è la proprietaria dell'omonimo motore di ricerca). L'altro sistema operativo molto diffuso per i dispositivi mobili è l'*iOS* (pr. *ài-ò-ès*) utilizzato dalla Apple.

Nel punto 1.3.2 del Modulo Computer Essentials forniremo ulteriori informazioni sui pricipali sistemi operativi per computer e dispositivi elettronici.

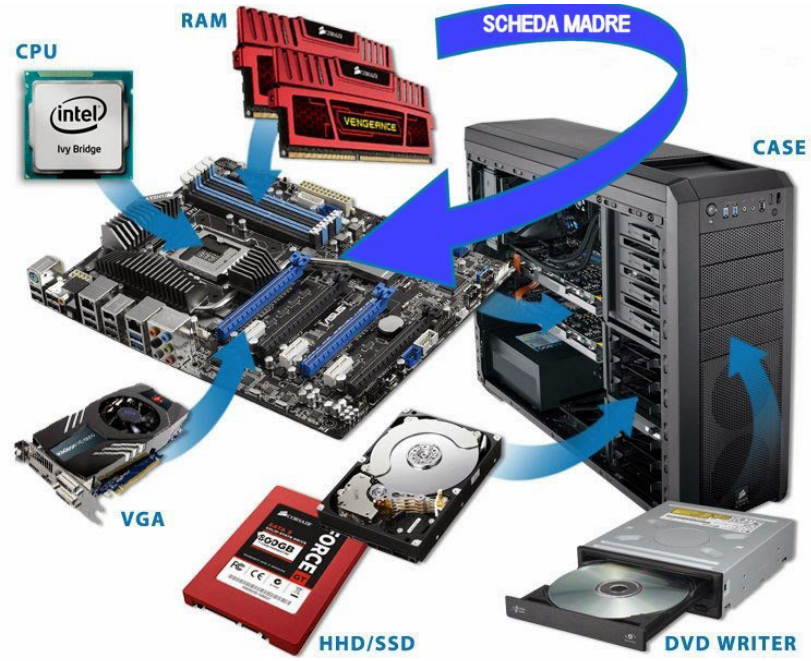
1.2 Caratteristiche logico-funzionali di un computer

1.2.1 Parti principali di un personal computer: unità centrale di elaborazione (CPU), tipi di memoria, disco fisso, dispositivi comuni di input/output

Le parti elettroniche e non elettroniche che formano un personal computer variano in rapporto all'uso e alla potenza del calcolatore. Esistono, comunque, alcune componenti fisse:

- una **unità centrale** (una scatola di metallo detta *case*; pr. *chéis*) nella quale si trovano l'unità centrale di elaborazione (o **CPU**), le varie schede necessarie per il funzionamento del computer: scheda madre, scheda video (serve, come si comprende dal nome, a visualizzare le immagini sul monitor), scheda audio (serve, tramite l'utilizzo di casse acustiche, a riprodurre e a registrare suoni e musica), ecc. Sulla prima di queste schede, la **scheda madre**, detta in inglese *mother board* (pr. *màter bórd*) oppure *mainboard* (pr. *mèin-bórd*) sono inserite la CPU, le memorie del computer, essenzialmente distinte in RAM e ROM),

le prese per il collegamento delle unità che si trovano all'interno del computer (hard disk, lettore-masterizzatore CD e DVD, ecc.), le porte per collegare dispositivi esterni (stampante, monitor, scanner, ecc.) (vedi figura sotto).



La scheda madre è così chiamata perché coordina il lavoro dei vari componenti del sistema, un po' come una mamma che guida i propri figli. Costituisce parte integrante della scheda madre anche il BIOS (la pronuncia corretta è bàios, ma in genere lo si pronuncia come si legge; deriva dalle iniziali di Basic Input/Output System) che gestisce l'accensione del computer. Soprattutto nei computer più economici sono utilizzate schede madri che hanno già incorporate la scheda audio, quella video e il modem.

- L'unità centrale è dotata anche di almeno un **disco fisso** (o *hard disk*, si pronuncia come si legge) o di una memoria allo stato solido (**SSD**) e spesso anche di un **lettore-masterizzatore per CD-ROM** (pr. *sì-dì-ròm*) e **DVD** (si pronuncia *dì-vì-d*).
- Una **tastiera**, un ***mouse*** (si pronuncia *màus*) e delle **casse acustiche**.
- Uno schermo o video, detto più propriamente **monitor**.

Ai componenti essenziali sin qui elencati se ne aggiungono spesso altri: stampante, scanner, ecc. chiamati anche **unità periferiche esterne** e messi in comunicazione con il computer attraverso delle **porte di collegamento**, la più famosa delle quali è la USB (pr. *u-èsse-bì*).

Per approfondire, leggi i punti 1.2.4 e 4.3.1 del Modulo Computer Essentials. Nel primo di essi identificheremo le più comuni porte di input, nel secondo esamineremo i principali tipi di supporti di memoria.

1.2.2 Principali componenti hardware e prestazioni del computer

Con la parola ***hardware*** (pr. ard-uèr), si indicano le parti fisiche (elettriche, elettroniche e meccaniche) del computer (i circuiti elettronici, la tastiera, il lettore DVD, lo schermo, ecc.) e dei dispositivi elettronici in genere. A volte, il termine hardware viene utilizzato per indicare anche gli altri apparecchi che vengono collegati al computer: stampante, scanner, ecc.

I componenti hardware che maggiormente influenzano le prestazioni di un computer sono la velocità della CPU (o processore) e la quantità di memoria RAM a disposizione.

La **velocità della CPU** è calcolata basandosi su una specie di orologio interno del computer detto *clock* (si pronuncia come si legge e significa, per l'appunto, "orologio") che emette un segnale regolare, una specie di battito. La frequenza con la quale il *clock* batte si misura in **hertz** (si pronuncia *èrz* e si abbrevia in *hz*). Un hertz corrisponde a un battito al secondo, per cui un ipotetico computer che funzionasse alla velocità di un hertz sarebbe in grado di effettuare solo un'operazione elementare (ad esempio una addizione) al secondo.

I moderni PC emettono milioni di battiti al secondo, indicati col termine **Megahertz** (si pronuncia *megaèrz* e di solito è abbreviato in MHz), o più spesso miliardi di battiti al secondo, indicati col termine **Gigahertz** (si pronuncia *gigaèrz* ed è abbreviato in GHz).

Le CPU prendono poi il nome anche dalla ditta produttrice e dal loro modello, ad esempio Intel Pentium IV a 1 Ghz, Intel Celeron 1800 MHz, Intel Centrino Core 2 Duo a 2,5 Ghz, AMD Athlon a 3400 MHz, Intel Core I7 a 3,4 Ghz e così via.

È importante tener presente che la velocità in MHz o Ghz vale solo per confrontare CPU costruite con la stessa tecnologia: non è ad esempio possibile paragonare la velocità di una vecchia CPU 80486 con quella di una CPU Pentium, cosi come quella di un Intel Centrino a un Intel Pentium 4. Anche per questo motivo, al fine di valutare la potenza di elaborazione di una CPU si utilizza un altro valore: il *MIPS* (acronimo di "Milioni di Istruzioni elementari Per Secondo"), che indica quanti milioni di operazioni elementari il processore riesce a effettuare in un secondo.

*La **CPU** (la pronuncia corretta è sì-pì-ù, ma molti la pronunciano cì-pì-ù; il termine deriva dalle iniziali di Central Processing Unit, che significa "unità centrale di elaborazione"), detta anche più comunemente processore o microprocessore, è la parte fondamentale del computer, quella che permette l'esecuzione di calcoli e controlla il funzionamento di tutti i programmi. È costituita da un quadratino di silicio (un materiale particolarmente adatto per i circuiti elettronici) della grandezza di uno o due centimetri quadrati, sul quale con le tecniche della microelettronica sono presenti diodi, circuiti e milioni di transistors!*

La CPU è installata sulla scheda madre ed è dotata di una o più ventole di raffreddamento, perché a causa della sua potenza di calcolo sempre maggiore, richiede la presenza di un numero crescente di componenti elettronici che producono molto calore, al punto che se il processore andasse in surriscaldamento, anche se per pochi secondi, potrebbe danneggiarsi.

La CPU è fondamentalmente costituita da due parti: una unità di controllo (detta anche C.U., dall'inglese "Control Unit") che vigila sul funzionamento dei programmi e coordina il lavoro degli altri componenti hardware del computer, e una unità logico-aritmetica (detta anche ALU, dall'inglese "Arithmetic Logic Unit") che svolge per l'appunto funzioni aritmetiche e logiche, vale a dire le operazioni di calcolo e di confronto tra dati.

La CPU è inoltre dotata di una propria memoria speciale, detta memoria cache (quest'ultimo termine si pronuncia chèsc, con la "sc" finale pronunciata come nella parola "scena", e significa "deposito"). Questa memoria speciale della CPU conserva una quantità di dati inferiore rispetto alla memoria RAM (della quale parleremo nel paragrafo 1.1.3.1), ma è molto più rapida, al punto che è definita anche memoria ad accesso immediato. Questa è la ragione per cui nella memoria cache il PC memorizza le informazioni che si utilizzano più frequentemente, in modo che – ricordando le funzioni svolte più spesso dal computer – la memoria cache è in grado di rendere più veloce l'elaboratore.

Come abbiamo detto, l'altro componente hardware che influenza fortemente le prestazioni è la quantità di memoria a disposizione. Con il termine **"memoria" di un computer** si indica un dispositivo in grado di conservare i dati necessari al funzionamento del computer stesso e dei programmi che esso svolge. Fondamentalmente ne esistono due tipologie: la **memoria di massa**, che conserva grandi quantità di dati in modo permanente, e la **memoria veloce** che il computer utilizza quando si avvia e quando elabora i dati.

Con il termine **memorie di massa** si indicano i dispositivi di memorizzazione nei quali i dati registrati (documenti, applicazioni, immagini, suoni, altri tipi di file) restano fin quando l'utente non decide di cancellarli. I principali tipi di memoria di massa sono il disco fisso, la memoria allo stato solido o SSD, la penna (o chiave o pendrive, pr. *pén-dràiv*) USB, le schede di memoria, i CD e i DVD.

La **"memoria veloce" (definita anche "memoria centrale")** consiste essenzialmente nelle cosiddette memorie RAM e ROM.

La **RAM** (si pronuncia come si legge; il nome è costituito dalle iniziali di "Random Access Memory", che significa "memoria ad accesso casuale") è quella parte dell'hardware che il computer utilizza per memorizzare temporaneamente dei dati durante il suo funzionamento (ad esempio le istruzioni dei programmi in esecuzione e i documenti ai quali si sta lavorando), dati che vengono cancellati quando il computer viene spento oppure quando si verifica un'improvvisa mancanza di corrente elettrica: proprio per questo è chiamata anche "memoria volatile".

Potremmo, perciò, paragonare la RAM a un foglietto di appunti che viene gettato via quando non serve più, oppure a una lavagna tradizionale, il cui scritto viene cancellato alla fine di una giornata di lezione. Le dimensioni della RAM si misurano in *Megabyte* e *Gigabyte.* I personal computer dell'ultima generazione sono venduti in genere con 4 o 8 Gigabyte di memoria RAM (del tipo *DDR*, più veloce rispetto al vecchio modello *SDRAM*), ma per RAM di dimensioni minori è spesso possibile ampliarle con l'aggiunta di ulteriori "moduli" di memoria.

Le **ROM** (si pronuncia come si legge; il nome è costituito dalle iniziali di "Read Only Memory", che significa "memoria di sola lettura") sono delle memorie che contengono del software permanente (detto "firmware", termine che si pronuncia *fìrm-uèr*). Le ROM possono trovarsi in alcune interfacce (di rete, del video, ecc.) e nell'elettronica del disco fisso. Una ROM particolare, la BIOS ROM, contiene del firmware che serve a controllare l'hardware del PC, a caricare il sistema operativo dal disco alla memoria fisica (RAM) e a controllare il corretto funzionamento delle periferiche.

Nel punto 1.2.2 del Modulo Computer Essentials troverai altre informazioni sui processori, la memoria RAM e le memorie di massa.

Vi sono però numerosi altri fattori che – seppure in modo più limitato – incidono sulle prestazioni di un computer: ad esempio l'utilizzo di **schede grafiche** dotate di una propria memoria autonoma, la velocità di trasferimento dati dal disco fisso e lo spazio libero disponibile sul disco stesso (quando è inferiore a circa il 10% della dimensione totale si verifica un rallentamento delle prestazioni del PC), la capacità della memoria cache (la memoria speciale utilizzata

dalla CPU), la velocità del BUS (si pronuncia *bas* ed è il circuito elettronico che consente il trasferimento di dati tra i vari componenti elettronici del computer), la presenza di porte e connettori che permettono trasferimenti veloci di dati tra il computer e le sue periferiche, ecc.

Infine, va ricordato che maggiore è il **numero di programmi contemporaneamente in esecuzione** e minore è la velocità di esecuzione di ogni singola applicazione. In alcuni casi, anzi, il funzionamento contemporaneo di un numero particolarmente elevato di programmi può determinare un malfunzionamento del PC.

1.2.3 Ruolo strumentale svolto dal computer nei vari ambiti

Ognuno di noi, quotidianamente, utilizzando un computer, un tablet o uno smartphone effettua delle azioni rese possibili dall'informatica. Ancora più numerose sono le azioni consentite dall'informatica che compiamo senza utilizzare personalmente dispositivi elettronici: ad esempio quando alla cassa del supermercato i codici a barre dei prodotti che abbiamo acquistato comunicano alla cassiera il tipo di prodotto, il suo prezzo ed eventuali promozioni; quando chiediamo un certificato al Comune, a Scuola o a un altro Ente; quando paghiamo un prodotto con la carta di credito; quando ritiriamo banconote a uno sportello Bancomat; quando prenotiamo viaggi in treno o in aereo, e in migliaia di altri casi.

Tutte queste azioni fanno parte delle **Tecnologie della Comunicazione e dell'Informazione**, la cosiddetta **ICT** (da "Information and Communication Technology", pr. *informèscion ènd commùnikescion tecnologi*), che è **la scienza che utilizza il computer, altri dispositivi elettronici e le tecnologie a essi collegati per archiviare, elaborare e trasmettere delle informazioni**.

