Laboratorio di Informatica

Lezione 07/11/2019

ALGORITMO

Il termine deriva dal nome del matematico persiano Abū Jaʿfar Muhammad ibn Mūsā al-Khwārizmī, nato nel 780 circa e autore di numerosi trattati scientifici. Un libro di al-Khwārizmī, tradotto in latino molti anni dopo, cominciava con le parole "Dixit Algorithmi". Il nome dell'autore era stato evidentemente latinizzato, e da qui ebbe origine il termine algoritmo.

Viviamo in un mondo dominato dagli algoritmi, e gli algoritmi regolano oramai, spesso anche a nostra insaputa, molte delle nostre azioni quotidiane.

Il telefonino sembra sapere molte cose di noi, tanto da suggerirci qualche **evento** a cui potremmo essere interessati

la pubblicità che riceviamo spesso è attentamente **profilata** su di noi e sui **nostri comportamenti**.

Molte delle nostre **relazioni sociali** e **professionali** si svolgono sui **social network**, come Facebook, Twitter o LinkedIn. Ancora una volta, i social network **utilizzano algoritmi** molto sofisticati per decidere cosa farci vedere nei loro **news feed**, in base a quello che sanno di noi. Anche **Amazon** o **Netflix** ci suggeriscono libri, film o altri oggetti che potrebbero piacerci, e molto spesso **conoscono** i nostri **gusti** molto meglio dei nostri vecchi amici.

Ma che cos'è esattamente un algoritmo?

Il concetto di algoritmo

- **Un algoritmo è una sequenza di passi necessari per risolvere un problema o eseguire una computazione**

- **In alcuni casi, lo stesso problema/computazione può essere risolto in modi diversi, ai cui corrispondono diversi algoritmi**

- **Un programma non è altro che la descrizione di un algoritmo scritta nel linguaggio di programmazione scelto.**

Diagrammi di flusso