*In italiano la si indica anche con l'acronimo (vale a dire dalle iniziali delle parole che rappresenta) **TIC**: **T**ecnologie dell'**I**nformazione e della **C**omunicazione. Ai tipi di servizi e di utilizzi dell'ICT (quali servizi Internet, tecnologie mobili, applicazioni di produttività di ufficio) è dedicato il punto 1.1.2 del Modulo Computer Essentials.*

Particolare rilevanza hanno le **applicazioni di produttività di ufficio legate all'ICT** e che consentono la creazione, l'archiviazione e lo scambio di documenti, la gestione della posta, delle attività e del calendario.

Nel punto 1.3.3 del Modulo Computer Essentials, troverai diversi esempi di applicazioni non solo di produttività di ufficio, ma anche di comunicazioni, reti sociali, elaborazioni multimediali, design, oltre ad esempi di applicazioni per dispositivi mobili.

Esercizi

1. In quali anni venne messo in funzione ENIAC, il primo computer elettronico digitale?

A Anni Ottanta dell'Ottocento.
B Anni Venti del Novecento.
C Anni Quaranta del Novecento.
D Anni Sessanta del Novecento.

2. Associa le diverse "generazioni di computer" indicate a sinistra con i loro elementi caratterizzanti elencati a destra.

A	Prima generazione.	1 Primo utilizzo dei "chip".
B	Seconda generazione.	2 Utilizzo delle valvole termoioniche.
C	Terza generazione.	3 Utilizzo dei transistor.
D	Quarta generazione.	4 Diffusione di tablet e smartphone.
E	Quinta generazione.	5 Invenzione del primo microprocessore, l'Intel 2004.

3. Cosa s'intende per "Codice ASCII"?

A Una raccolta di norme di diritto internazionale riguardante la repressione dei reati informatici.
B Una raccolta di norme nazionali che regolano la raccolta e lo scambio di informazioni tramite computer.
C Un sistema di codifica di caratteri, simboli e istruzioni utilizzato nei calcolatori.
D Un codice di autoregolamentazione del commercio elettronico.

4. Individua, tra le seguenti, le affermazioni corrette riguardanti il "byte" (devi barrare due risposte).

A Costituisce l'unità di misura fondamentale della memoria dei computer.
B È formata da una sequenza di otto bit.
C È un tipo di linguaggio di programmazione.
D È un sistema operativo.

5. Con quale termine inglese si indica, convenzionalmente, l'unità centrale del computer, nella quale si trovano l'unità centrale di elaborazione, la scheda madre, le memorie del computer, ecc.?

A Case.
B CPU.
C Motherboard.
D SSD.

6. Con quale termine si indicano componenti aggiuntivi di un computer come stampanti o scanner?

A Memorie esterne.
B Memorie interne.
C Unità periferiche esterne.
D Unità periferiche interne.

7. Quale unità di misura si utilizza per misurare la velocità del processore di un computer?

A Byte.
B Giga.
C Hertz.
D Pixel.

8. Individua, tra le seguenti, due tipi di memoria di massa (devi barrare due risposte).

A Disco fisso.
B RAM.
C ROM.
D SSD (memoria allo stato solido).

9. Il termine RAM deriva dalle iniziali di:

A Random Access Memory.
B Random Automatic Memory.
C Read Access Memory.
D Read Automatic Memory.

10. Il termine ICT deriva dalle iniziali di:

A Information and Communication Technology.
B Internet and Communication Technology.
C Internet and Computer Technology.
D Information and Computer Technology.

2 FASI RISOLUTIVE DI UN PROBLEMA, ALGORITMI E LORO RAPPRESENTAZIONE

2.1 Analizzare, risolvere problemi e codificarne la soluzione

2.1.1 Dal problema al programma

Il computer (e una lunga serie di dispositivi che, in un certo senso, da esso sono derivati, compresi tablet e smartphone) è una macchina capace di eseguire milioni di operazioni al secondo e dotata di una memoria in grado di contenere quantità sempre maggiori di dati, intendendo con il termine "dati" non solo numeri o caratteri, ma anche documenti, suoni, immagini, video.

Potenzialmente, i computer (e i suoi "derivati": tablet, smartphone, ecc.) sono dunque in grado di svolgere compiti numerosi e complicati e il numero e la complicatezza di questi compiti aumenta col progredire della tecnica di produzione dei prodotti informatici.

Il compito che deve essere svolto dal computer è in genere la soluzione di un **problema** che può essere di diversi tipi: matematico, scientifico, economico, finanziario, ecc.

Però, si tratta in sostanza di macchine "stupide", in quanto da sole non sarebbero capaci di risolvere alcun problema e quindi di svolgere un qualsiasi compito: è indispensabile che sia l'uomo a indicare le operazioni da compiere, attraverso una serie di **istruzioni** che indicano alla macchina come elaborare i **dati** e che vengono complessivamente chiamate **programma** o, più recentemente, applicazione o app.

2.1.2 Definire il termine "algoritmo"

Il **programmatore**, o **analista**, studia il problema da risolvere e individua quali operazioni dovrà svolgere il computer per elaborare i dati e raggiungere il risultato. L'insieme di queste istruzioni è detto **algoritmo**.

La parola "algoritmo" deriva dal nome di un matematico arabo vissuto nel nono secolo: Al-Khwarizmi, che inventò la tecnica – che usiamo ancora – per effettuare una moltiplicazione tra due numeri mediante la disposizione a cifre incolonnate.

Possiamo quindi dire che l'algoritmo è un metodo di calcolo che, partendo da dei dati iniziali, permette di arrivare a un determinato risultato attraverso un preciso numero di regole e di operazioni.

In un certo senso, perciò, anche preparare un piatto di spaghetti al sugo può essere definito un algoritmo: si parte da alcuni dati iniziali (gli spaghetti, l'acqua per cuocerli, il sale, i pomodori per il sugo), si adoperano delle regole (la ricetta di cucina) e si compiono delle operazioni

(si cuoce la pasta, si ricava il sugo dai pomodori, lo si aggiunge alla pasta scolata) per arrivare al risultato (vale a dire un piatto di spaghetti al sugo pronto per essere mangiato).

In matematica, ad esempio, il procedimento che calcola la soluzione di un'equazione, oppure quello che trova il minimo comune multiplo in una serie di numeri sono degli algoritmi.

Come abbiamo già detto, le informazioni sono definite "dati", le operazioni sono chiamate "istruzioni" e tutto l'insieme è detto "programma" (o "applicazione" o "app").

Un algoritmo è una sequenza finita di istruzioni che risolvono una classe di problemi. Le istruzioni sono comandi eseguibili dal Sistema di Elaborazione. Il termine algoritmo indica un "procedimento di calcolo" per risolvere problemi di tipo generale, una "classe" di problemi, appunto.

Ad esempio, se vogliamo calcolare l'area di un triangolo di base 15 cm e altezza 18 cm, stiamo risolvendo un singolo problema; se, invece, vogliamo l'area di un generico triangolo, qualunque siano l'altezza h e la base b, allora stiamo affrontando la classe di problemi "area di un triangolo".

2.1.3 Caratteristiche dell'algoritmo

Un algoritmo, per essere tale, deve avere determinate caratteristiche:

- Deve essere **finito**: deve essere composto da un numero finito di istruzioni, ciascuna delle quali deve essere eseguita un numero finito di volte; deve quindi essere sempre possibile individuare il punto di inizio e di fine di ogni algoritmo.
- Deve essere **generale**: deve essere in grado di risolvere tutti i problemi della classe, cioè dare un risultato corretto qualunque siano i valori di partenza.
- Deve essere **riproducibile**: per valori di ingresso uguali deve dare gli stessi risultati.
- Deve essere **ordinato**: si deve conoscere l'ordine di esecuzione delle istruzioni.
- Deve essere **deterministico**: non deve ammettere ambiguità, quindi in ogni momento deve essere sempre chiara quale sia la prossima istruzione da eseguire.

2.1.4 Descrivere in forma algoritmica la procedura risolutiva di semplici problemi

Supponiamo di voler **creare un programma per computer che effettui l'addizione di due numeri**. L'algoritmo che è alla base di questo programma deve indicare al computer, in modo chiaro e seguendo un preciso ordine, i passi necessari per eseguire l'addizione:

1. il computer dovrà chiederci di digitare un numero e poi di premere il tasto *Invio* per confermare l'introduzione del primo numero;
2. il computer dovrà chiederci di digitare un secondo numero e poi di premere nuovamente *Invio;*
3. il computer dovrà sommare il primo numero con il secondo numero;
4. il computer dovrà mostrarci il risultato dell'addizione sul monitor.

Ecco realizzato il nostro algoritmo!

2.1.5 Le variabili

Possiamo vedere che, nel fare le operazioni, si presenta la necessità di memorizzare dei valori, per questo si introduce il concetto di variabile. Queste sono dei nomi formali che indicano dei contenitori di dati, e corrispondono, una volta effettuato il passaggio dall'algoritmo al programma, a degli spazi fisici della memoria del computer dove questi valori saranno conservati.

Una variabile rappresenta simbolicamente uno spazio di memoria dove vengono salvati i dati durante l 'elaborazione.

Il valore di una variabile può cambiare durante l'esecuzione del programma, e di essa va specificato il nome e il tipo: il primo deve essere univoco, cioè individuare la variabile all'interno del programma, mentre il tipo indica la tipologia di informazioni in essa contenute, da cui dipende la rappresentazione interna in memoria. Ad esempio, se la variabile conterrà soltanto valori interi, il suo TIPO sarà "intero con segno" (la rappresentazione in memoria è descritta nella Sezione 3).

In un algoritmo si individuano: variabili di input, variabili di output e variabili di lavoro.

Nelle variabili di input (dall'inglese "in" dentro e "put" mettere) saranno memorizzati i valori iniziali, mentre quelle di Output (dall' inglese "out" fuori e "put" mettere) conteranno i risultati finali, tutte le altre variabili servono per i valori intermedi e i risultati parziali.

Quindi, se utilizziamo due variabili a e b per i due numeri da sommare, e s per il totale, avremo:

1. leggi il valore di a
2. leggi il valore di b
3. s = a + b
4. comunica il valore di s

2.1.6 Tipi di istruzioni

Le istruzioni utilizzabili in un algoritmo possono essere di diversi tipi, a seconda dell'effetto che producono.

- **Istruzioni di dichiarazione**: non modificano il valore delle variabili, ma sono necessarie per comunicare al Sistema di Elaborazione quali sono le variabili che saranno utilizzate e il loro Tipo, per poter riservare loro il giusto spazio di memoria
- **Istruzioni di input**: consentono di modificare il valore di una variabile mediante un input dalla tastiera del computer. Quando viene inserito un valore, quello precedentemente memorizzato nella variabile viene sovrascritto
- **Istruzioni di OUTPUT**: consentono la visualizzazione del contenuto di una variabile
- **Istruzioni di ASSEGNAZIONE:** determinano la modifica del valore di una variabile, il valore precedente andrà sovrascritto. È possibile assegnare ad una variabile direttamente un valore definito, oppure il contenuto di un'altra variabile, o il risultato di una espressione.

- **Istruzioni di Controllo:** Non modificano il contenuto della memoria, ma modificano l'ordine di esecuzione delle istruzioni, che altrimenti è sequenziale. Queste istruzioni sono di diverso tipo: Alternativa, ciclo con contatore, ciclo per vero, ciclo per falso, selezione. Tra queste ci sono anche le istruzioni di INIZIO e FINE.

Gli algoritmi possono essere rappresentati, come abbiamo visto, descrivendo le operazioni attraverso il linguaggio naturale, ma più spesso si tende ad usare una descrizione schematica, vicina al linguaggio di programmazione che si intende utilizzare. Questo tipo di rappresentazione viene chiamata **pseudocodifica**.

2.1.7 Rappresentare algoritmi mediante diagrammi

L'algorimo può essere rappresentato anche graficamente, utilizzando quello che si chiama "diagramma di flusso" o "flow chart" (quest'ultimo termine si pronuncia flò-ciàrt).

Il **diagramma di flusso** è uno schema che permette di rappresentare graficamente un algoritmo.

I diagrammi di flusso permettono alla persona che progetta un programma per computer (il cosiddetto *programmatore* o *analista*) di rappresentare graficamente il programma, in modo che poi sia più facile tradurlo nel "linguaggio" che adopera il computer.

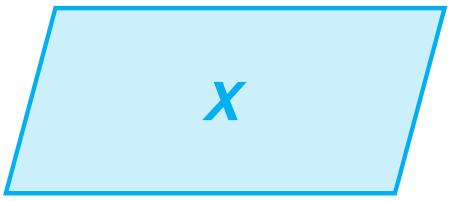
I diagrammi di flusso vengono chiamati anche **diagrammi a blocchi** e sono formati da simboli grafici chiamati "blocchi", generalmente posti uno sotto l'altro e collegati tra loro da frecce che indicano in che ordine devono essere eseguite le diverse istruzioni, ognuna delle quali è rappresentata da un blocco. All'interno di ciascun blocco, infatti, è inserita una descrizione dell'operazione o delle operazioni che devono essere eseguite.

I simboli grafici o blocchi che vengono utilizzati nei diagrammi di flusso indicano il TIPO di istruzioni che vengono utilizzate.

ISTRUZIONI DI DICHIARAZIONE: solitamente non vengano rappresentate nel diagramma; le indicazioni sulle variabili usate e il loro tipo vengono fornite a parte al programmatore o, molto spesso, sono intuitive.

ISTRUZIONI DI INPUT: sono rappresentate con un parallelogramma, al cui interno è indicata la variabile che conterrà il valore inserito, e una lettera I per indicare, appunto, l'input.

ISTRUZIONI DI OUTPUT: sono rappresentate con un parallelogramma: al suo interno è indicata la variabile il cui valore deve essere visualizzato e la lettera O per indicare l'output; può anche essere indicato direttamente un valore numerico o un valore alfanumerico racchiuso tra virgolette.



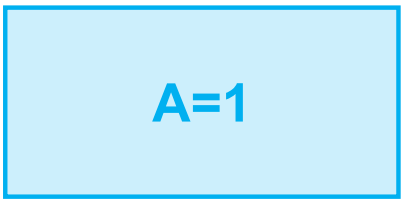
O viene visualizzato il valore di **X**

O viene visualizzato il valore **1**

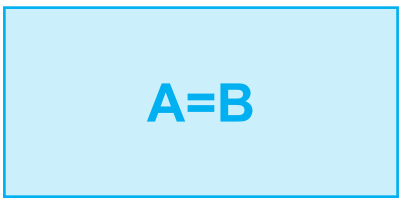
O viene visualizzata la stringa **"esatto"**

ISTRUZIONI DI ASSEGNAZIONE: viene modificato il valore di una variabile; il nuovo valore può essere: un valore esplicito, il contenuto di un'altra variabile o il risultato di una espressione.

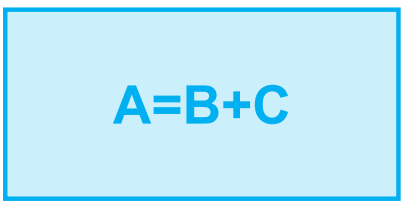
Queste istruzioni sono rappresentate da un rettangolo, al cui interno viene scritta una espressione; ad esempio:



nella variabile **A** viene inserito il valore **1**



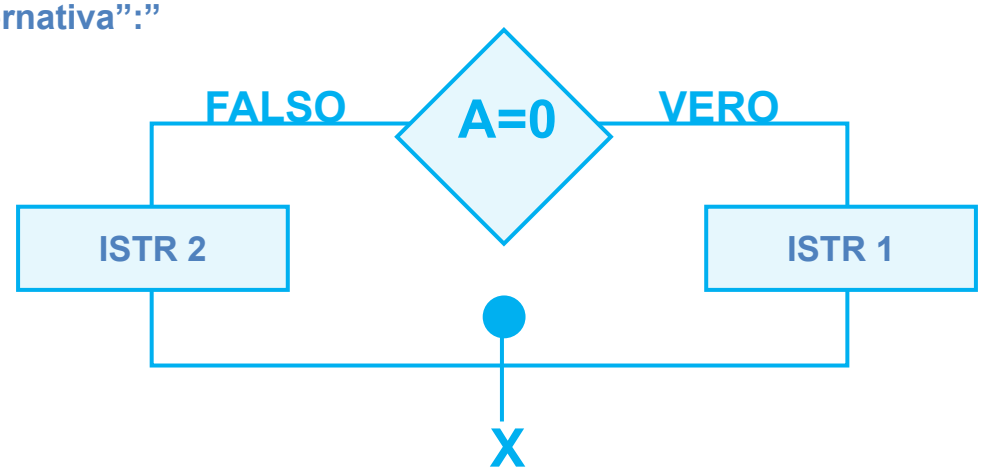
nella variabile **A** viene inserito il valore di **B**



nella variabile **A** viene inserito il risultato dell'espressione **B+C**

ISTRUZIONI DI CONTROLLO: non modificano i valori delle variabili, ma la sequenza con cui le istruzioni vengono eseguite. Ci sono vari tipi di istruzioni di controllo, tutte individuate dalla presenza del cosiddetto "blocco di controllo" o TEST, questo è rappresentato da un rombo con due possibili uscite, a seconda che la condizione indicala all' interno sia vera o falsa.

Blocco "alternativa":"

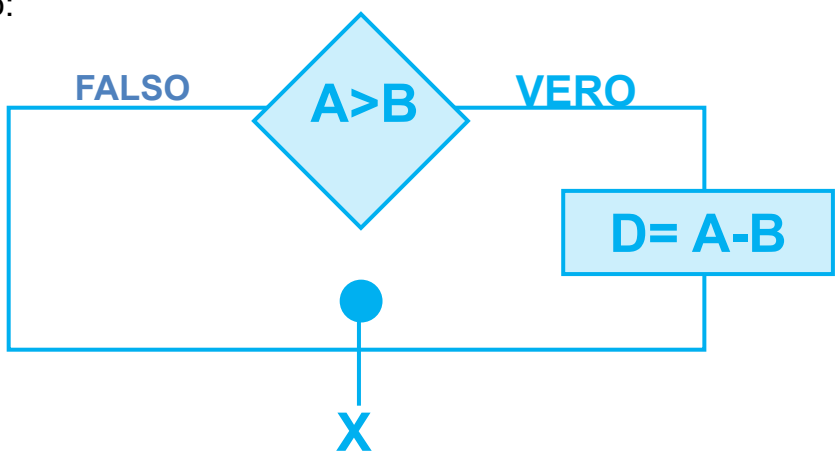


In questo caso saranno eseguite le istruzioni del blocco "ISTR1" se A=0, le istruzioni del blocco "ISTR2" se A è diverso da zero. In entrambi i casi, dopo l'esecuzione di uno dei due blocchi indicati il programma proseguirà dal punto X.

Per convenzione il ramo di destra indica VERO, quello di sinistra FALSO.

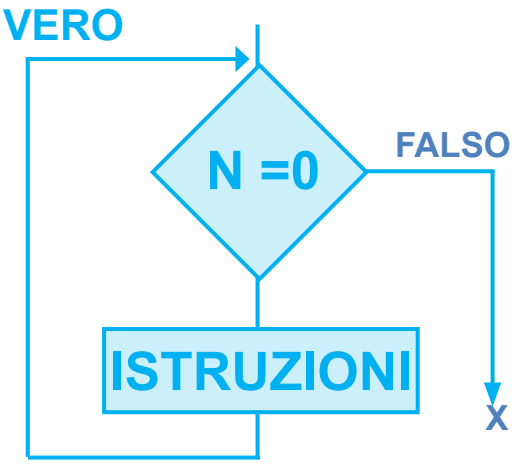
Ovviamente i blocchi possono contenere più istruzioni o addirittura nessuna.

Ad esempio, supponiamo di voler effettuare la differenza A-B, ma solo in caso in cui A sia maggiore di B, avremo:



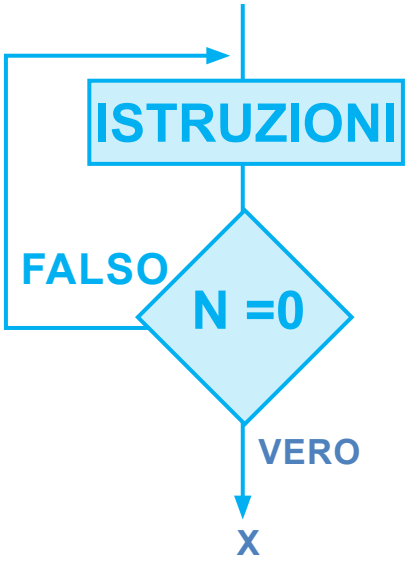
BLOCCO "CICLO PER VERO":

In questo caso il blocco di istruzioni viene ripetuto mentre la condizione è vera; quando questa diventa falsa, la ripetizione si interrompe e il programma riprende dal punto X.



BLOCCO “CICLO PER FALSO”:

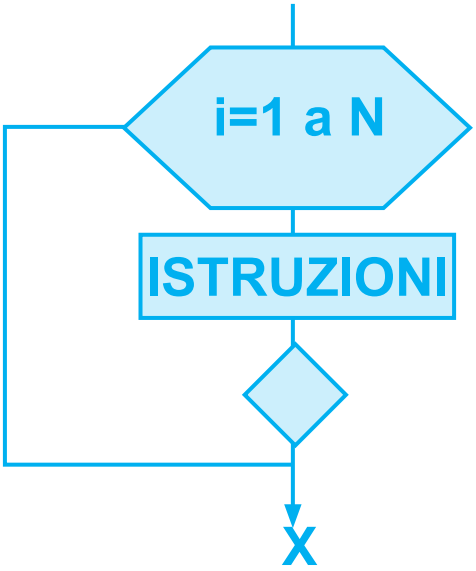
In questo caso il blocco di istruzioni viene eseguito almeno una volta, quindi, se la condizione è falsa, viene ripetuto. Quando la condizione diventa vera la ripetizione si interrompe e l’algoritmo prosegue da X.

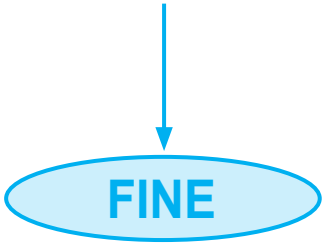


BLOCCO “CICLO CON CONTATORE”:

In questo caso il blocco di istruzioni viene ripetuto un numero N di volte prefissato, il valore della variabile i, detta” indice”, viene incrementato di volta in volta.

Tra le istruzioni di controllo ce ne sono due che indicano l’inizio e la fine dell’algoritmo, e si rappresentano con un ovale e le parole “INIZIO” e “FINE” (molto spesso si preferiscono i termini inglesi “START” e “END”).





Con questi blocchi, combinati opportunamente, è possibile risolvere classi di problemi, anche complessi.

Una volta capito cosa è un algoritmo (un procedimento di calcolo) e come esso è rappresentato graficamente (attraverso i diagrammi di flusso), proviamo a progettare un altro semplice algoritmo numerico e a rappresentarlo poi graficamente.

Creiamo, ad esempio, **un programma che insegni al nostro computer come trovare il maggiore fra tre numeri da noi inseriti**.

Esponiamo allora il problema da risolvere:

Dati tre numeri, l'algoritmo dovrà determinare il maggiore tra questi numeri, che sarà chiamato MASSIMO, e mostrarcelo.

Cominciamo con l'identificare ognuno dei tre numeri con una variabile: rispettivamente *a, b, c,* quindi utilizzeremo più volte una variabile M, che conterrà il numero massimo trovato.

A questo punto dovremo "dire" al nostro computer di confrontare tra loro i primi 2 numeri (*a* e *b*) in modo da determinare quale tra essi è superiore all'altro.

Se *a* è superiore a *b* allora "diremo" al nostro computer che *a* è da considerarsi il numero massimo, quindi porremo *M* = *a*, nel caso opposto porremo *M* = *b*.

Quindi faremo confrontare il massimo *M* con il terzo numero (*c*).

Se *M* è superiore a c la situazione resterà immutata, nel caso opposto il valore di c sarà il nuovo massimo, quindi porremo *M*=*c*.

Infine, "diremo" al computer di mostrarci (ad esempio sul monitor) il numero massimo che risulta da queste operazioni, cioè il valore di *M*.

Per essere ancora più chiari, utilizziamo l'algoritmo che abbiamo appena progettato con dei veri e propri numeri: 90, 65 e 98.

In base al programma che abbiamo scritto, per il computer il numero 90 corrisponderà ad *a*, il numero 65 a *b* e il numero 98 a *c*.

Il computer confronterà *a* con *b* per vedere se *a* ha un valore superiore a *b*.

Dal momento che è proprio così (90 è superiore a 65) il computer considererà quale numero massimo *a*, quindi assegnerà a *M* il valore 90.

A questo punto il computer paragonerà *c* (98) a *M* (90); dal momento che tra i due numeri *c* è superiore, sarà proprio quel numero (98) a costituire il numero massimo, sostituendo il precedente valore di *M*.

Infine, il computer mostrerà sul monitor qual è il valore di *M*: 98.

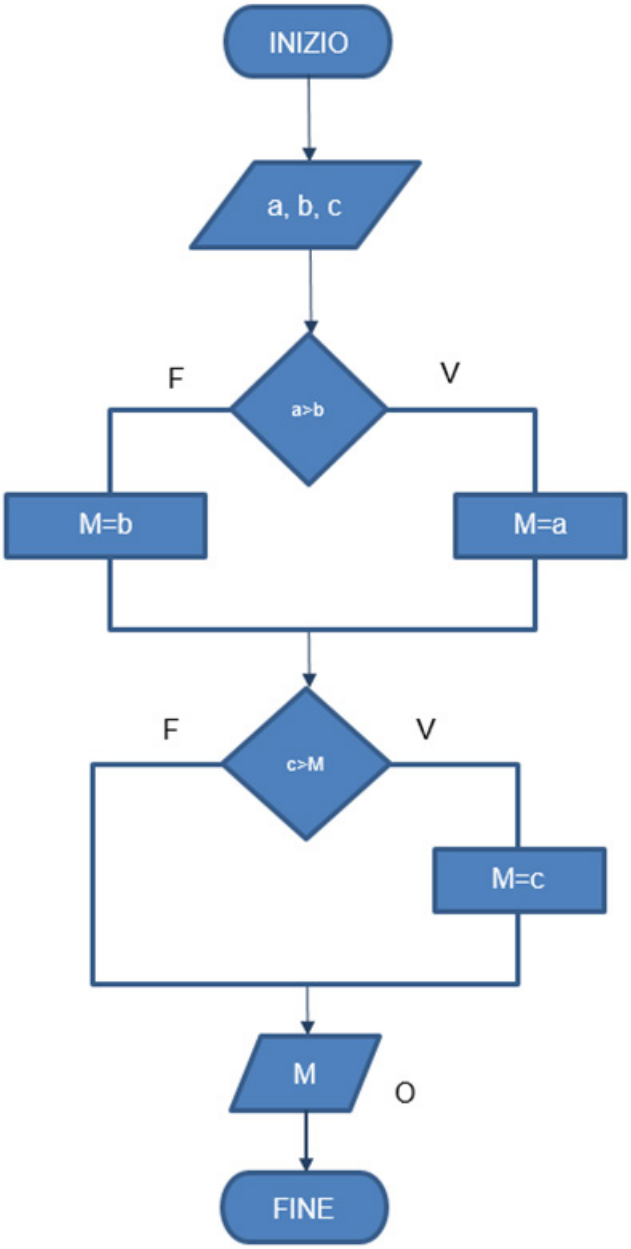
Qualcuno si chiederà: ma è necessario un computer per determinare quale tra soli 3 numeri è superiore? La risposta è, ovviamente, no; ma la grande capacità dei computer è quella di effettuare operazioni semplici ma a una velocità stratosferica.

Per questo motivo, una volta creato l'algoritmo base, il computer è in grado di trovare il numero massimo non solo fra tre numeri ma anche fra tremila o trecentomila miliardi di numeri. Non solo: in tutti i casi il calcolo richiederebbe una frazione di secondo e il computer potrebbe effettuarlo anche utilizzando dei dati già presenti in archivi.

Sarebbe, ad esempio, possibile chiedergli di attingere alla banca dati del censimento nazionale, per chiedergli di mostrarci tra i circa sessanta milioni di Italiani esistenti chi ha l'età maggiore: a noi servirebbe una vita per confrontare sessanta milioni di numeri, al

computer basterà qualche secondo e l'algoritmo necessario, fondamentalmente, è lo stesso che abbiamo creato nel nostro esempio!

Rappresentiamo ora graficamente l'algoritmo che abbiamo appena creato. Ecco il diagramma di flusso che lo rappresenta:



Esercizi

1. Cos'è un algoritmo?

A. È un programma di calcolo.
B È un programma per riprodurre ritmi musicali.
C È un metodo di calcolo contenente le istruzioni necessarie per risolvere un problema.
D È un computer adoperato per effettuare delle operazioni matematiche.

2. Supponiamo di voler creare un semplice programma per computer che permetta di effettuare addizioni di due numeri. L'algoritmo che è alla base di questo programma deve indicare al computer:

A l'origine dei dati in ingresso nel computer.
B lo schema delle componenti del computer.
C i passi necessari per eseguire l'addizione.
D la formula di calcolo.

3. Cosa sono i diagrammi di flusso?

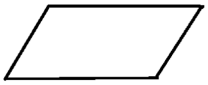
A Sono rappresentazioni statiche di programmi.
B Sono rappresentazioni dinamiche di programmi.
C Sono rappresentazioni grafiche di algoritmi.
D Sono rappresentazioni schematiche delle componenti di un computer.

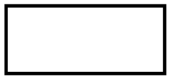
4. In un diagramma di flusso, cosa indica la figura geometrica del rombo?

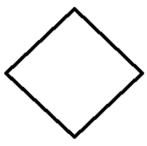
A L'inizio dell'algoritmo.
B La fine dell'algoritmo.
C Una scelta tra due percorsi alternativi.
D Un'operazione di immissione dei dati.

5.Collega, con un tratto di penna, ciascuna figura con il contenuto che viene inserito al suo interno in un diagramma di flusso.

A 
1 Contiene una domanda con scelta fra due possibilità.

B 
2 Contiene un comando da eseguire.

C 
3 Contiene l'indicazione di dati in entrata o in uscita.

D 
4 Contiene le scritte "inizio" o "fine".

6. Quale tra le seguenti affermazioni è corretta?

A Un algoritmo DEVE essere finito, indeterminato, breve
B Un algoritmo DEVE essere finito, riproducibile, generale
C Un algoritmo DEVE essere infinito, indeterminato, ordinato
D Un algoritmo DEVE essere finito, particolare, complesso

7. Quale tra le seguenti affermazioni NON è corretta

A Le istruzioni di Input/output si rappresentano con un parallelogramma
B Le istruzioni di Assegnazione si rappresentano con un rettangolo
C Il rombo rappresenta una scelta tra due alternative
D Le istruzioni di controllo rappresentano un ciclo

8. Come si rappresentano le istruzioni di dichiarazione in un diagramma di flusso?

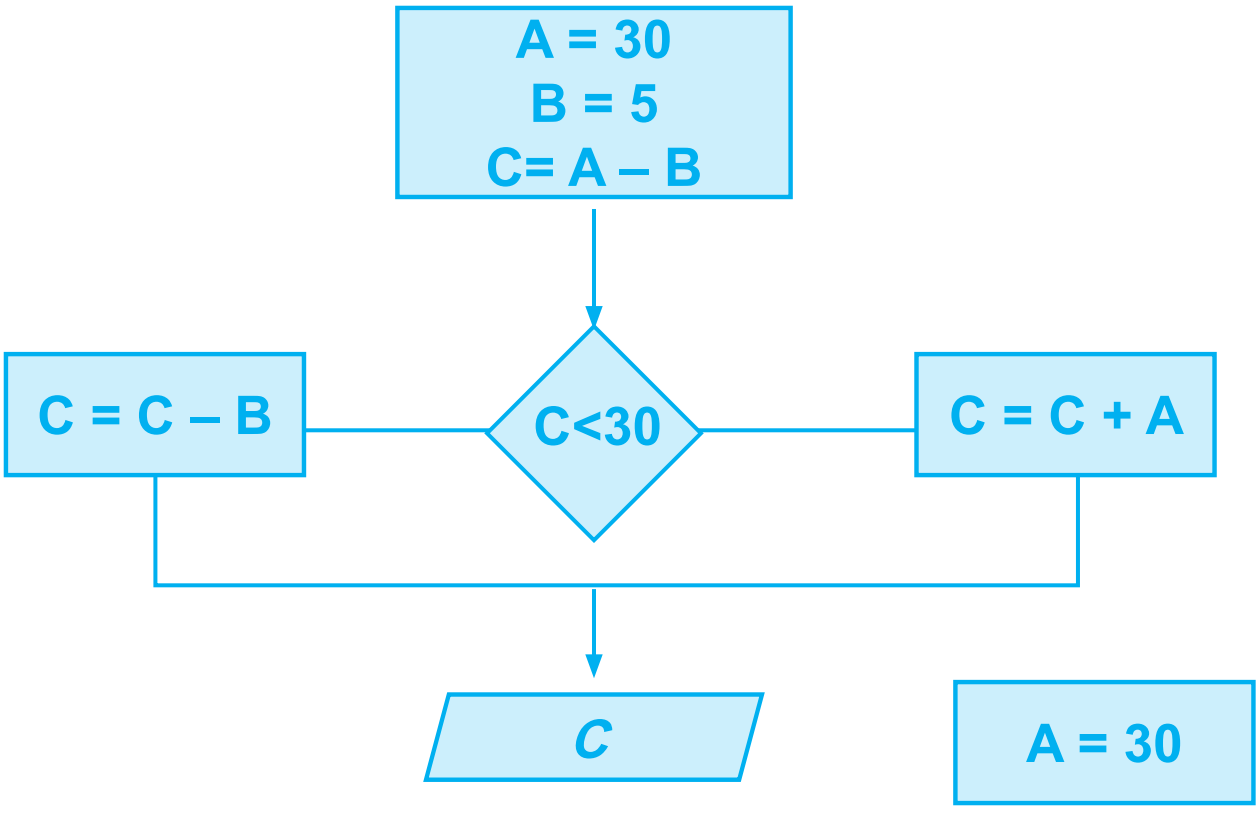
A Con un parallelogramma
B Non si rappresentano in un diagramma di flusso
C Con un ovale
D Con un rettangolo

9. Che tipo di istruzione è il Ciclo per falso?

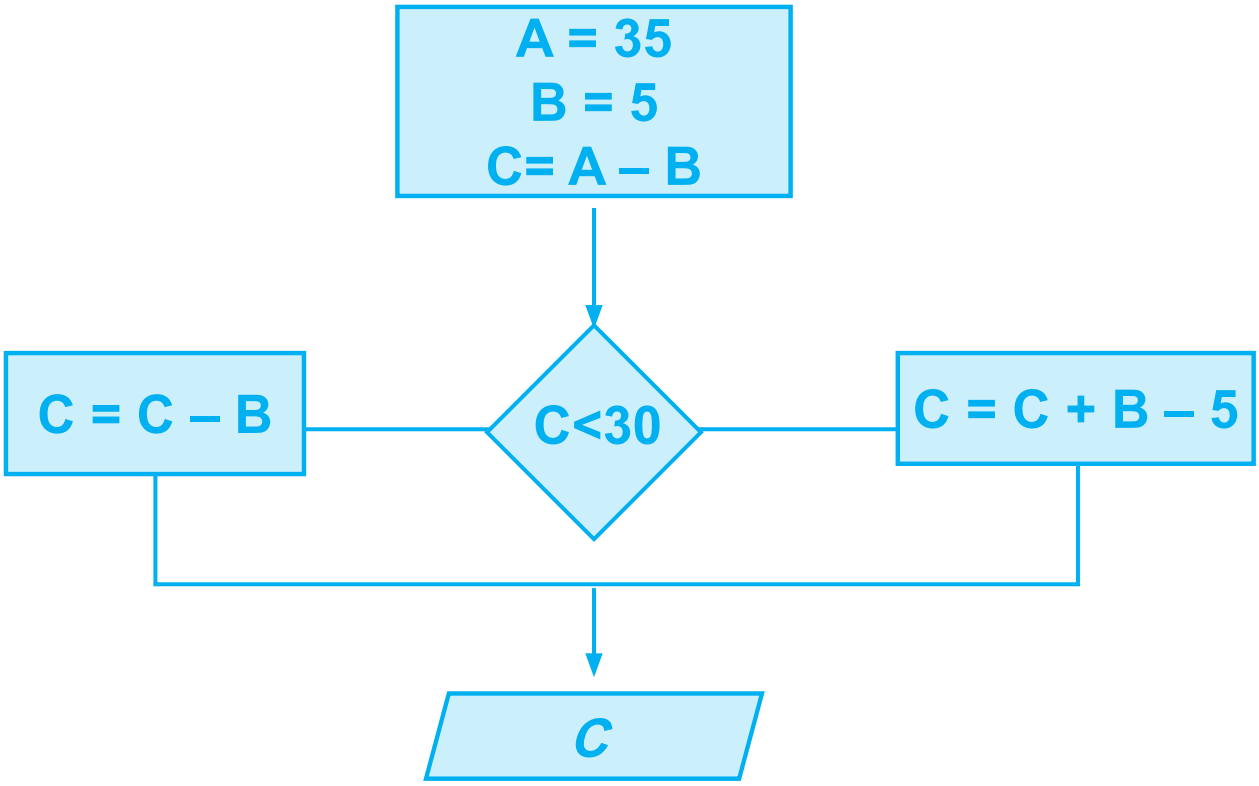
A Controllo
B Assegnazione
C Alternativa
D Input/output

10. Quale tra i seguenti gruppi di istruzioni dà come risultato 30?

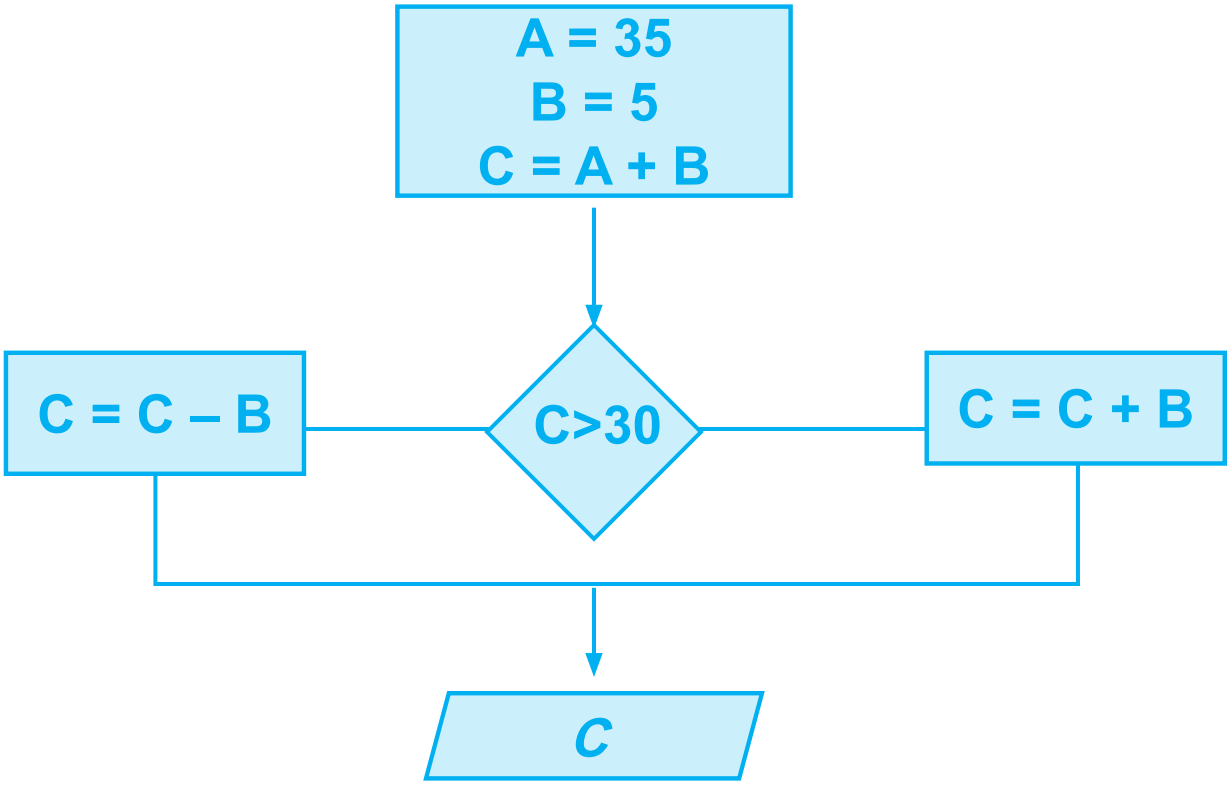
A



B



C



3 FONDAMENTI DI PROGRAMMAZIONE E SVILUPPO DI SEMPLICI PROGRAMMI

3.1 Rappresentazione dei dati

3.1.1 Sistemi di numerazione

Sin dall'antichità si hanno tracce di metodi che gli uomini hanno utilizzato per poter contare ed effettuare dei calcoli. Questi metodi erano strettamente legati alla necessità di rappresentare le quantità numeriche.

I sistemi di numerazione possono essere posizionali o non-posizionali.

In questi ultimi le cifre, cioè i simboli utilizzati, hanno sempre lo stesso valore, mentre nei sistemi posizionali il valore delle singole cifre dipende dalla posizione che occupano rispetto agli altri simboli.

Per esempio, i Romani utilizzavano un sistema di numerazione non posizionale, di tipo additivo, in cui i numeri erano rappresentati da lettere che avevano sempre lo stesso valore all'interno della rappresentazione.

Per contare e per eseguire calcoli noi usiamo il sistema decimale, che rappresenta i numeri attraverso dieci simboli che vanno dallo 0 al 9, ma ciò non significa che esso sia un sistema migliore o peggiore di altri.

Questo sistema, di origini arabe, è di tipo posizionale su base 10: in pratica un numero è rappresentato da una combinazione di cifre che modificano il loro valore secondo una successione di potenze di 10 (unità, decine, centinaia, ecc).

La scelta del sistema decimale deriva dal fatto che gli uomini hanno dieci dita, per cui sin dall'antichità l'uso del sistema decimale permetteva di eseguire più facilmente i calcoli utilizzando proprio le dita per aiutarsi, come fanno ancora oggi i bambini e le persone poco abili in matematica.

Quindi, se gli uomini avessero avuto 12 dita invece di 10 è molto probabile che oggi utilizzeremmo un sistema in base 12, che richiederebbe l'utilizzo di 12 e non di 10 simboli. Per il resto, le norme che regolano le operazioni sarebbero identiche.

Il numero 23, ad esempio, ha un valore diverso dal numero 32, pur contenendo le stesse cifre. Questo perché 23 = 3*1 + 2 *10, mentre 32 = 3*1 + 2*10.

A partire da destra verso sinistra, infatti, le cifre vengono moltiplicate per potenze successive di 10 (100, 101, 102, …), quindi i valori vengono sommati.

Un altro esempio: 341 = 1 X 100 (unità) + 4 X 101 (decine) + 3 X 102 (centinaia) = 1 X 1 + 4 X 10 + 3 X 100

I sistemi posizionali possono avere qualunque base, secondo lo stesso principio:

un numero x può essere rappresentato in una base b con n cifre ai da 0 a b-1, tali che

$a_0 * b_0 + a_1 * b_1 + \ldots + a_n b_n = x$

Per passare da un numero decimale ad un numero in un'altra base, si eseguono una serie di divisioni successive per la nuova base, fino a che il risultato sia 0, e si trascrivono i resti (sempre compresi tra 0 e b-1) in ordine inverso.

Esempio:

Per trasformare 341 in base 8, si avranno i seguenti passi

- 341 : 8 = 42 con resto 5
- 42 : 8 = 5 con resto 2
- 5 : 8 = 0 con resto 5

Si avrà $341_{10} = 525_8$ (il numero in basso indica la base, se è omesso si sottintende base decimale)

Per l'operazione inversa, basterà applicare la formula indicata in precedenza, quindi:

$5258 = 5 * 8^0 + 2 * 8^1 + 5 * 8^2 = 5 * 1 + 2 * 8 + 5 * 64 = 341$

3.1.2 Il Sistema binario

Il problema più complicato che i programmatori hanno dovuto affrontare sin dagli inizi dell'era del computer, circa settant'anni fa, è stato quello di mettere in comunicazione la macchina con l'utente. Il computer ha, infatti, un metodo completamente diverso dall'uomo per immagazzinare, catalogare ed elaborare informazioni: utilizza il sistema matematico del calcolo **binario**.

Ciò significa che ogni informazione, di qualsiasi tipo (numerica, alfabetica, grafica), viene memorizzata dal PC sotto forma di lunghe file di zeri e di uno.

La scelta del sistema binario è stata determinata dal fatto che l'utilizzo di due sole cifre (lo 0 e l'1) si adatta perfettamente al funzionamento dei dispositivi elettronici, perché la trasmissione dei dati all'interno del computer avviene tramite un circuito elettrico: quando il circuito è percorso dalla corrente elettrica viene detto "chiuso" e questa condizione comunica al computer la cifra 1, quando non c'è tensione elettrica il circuito è "aperto" e il computer interpreta questo stato con la cifra 0.

Si è pensato, allora, di utilizzare il **Sistema di Numerazione in base 2**, che utilizza solo le cifre 0 e 1, e consente di rappresentare tutti i numeri ed effettuare operazioni.

Ogni cifra – 0 oppure 1 – si chiama bit (deriva da "BInary digiT", che significa. "cifra binaria") e una sequenza di otto bit serve al computer per identificare un qualsiasi carattere: lettera, numero o simbolo.

Perciò, l'unità di misura fondamentale della memoria di un computer è formata da una sequenza di otto bit, che equivale a un carattere ed è chiamata byte.

Si potrebbe obbiettare che il sistema binario richiede l'utilizzo di molte cifre anche per rappresentare numeri piccoli. Ecco, ad esempio, un confronto tra come vengono rappresentati alcuni numeri con il sistema decimale e con il sistema binario.

SISTEMA DECIMALE	SISTEMA BINARIO
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
15	1111
100	1100100

I computer sono fondamentalmente delle macchine abbastanza stupide, perché sono in grado di eseguire calcoli molto semplici solo se opportunamente programmati; ma, come abbiamo già detto, hanno l'enorme vantaggio di eseguirli in tempi rapidissimi.

Ogni computer trasforma perciò le cifre decimali che noi inseriamo in cifre binarie, effettua i calcoli richiesti e poi fornisce i risultati ritrasformando le cifre binarie in cifre decimali nello spazio di un millesimo di secondo, quindi il fatto che per riconoscere un numero impieghi più simboli di quanti siamo soliti utilizzarne noi non ha praticamente alcun effetto negativo.

Il sistema binario è così adatto ai computer, che viene utilizzato per conservare nella memoria degli elaboratori non solo dati numerici, ma qualsiasi tipo di informazioni: parole, immagini, suoni, ecc. Questo sistema di memorizzazione, che utilizza i numeri *0* e *1* è detto **digitale**.

Anche se i termini digitale e binario sono considerati spesso sinonimi, tecnicamente si tratta di due cose diverse, in quanto con "digitale" intendiamo un sistema descritto da valori non continui (al contrario di quanto accade nel sistema "analogico"), mentre con "binario" si indica un tipo di codifica che utilizza due soli simboli.

3.1.3 Convertire numeri dal sistema decimale a quello binario e viceversa

Per **convertire un numero dal sistema decimale a quello binario** si divide il numero per 2 e si prende nota del resto; poi si ripete la procedura con il risultato della divisione e così via sino a quando si ottiene 1 come risultato. Il numero binario corrispondente è costituito da 1 seguito dall'elenco dei resti ottenuti partendo dall'ultimo e arrivando al primo. Questa tecnica è detta **algoritmo della divisione ripetuta**.

Per essere più chiari, come al solito passiamo a un esempio pratico: applichiamo la procedura appena descritta per calcolare l'equivalente binario del numero decimale 67:

DIVISIONE	67:2=33	33:2=16	16:2=8	8:2=4	4:2=2	2:2=1	
RESTO	resto=1	resto=1	resto=0	resto=0	resto=0	resto=0	
NUMERO BINARIO (da leggere da destra a sinistra)	1	1	0	0	0	0	1

Il numero decimale 67, infatti, corrisponde in sistema binario al numero 1000011

Nel sistema numerico binario un numero pari termina sempre con un bit 0, mentre un numero dispari termina sempre con un bit 1.

Per **convertire**, invece, **un numero binario in numero decimale** bisogna moltiplicare le cifre che compongono il numero binario per le potenze crescenti di 2 iniziando a destra con il fattore 2^0 (che per convenzione vale 1) e poi proseguendo verso sinistra con 2^1 (che vale 2) 2^2 (che vale 4) 2^3 (che vale 8) 2^4 (che vale 16) 2^5 (che vale 32) e così via.

Applichiamo questa procedura per riconvertire il numero binario 100011:

NUMERO BINARIO	1	0	0	0	0	1	1
MOLTIPLICAZIONE	1×2^6	0×2^5	0×2^4	0×2^3	0×2^2	1×2^1	1×2^0
RISULTATO(da sommare)	64	0	0	0	0	2	1

Il numero binario 1000011 corrisponde, infatti, al numero decimale 67
(vale a dire 64 + 2 + 1).

3.1.4 Rappresentazione dei dati in memoria

La memoria del computer, come abbiamo detto, si basa su gruppi di 8 bit, detti byte, che rappresentano il numero di bit trattati contemporaneamente.

Con l'evoluzione dei Sistemi di Elaborazione, i computer oggi lavorano su "parole" più lunghe, 2 byte (word), 4 byte (double word) o addirittura 8 byte (quad word), e la lunghezza della parola è uno dei fattori che determinano la potenza di un elaboratore.

Attraverso queste sequenze di bit vengono rappresentati numeri e caratteri alfanumerici, secondo metodi differenti.

NUMERI INTERI

I numeri interi senza segno vengono rappresentati semplicemente trasformando il valore decimale in binario, ed eventualmente aggiungendo zeri non significativi nei bit a sinistra, come mostrato precedentemente.

Se invece il numero da rappresentare è un intero con segno, bisogna tenere conto anche dei numeri negativi, si utilizza quindi un metodo chiamato “complemento a 2”.

Si usano in genere 16 bit, di cui il più significativo (quello a sinistra) è il bit del segno, che avrà valore 0 per i numeri positivi (segno +) e 1 per quelli negativi (segno -).

Resteranno dunque solo 15 bit per i numeri, quindi potremo rappresentare valori positivi da 0 a +32767.

Per poter conservare la possibilità di effettuare operazioni tra i numeri, i numeri negativi verranno convertiti con il complemento a 2:

- Si trasforma il modulo del numero in binario secondo la regola precedente
- Si sostituisce alle cifre 0 il valore 1 e viceversa
- Si aggiunge 1

In questo modo si ottiene un numero binario che, sommato al suo opposto, dà come risultato 0.

Facciamo un esempio:

$43_{10} = 101011_2$

Con il complemento a 2 abbiamo:

0000000000101011

1111111111010100 +

1 =

1111111111010101

$-43_{10} = 1111111111010101_2$

Se sommiamo abbiamo:

0000000000101011 +

1111111111010101 =

10000000000000000

Il primo bit a sinistra va trascurato, perché eccede i 16 bit disponibili, dunque abbiamo come risultato il valore 0.

Il minimo valore negativo rappresentabile è -32768

NUMERI REALI

Per i numeri reali la rappresentazione è un po' più complicata. Abbiamo la rappresentazione in virgola fissa, in cui vengono usati un certo numero di byte per la parte intera (compreso il segno) e un certo numero per la parte decimale, codificate come numeri interi.

Per la rappresentazione in virgola mobile, invece, si parte da quella che noi conosciamo come notazione esponenziale normalizzata, usata quando si devono rappresentare numeri molto grandi o molto piccoli.

Ad esempio, il numero 3240000 si rappresenta come 0.324E-7, cioè 0,324 X 10^{-7}, quindi in memoria vengono salvate, trasformate in binario, la "mantissa", cioè la parte dopo la virgola, e il suo segno, e l'esponente con il suo segno.

3.1.5 Rappresentare i caratteri in forma binaria. Definire le nozioni di bit e di byte

Come abbiamo detto, **il sistema binario** è **utilizzato per conservare nella memoria degli elaboratori non solo dati numerici, ma qualsiasi tipo di informazione in formato digitale**: parole, immagini, suoni, ecc.

Quindi, se noi premiamo sulla tastiera di un computer il tasto A, il computer traduce questa nostra azione in un insieme di bit che rappresenta il codice associato al carattere A e questo stesso insieme di bit permetterà, quando necessario, la visualizzazione del carattere A sul monitor o sulla stampante.

Attualmente si usa un codice a 8 **bit**, perché la memoria dei calcolatori elettronici è generalmente organizzata in celle di 8 bit ciascuna, ognuna delle quali è detta **byte**. Il byte è l'unità di misura per diverse quantità in ambito informatico: la capacità di archiviazione di una penna USB o di un hard disk, la memoria di un computer, la velocità di trasferimento dei dati, ecc. (v. punto 4.3.2 del Modulo Computer Essentials).

Il metodo di codifica più diffuso e che costituisce uno standard internazionale è chiamato codice ASCII standard, dalle iniziali di *American Standard Code for Information Interchange* che significa "codice standard americano per lo scambio di informazioni".

Nella tabella ASCII standard (fig. a pagina successiva) troviamo 128 combinazioni diverse, che rappresentano 26 lettere minuscole, 26 lettere maiuscole, 10 cifre decimali, segni di punteggiatura (virgola, punto, punto e virgola, due punti, parentesi, ecc.), simboli matematici (più, meno, per, diviso, maggiore, minore, uguale, ecc.), altri caratteri per uso speciale (asterisco, cancelletto, simbolo del dollaro, ecc.) e anche istruzioni particolari come "vai a capo", "torna indietro di un carattere", "vai all'inizio della linea di testo" e così via.

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Dal momento che per rappresentare 128 caratteri sono sufficienti 7 bit e che bisognava invece arrivare ad 8 bit, ad ogni carattere del codice ASCII standard è stato aggiunto uno 0 iniziale. In una configurazione binaria il bit più a sinistra si chiama, infatti, **LSB**, dalle iniziali di “Least Significant Bit”, che significa “bit meno significativo. Al contrario il bit più a destra si chiama **MSB**, dalle iniziali di “Most Significant Bit”, ossia “bit più significativo”.

Utilizzando il codice ASCII standard possiamo rappresentare ogni testo in forma binaria, procedendo carattere per carattere. Proviamo, ad esempio, a rappresentare in forma binaria il titolo di questo libro (ICDL *piu’*):

I	C	D	L		P	i	U	’
01001001	01000011	01000100	01001100	00100000	01110000	01101001	01110101	00100111

Avrete forse notato che non abbiamo utilizzato la lettera “u” accentata (ù), ma l’abbiamo apostrofata (u’). Questo perché il ASCII standard non rappresenta le vocali accentate (che non si usano nella lingua inglese) e altri simboli di uso meno comune.

Per tale motivo sono stati successivamente creati prima un **codice ASCII esteso** costituito da 256 caratteri e poi un codice **Unicode** (pr. ùnicod) basato non su 8 ma su 16 bit e quindi in grado di rappresentare ben 65.536 caratteri e di conseguenza anche lingue specifiche come il cinese, il greco, il cirillico, ecc. I primi 256 caratteri dell'Unicode corrispondono però al codice ASCII per assicurare la compatibilità tra i due codici.

3.1.4 Il sistema numerico esadecimale

In Informatica è molto usato anche il **sistema numerico esadecimale**, spesso abbreviato in **esa** oppure in **hex**. Mentre il sistema binario è basato su 2 numeri e quello decimale su 10 numeri, il sistema esadecimale - come si può capire dal suo nome – utilizza 16 simboli: da 0 a 9 per le prime dieci cifre e poi le lettere da A a F per le successive 6 cifre.

Ecco una tabella che confronta le rappresentazioni decimale, binaria ed esadecimale.

SISTEMA DECIMALE	SISTEMA BINARIO	SISTEMA ESADECIMALE
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
15	1111	F
100	1100100	64

La diffusione del sistema esadecimale in informatica è dovuta principalmente alla relazione diretta che c'è tra una cifra esadecimale e quattro cifre binarie, ragion per cui è possibile esprimere un byte con sole due cifre esadecimali, invece che con tre decimali.

3.2 Elementi di logica

3.2.1 Logica e proposizioni

Come abbiamo detto, il principio di base di un computer è il bit, la cifra binaria, che corrisponde all'unità di memorizzazione. Ma questo concetto di valore binario ha anche un significato più operativo, applicando i principi della logica matematica.

Nell'antica Grecia, la "logica" era una parte della filosofia, e proprio Aristotele se ne occupò in maniera approfondita, partendo dal concetto di "Proposizioni". In seguito il matematico George Boole tradusse questi principi fondamentali in termini matematici, la cosiddetta "Algebra booleana" che è alla base dei principi che regolano l'informatica moderna.

Una proposizione è una affermazione di cui, in ogni momento, si può stabilire se sia vera o falsa.

Alle proposizioni si possono applicare delle funzioni logiche, unarie o binarie, che consentono la valutazione di enunciati composti partendo dal valore delle proposizioni che lo compongono.

Poiché i valori logici possibili sono soltanto due, vero o falso, vengono rappresentati con le cifre binarie: 0 per falso, 1 per vero.

A ciascuna funzione si possono associare delle "Tavole di verita" che ne consentono la valutazione.

3.2.2 I connettivi logici: AND, OR, NOT

CONGIUNZIONE AND: quando due proposizioni sono collegate attraverso il connettivo AND il risultato è vero se entrambe sono vere, secondo la tavola, datxe due proposizioni *p* e *q*:

p	q	p AND q
0	0	0
0	1	0
1	0	0
1	1	1

Quindi, ad esempio, se *p* = "papà guarda la TV" e *q* = "papà siede sul divano", avremo che *p* AND *q* sarà vero, cioè avrà valore 1, solo se papà guarda la tv e siede sul divano contemporaneamente.

Facciamo un altro esempio: se lungo un tubo dell'acqua vi sono due rubinetti collegati in serie (vale a dire in modo dipendente l'uno dall'altro, come succede nelle nostre case per la chiave di arresto generale dell'acqua e per un qualsiasi rubinetto interno), perché l'acqua scorra occorrerà che siano aperti il primo AND il secondo rubinetto.

DISGIUNZIONE OR: quando due proposizioni sono collegate attraverso il connettivo OR il risultato è vero se almeno una delle proposizioni è vera, secondo la tavola, date due proposizioni *p* e *q*:

p	q	p OR q
0	0	0
0	1	1
1	0	1
1	1	1

Quindi, ad esempio, se *p* = "Giulio guarda la TV" e *q* = "Giulio mangia la minestra", avremo che *p* OR *q* sarà falso, cioè avrà valore 0, solo se Giulio non guarda la tv e non mangia la minestra.

Se facciamo riferimento all'esempio precedente, se i due rubinetti sono collegati in parallelo (vale a dire in modo indipendente l'uno dall'altro, come accade normalmente nelle nostre case per il rubinetto del lavello in cucina e quello del lavandino nel bagno), basterà che sia aperto il primo OR il secondo perché l'acqua scorra.

NEGAZIONE NOT: il NOT inverte il valore di verità della proposizione: se era vera il risultato è falso e viceversa.

In questo caso la tavola di verità è più semplice:

p	NOT p
0	1
1	0

Ad esempio, se nell'archivio dati di un comune voglio cercare persone straniere, utilizzerò un comando del tipo "NOT italiana" per individuare nel campo "nazionalità" le persone di nazionalità straniera, perché il computer mi fornirà come risultato l'elenco di tutte le persone per le quali nel campo "nazionalità" NON è riportato "italiana".

3.3 Linguaggi e programmi

3.3.1 Linguaggio naturale e linguaggi di programmazione

Con il termine "linguaggio" si intende un sistema di simboli e regole che permette a persone che conoscono quel linguaggio di comunicare tra loro. Nel caso degli esseri umani, il **linguaggio naturale** è costituito dalle parole e dalle regole grammaticali.

I computer, invece, utilizzano il sistema binario e attraverso esso hanno un proprio "linguaggio" – molto diverso dal nostro – che è chiamato **codice macchina o codice binario o linguaggio a basso livello**, sostanzialmente costituito da una serie di comandi (che potremmo paragonare alle nostre frasi) scritti in modo da poter essere interpretati senza dar luogo a nessun equivoco e chiamati *istruzioni macchina* o semplicemente *istruzioni*.

Quando una persona utilizza un computer, si trovano di fronte due "esseri" che parlano linguaggi totalmente diversi. Per questo motivo sono stati creati dei **linguaggi di programmazione o linguaggi ad alto livello**, più semplici da usare. Essi possono essere paragonati a una specie di "interprete" che si pone tra l'uomo e la macchina, traducendo il linguaggio dell'essere umano in quello del computer e viceversa.

Esistono migliaia di linguaggi di programmazione: in passato erano molto utilizzati *Basic* (si pronuncia *bèsic*), *Cobol*, *C* (si pronunciano entrambi così come si leggono); attualmente tra i più diffusi vi sono *Java, C++*, *PHP* e *Python* (si pronunciano rispettivamente *giàva, cì-plas-plas*, *pì-àcca-pì* e *pàiton*). In genere utilizzano termini della lingua inglese per impartire le istruzioni al computer.

Un programma scritto in un linguaggio di programmazione viene detto *codice sorgente*, perché non può essere eseguito immediatamente da un computer, ma richiede una traduzione in *codice macchina*, traduzione che viene eseguita da un programma detto *compilatore* o *traduttore*.

3.3.2 Distinguere fra linguaggio macchina e linguaggi procedurali

Il **linguaggio macchina**, detto anche *linguaggio di basso livello*, è formato da lunghissime sequenze di *0* e di *1*, immediatamente interpretabili dal computer ma difficilmente comprensibili da un programmatore.

Per questo motivo, i programmatori adoperano dei linguaggi che procedono in maniera sequenziale (vale a dire con una serie di operazioni che si susseguono l'un l'altra), indicando man mano le varie procedure da svolgere, per questo motivo questi linguaggi vengono definiti **linguaggi procedurali**, oppure *linguaggi di alto livello*.

3.1.8 Scrivere algoritmi con l'uso dello pseudo-codice

Abbiamo già visto (nel punto 2.1.7) che un algoritmo può essere rappresentato graficamente attraverso un diagramma di flusso. Esso può anche essere rappresentato con uno **pseudo-codice**, che è un linguaggio a metà strada tra quello parlato e un linguaggio di programmazione ad alto livello.

La sua scrittura assomiglia alle istruzioni di un linguaggio di programmazione, perché sfrutta alcune **parole-chiavi** e l'**indentazione**. Le parole-chiavi che utilizziamo sono quelle necessarie ad un qualsiasi algoritmo:

INPUT (legge dei dati)

OUTPUT o **PRINT** (scrive dei dati)

IF... ELSE... (blocco di controllo)

WHILE ... o **FOR...** (blocco iterativo)

L'**indentazione** è una modalità di visualizzazione dei "Blocchi di codice", per cui le istruzioni vengono spostate verso destra di un certo numero di caratteri (in genere 4).

Rappresentiamo l'algoritmo del massimo tra 3 numeri interi utilizzando lo pseudo-codice:

PSEUDO-CODICE	OSSERVAZIONI
`INPUT a,b,c`	Legge i valori di a,b,c
`if a<b then`	se a<b allora (confronta prima **a** con **b**)
`Massimo=b`	Il valore Massimo è **b**
`else`	altrimenti
`Massimo=a`	Il valore Massimo è **a**
`if c>Massimo then`	se c>Massimo allora (confrontiamo il valore ottenuto tra i primi due, Massimo, col terzo, **c**)
`Massimo=c`	Il Massimo è **c**
`Print Massimo`	Stampa sul video il valore contenuto nella variabile Massimo

Come esempio di “blocco iterativo”, esaminiamo il problema seguente:

Assegnato un numero intero *N*, sommare tutti gli interi compresi tra *1* ed N. (Il flow-chart è tra gli esercizi svolti allegati)

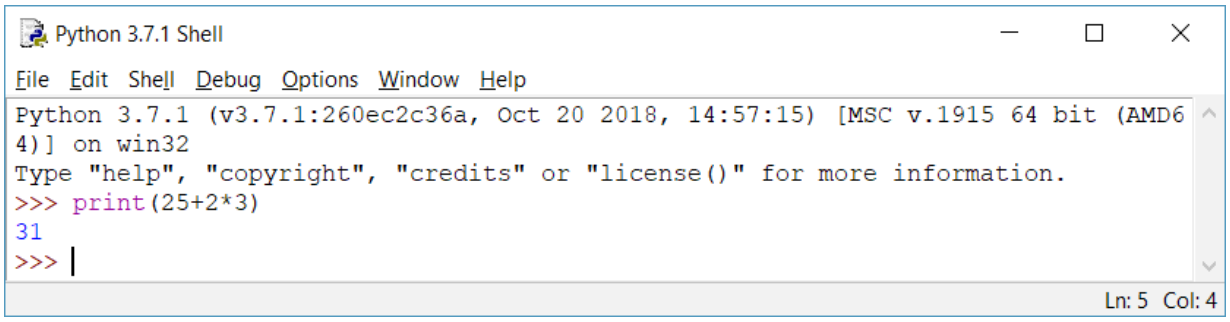
Un possibile algoritmo, scritto in pseudo-codice, che risolve il problema è il seguente:

PSEUDO-CODICE	OSSERVAZIONI
`INPUT N`	Legge il valore di N
`contatore=0 ; somma=0`	Pone a zero le variabili contatore e somma
`while contatore<N  ripeti`	Finché la variabile contatore è minore di N
`  contatore=contatore+1`	Incrementa il contatore di una unità
`  somma=somma+contatore`	Incrementa il valore della variabile somma con contatore
`Print somma`	Stampa sul video il valore contenuto nella variabile somma

3.3.4 Scrivere programmi in Python

Python è un **linguaggio di programmazione ad alto livello**, rilasciato pubblicamente per la prima volta nel 1991 dal suo creatore, il programmatore olandese **Guido van Rossum**. Python è un linguaggio semplice e didatticamente efficace, che può essere utilizzato anche come interprete: può analizzare ed eseguire ogni singola istruzione scritta. I *blocchi di istruzioni* sono indentati come nello pseudo-codice, per cui è semplice passare dallo pseudo-codice al programma scritto in python.

Possiamo scaricare il programma dal sito www.python.org; una volta eseguito avremo la seguente schermata, in cui abbiamo digitato l’istruzione `print(25+2*3)`, dove il simbolo * rappresenta la moltiplicazione; possiamo verificare come l’interprete risponderà istantaneamente.

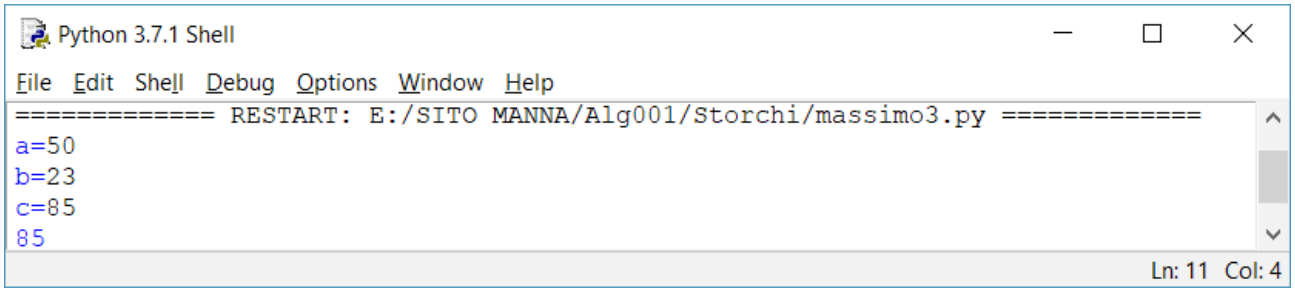


A titolo esemplificativo, trasformiamo i due algoritmi scritti in pseudo-codice in due programmi python. Il primo riguarda il calcolo del massimo tra tre numeri interi:

PSEUDO-CODICE	PROGRAMMA PYTHON
`INPUT a,b,c`	a=int(input(“a=”)) b=int(input(“b=”)) c=int(input(“c=”))
`if a<b then`	if a<b:
`  Massimo=b`	Massimo=b
`else`	else:
`  Massimo=a`	Massimo=a
`if c>Massimo then`	if c>Massimo:
`  Massimo=c`	Massimo=c
`Print Massimo`	print(Massimo)

Andando su **File** e poi su **New File**, si può digitare il testo del programma, salvarlo con un nome a piacere, per esempio *"massimo.py"*, e mandarlo in esecuzione con **Run**.

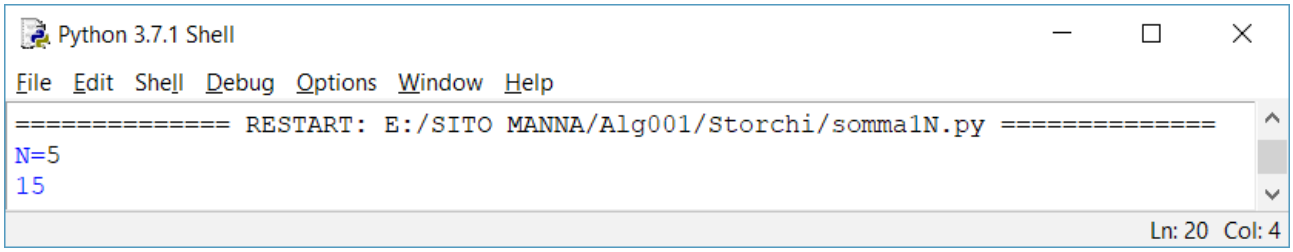
La figura seguente illustra il caso in cui si inseriscono i tre numeri 50, 23, 85.



Secondo riguarda la somma degli interi tra 1 ed N:

PSEUDO-CODICE	PROGRAMMA PYTHON
INPUT N	N=int(input("N="))
contatore=0 ; somma=0	contatore=somma=0
while contatore<N ripeti	while contatore<N:
contatore=contatore+1	contatore=contatore+1
somma=somma+contatore	somma=somma+contatore
Print somma	print(somma)

L'immagine seguente restituisce il caso in cui si pone N=5:



Come possiamo notare, sono minime le differenze tra un algoritmo scritto in pseudo-codice ed un programma scritto in python. Analizziamole brevemente partendo dall'istruzione INPUT.

```
N=input("N=")
```

scrive sul video il "suggerimento" proposto all'interno delle parentesi tonde ed attende che l'utente digiti dei dati sulla tastiera; tali dati vengono inseriti nella variabile N. La funzione **int** trasforma il dato inserito in un numero intero in modo che ad esso possano essere applicate tutte le relative operazioni.

Un altro aspetto fondamentale, non trascurabile, è la terminazione con i due punti ":" di tutte le istruzioni di controllo ed iterative

```
if a<b:     if c>Massimo:     else:     while contatore<N:
```

dopo i due punti ci si aspetta un **"blocco di codice"** che deve essere indentato con 4 spazi.

Ulteriori informazioni sul linguaggio Python sono reperibili sul sito **www.imparafacendo.it**.

Esercizi

1. Il sistema binario utilizza esclusivamente i numeri:

A 0 e 1.
B 0 e 2.
C 1 e 2.
D 2 e 2.

Le informazioni immagazzinate nella memoria di un computer:

A sono espresse sotto forma di numeri del sistema decimale.
B sono espresse sotto forma di numeri del sistema binario.
C sono trasformate in una sequenza di 1 e 2.
D sono trasformate in una sequenza di 0 e 2.

3. A cosa corrisponde un byte?

A A un numero binario.
B A una coppia di numeri binari.
C A 4 numeri binari.
D A 8 numeri binari.

4. Nei programmi in linguaggio macchina:

A la sintassi è identica al modo di parlare umano.
B le istruzioni sono impartite utilizzando le lettere dell'alfabeto.
C sia le istruzioni sia i dati sono scritti sotto forma di stringhe di bit.
D le istruzioni variano il loro aspetto in funzione dell'elaboratore sul quale dovrebbero essere eseguite.

5. Qual è il ruolo dei connettivi logici nell'informatica?

A Consentono la traduzione di un programma sorgente in un linguaggio macchina.
B Mettono in relazione le istruzioni contenute nei linguaggi destinati al computer.
C Indicano il nome del programma.
D Indicano lo scopo del programma.

6. Quando due istruzioni sono collegate attraverso il connettivo logico OR:

A il risultato è vero se almeno una delle due affermazioni contenute nelle istruzioni è vera.
B il risultato è vero se entrambe le affermazioni contenute nelle istruzioni sono vere.
C il risultato è falso se entrambe le affermazioni contenute nelle istruzioni sono vere.
D viene invertito il valore di verità dell'istruzione: se era vera il risultato è falso e viceversa.

7. Cosa si intende per linguaggio procedurale?

A È un linguaggio di programmazione in cui le azioni da compiere vengono specificate sotto forma di espressioni o funzioni.

B È un linguaggio di programmazione in cui le istruzioni per il computer sono espresse indicando la sequenza di operazioni che devono essere svolte.

C È un sistema di programmazione grafica in cui l'utilizzatore associa le icone dei blocchi funzionali con linee che rappresentano scambi di dati.

D È un linguaggio di programmazione in cui le istruzioni per il computer riguardano l'oggetto della programmazione.

8. Oltre che con un diagramma di flusso, in quale altro tra i seguenti modi può essere rappresentato un algoritmo?

A Adoperando un linguaggio naturale.

B Con il codice ASCII.

C Con uno pseudolinguaggio.

D Utilizzando i soli connettivi logici.

9. Quando due istruzioni sono collegate attraverso il connettivo logico AND:

A il risultato è vero se almeno una delle due affermazioni contenute nelle istruzioni è vera.

B il risultato è vero se entrambe le affermazioni contenute nelle istruzioni sono vere.

C il risultato è falso se entrambe le affermazioni contenute nelle istruzioni sono vere.

D viene invertito il valore di verità dell'istruzione: se era vera il risultato è falso e viceversa

10. Il sistema numerico esadecimale:

A è basato su 2 numeri

B è basato su 12 numeri

C utilizza lettere e numeri

D utilizza i simboli da 0 a 6

Scratch è un linguaggio di programmazione creato per consentire anche agli utenti più giovani e inesperti un facile approccio al coding. Creato nel 2003 nella sua versione desktop da un gruppo di ricercatori del M.I.T. (Massachusetts Institute of Tecnology) di Boston, dal 2007 è diventato una piattaforma online dove gli utenti possono condividere i loro progetti, consentendo addirittura uno sviluppo cooperativo.

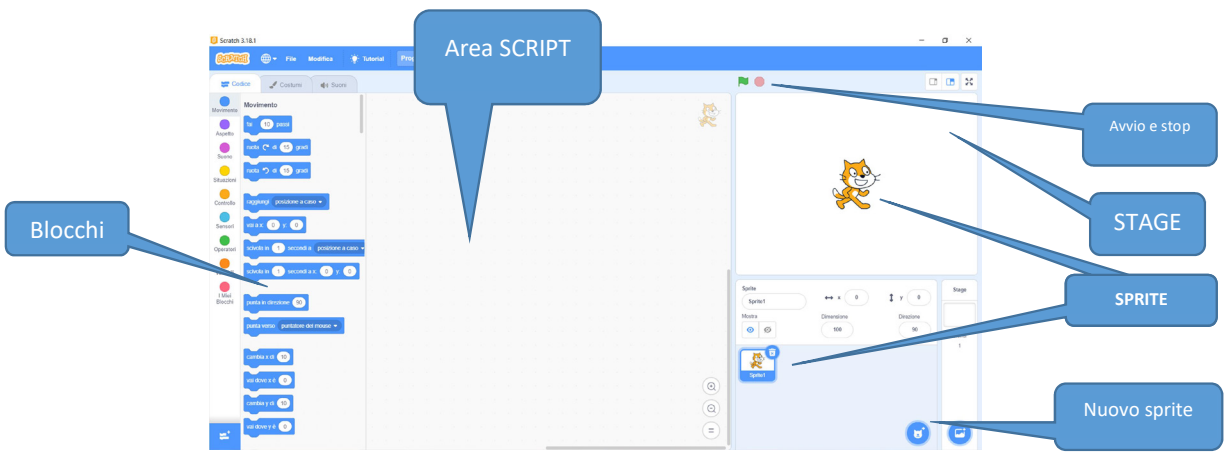
La grande innovazione di Scratch, poi utilizzata da altre piattaforme, è stata quella di sostituire le istruzioni con blocchi colorati che vanno incastrati tra di loro come in un puzzle. I colori dei blocchi indicano la TIPOLOGIA di istruzioni (controllo, matematiche, ecc...) per cui si possono realizzare non solo semplici applicazioni, ma anche veri e propri programmi, trasformando gli algoritmi progettati come abbiamo visto in precedenza.

Ambiente di sviluppo

L'ambiente di programmazione è grafico e intuitivo, e consente di utilizzare animazioni e immagini predefinite, o creare elementi grafici personalizzati.All'apertura dell'applicazione, che esiste sia in versione desktop sia online, ci troviamo di fronte tre aree:

- lo **STAGE**, a destra, rappresenta la finestra entro la quale sarà eseguito il nostro programma, e qui compariranno gli oggetti grafici che in esso abbiamo voluto includere. Al disotto vediamo un'area attraverso la quale possiamo gestire gli "Sprite" inseriti nel programma, è c'è la possibilità di eliminarli, modificarli o inserirne di nuovi. Nella parte superiore dello STAGE ci sono due pulsanti: la "Bandiera verde" per avviare i programmi e il pulsante di STOP per interrompere l'esecuzione.
- L'area **BLOCCHI**, a sinistra, che contiene i blocchi necessari a costruire il codice di programmazione
- L'area **SCRIPT**, al centro, dove saranno inseriti i blocchi che costituiscono il programma.

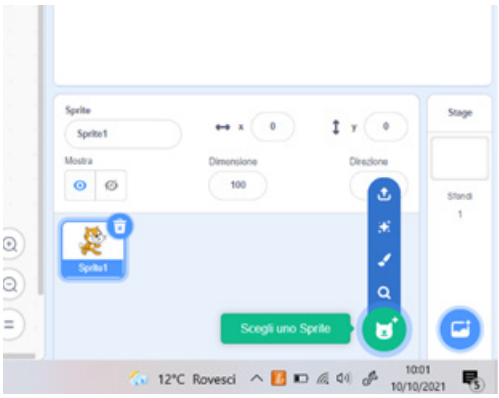
Nella parte alta della finestra c'è la Barra dei menu, con i consueti comandi di apertura e salvataggio dei file.



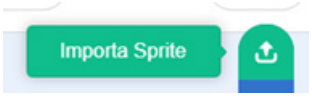
SPRITE

Gli oggetti di base di un programma Scratch sono degli elementi grafici detti SPRITE, che si possono inserire selezionandoli dalla libreria o importandone l'immagine dalla propria galleria, o addirittura creando un'immagine nuova con il semplice editor di immagini incluso nell' applicazione.

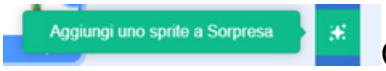
Gli Sprite non sono solo degli oggetti decorativi, ma possono essere animati attraverso diversi "costumi", e ad essi si possono associare azioni e comportamenti.



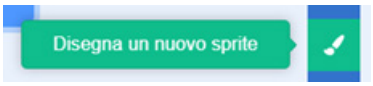
Per creare un nuovo sprite si clicca sul pulsante "Nuovo sprite", quindi è possibile scegliere le diverse opzioni dal menù contestuale:



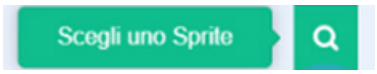
Caricare un'immagine dal proprio dispositivo



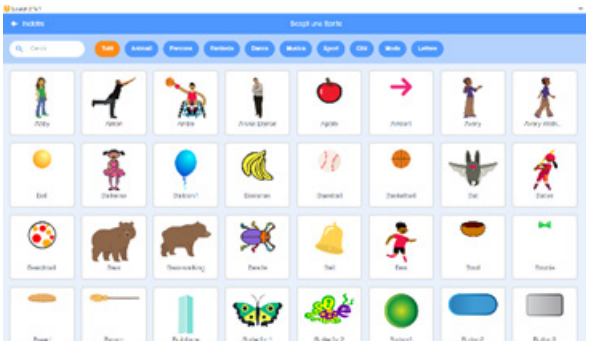
Creare uno sprite a sorpresa



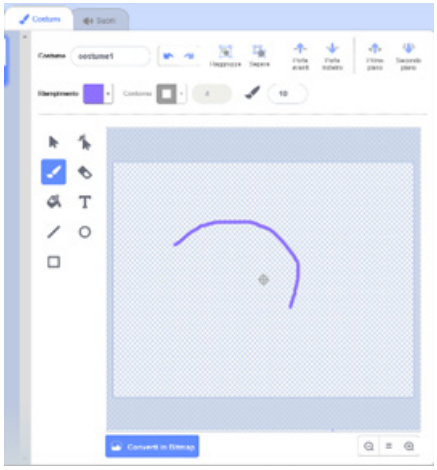
Disegnare uno sprite con l'apposito editor



Scegliere uno Sprite dalla libreria



Libreria degli Srite



Editor degli Srite

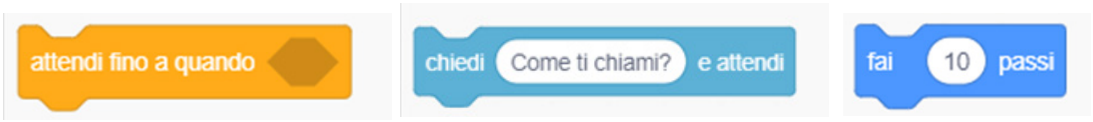
Blocchi

Costituiscono le vere e proprie istruzioni del programma, di colore diverso a seconda del tipo di istruzioni che rappresentano: il tipo di istruzione viene selezionato nella colonna a destra “Categorie”.

Anche la forma dei blocchi ha la sua importanza, e determinano come i blocchi possono essere posizionati e combinati con gli altri blocchi.

I blocchi possono essere di tre tipi principali:

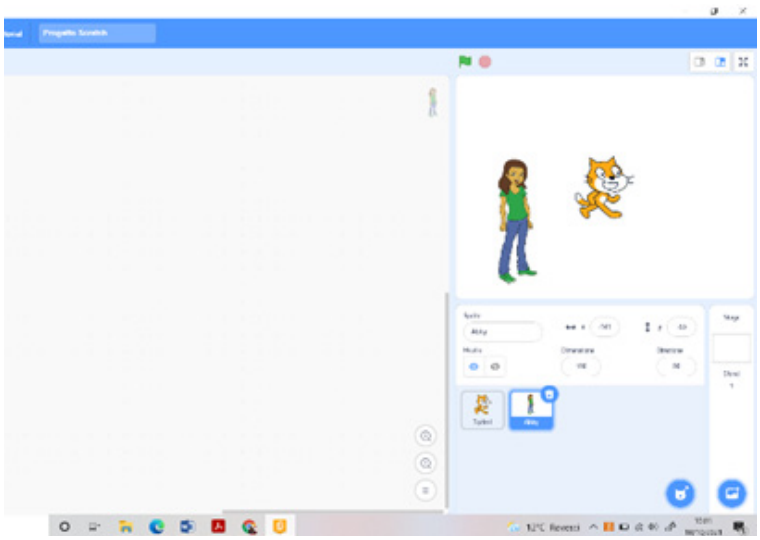
STACK: hanno la forma di pezzi di un puzzle, ad indicare che possono essere inseriti al disotto o al disopra degli altri blocchi



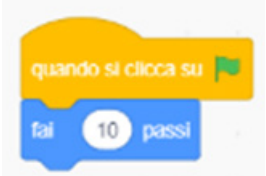
HAT: sono blocchi che possono essere inseriti sopra altri blocchi ma non sotto

REPORTER: hanno forme specifiche e vanno utilizzati solo se inseriti all’interno di altri blocchi

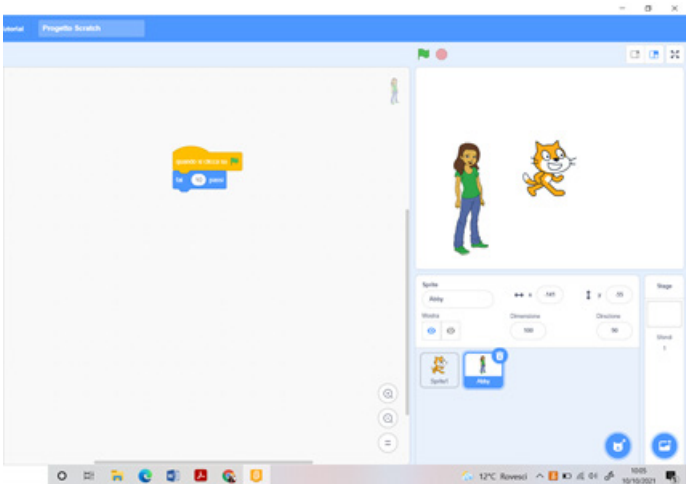
Creare un programma in Scratch



Cliccando sull’icona dello sprite nell’area a sinistra, è possibile inserire i blocchi di istruzioni collegati a quello sprite nell’area dello script, o del codice, al centro

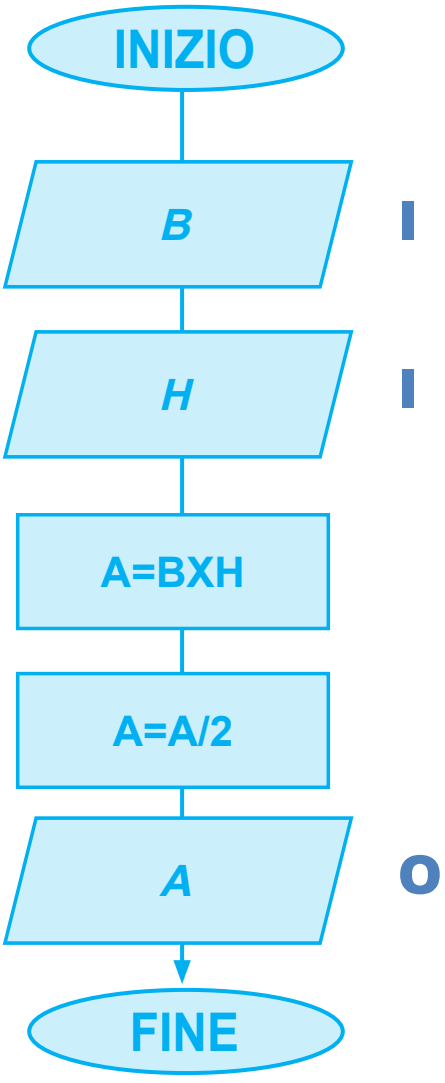


Ad es. se inserisco i blocchi in figura quando è selezionata la sprite “Abby” (per verificare a quale sprite si riferiscono i blocchi basta controllare l’immagine schiarita che compare nell’area Script in alto a destra)

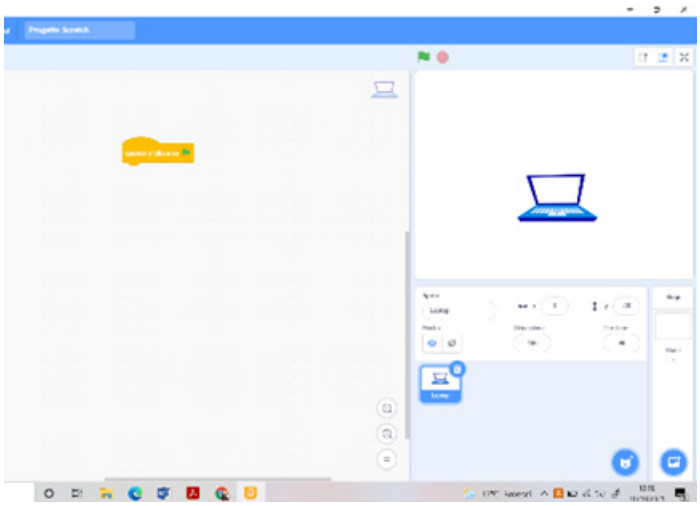


Nell’esempio sopra sarà lo sprite abby a muovere dieci passi quando si clicca sulla bandierina verde.

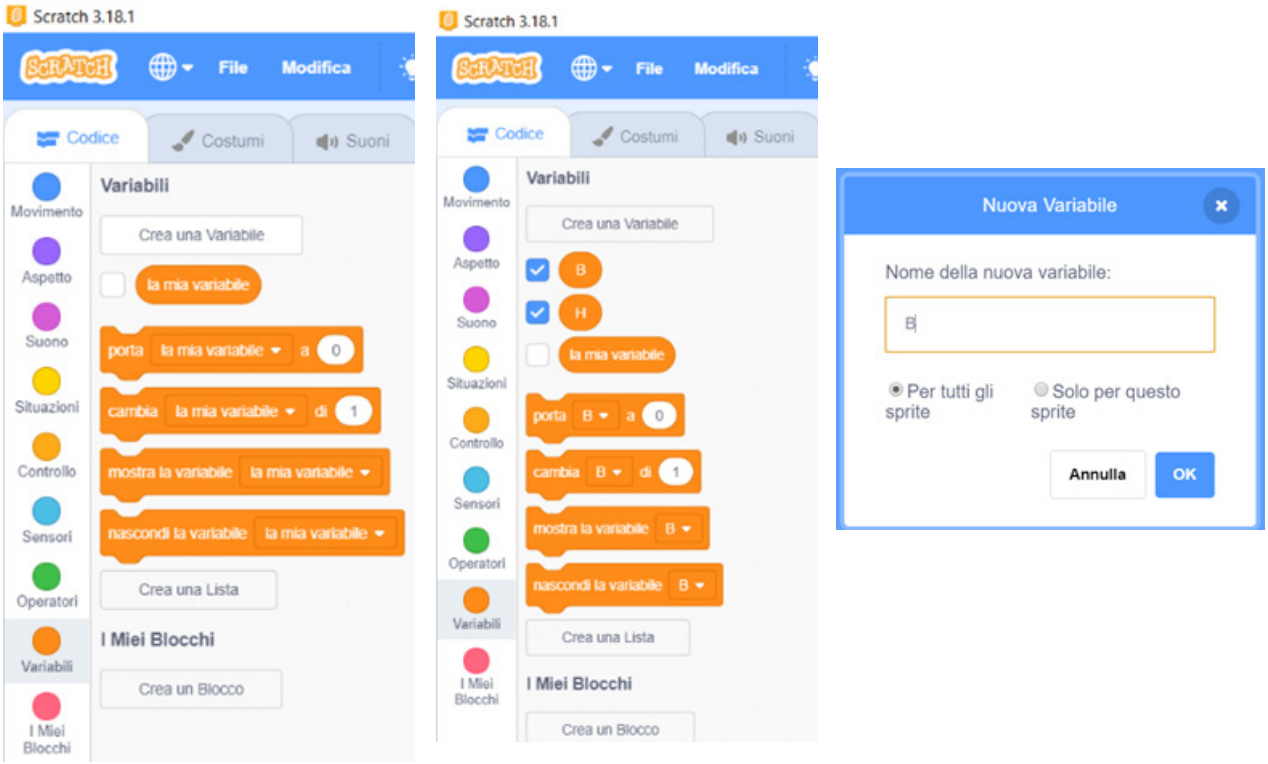
Vediamo il semplice algoritmo area di un triangolo visto precedentemente



Abbiamo inserito come sprite un computer. L'algoritmo andrà in esecuzione al cliccare sulla bandierina verde, tale indicazione è data dall'istruzione apposita che selezioniamo dalla categoria "situazioni".



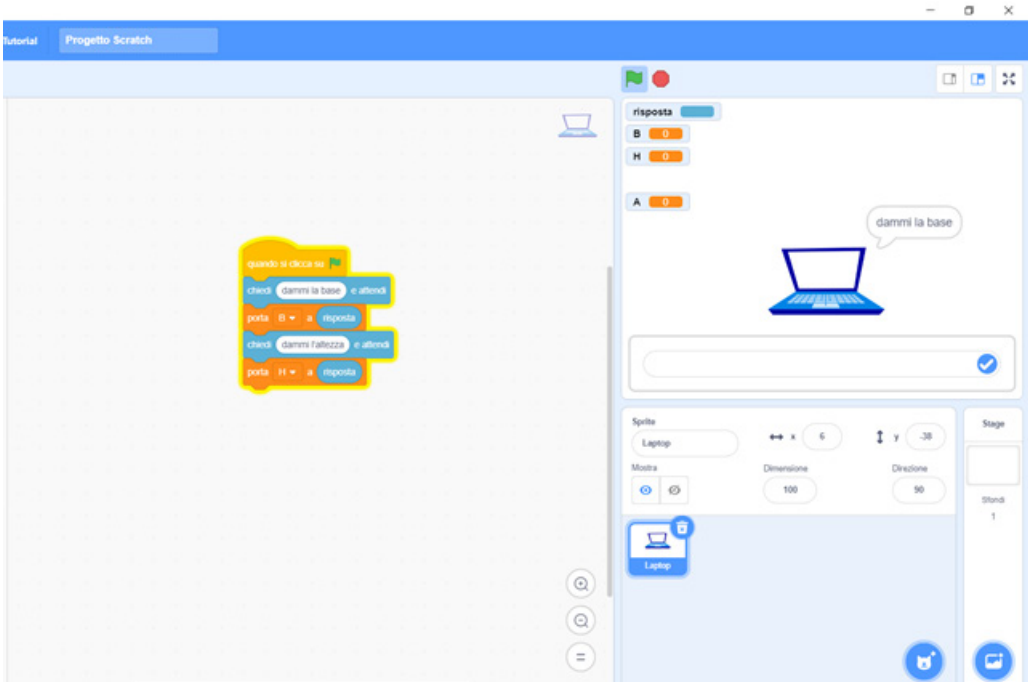
Prima di inserire le istruzioni, dobbiamo creare le variabili nella categoria "variabili".



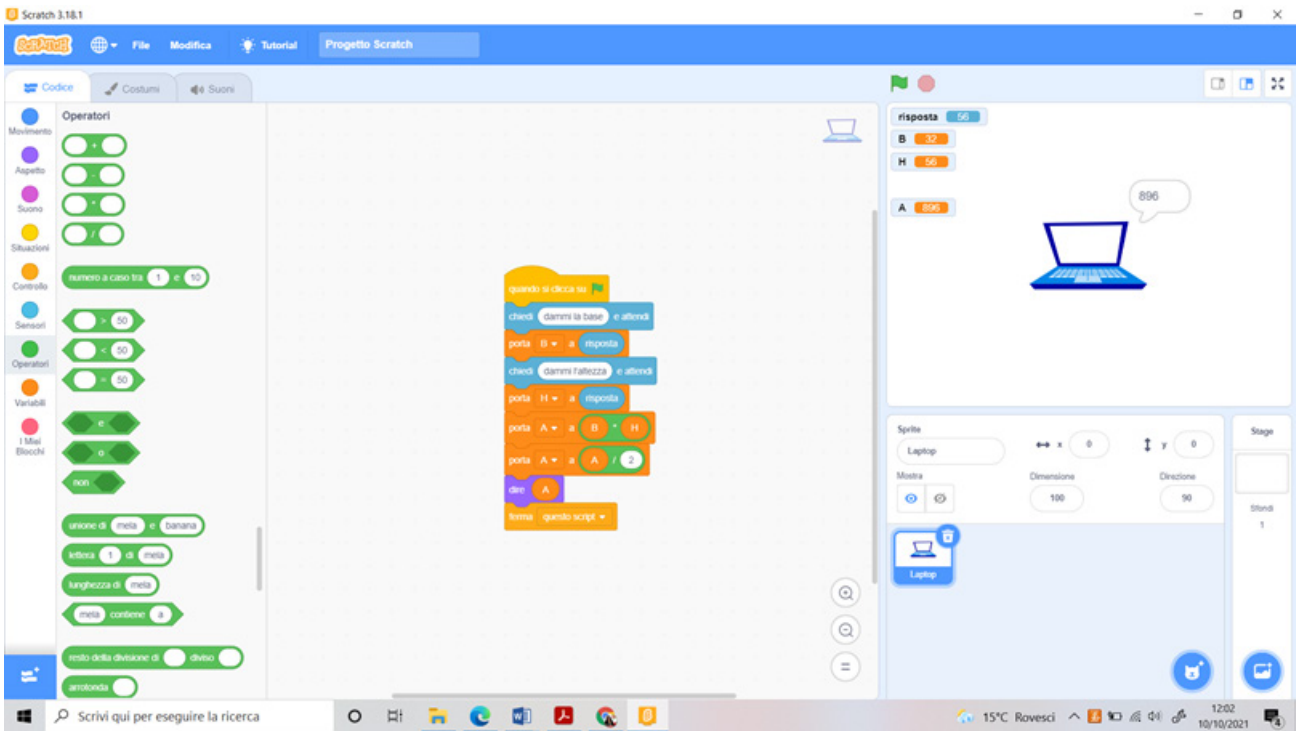
Se accanto al nome delle variabili è selezionata la casella di controllo, il loro contenuto sarà visibile sullo stage. È consigliabile mantenere la spunta durante la fase di progettazione del programma, per controllare il valore delle variabili, per poi toglierla quando si rilascia la versione definitiva.

Inseriamo quindi le istruzioni di input, le scegliamo nella categoria “sensori”, l’istruzione è “chiedi e attendi”; selezionando la risposta, questa sarà visibile sullo stage. Nello spazio destinato al testo della domanda scriviamo “dammi la base” e “dammi l’altezza”.

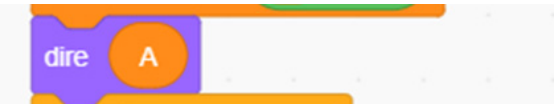
Dopo aver ricevuto la risposta di ciascuna istruzione, questa va salvata nella variabile corrispondente, selezionando l’apposita istruzione dalla categoria “variabili”.



A questo punto dobbiamo eseguire le operazioni, utilizzando l'assegnazione e selezionando gli operatori dalla categoria "operatori".



Per le istruzioni di output si utilizzano i blocchi della categoria "aspetto", che appunto definiscono ciò che deve apparire nello stage. Si seleziona "dire…".



Dalla categoria "controllo" si seleziona l'istruzione che ferma l'esecuzione del programma.

Per quanto riguarda le altre categoria, abbiamo "movimento", che contiene le istruzioni per far spostare gli sprite all'interno dello stage: le posizioni sono indicate dalle coordinate orizzontali, x, e verticali, y.

Il sistema funziona come il Piano cartesiano, quindi le coordinate 0,0 corrispondono alla posizione al centro dello stage. Nella categoria "aspetto" oltre alle istruzioni di output, ci sono comandi per modificare la grafica e le dimensioni degli sprite.

La categoria "suono" serve per gestire l'output audio (se il dispositivo è dotato di altoparlanti)

La categoria "situazioni" gestisce gli *eventi* che si possono verificare.

NB: se si seleziona la lingua inglese, la descrizione dei blocchi è molto vicina alle istruzioni dei più comuni linguaggi di programmazione.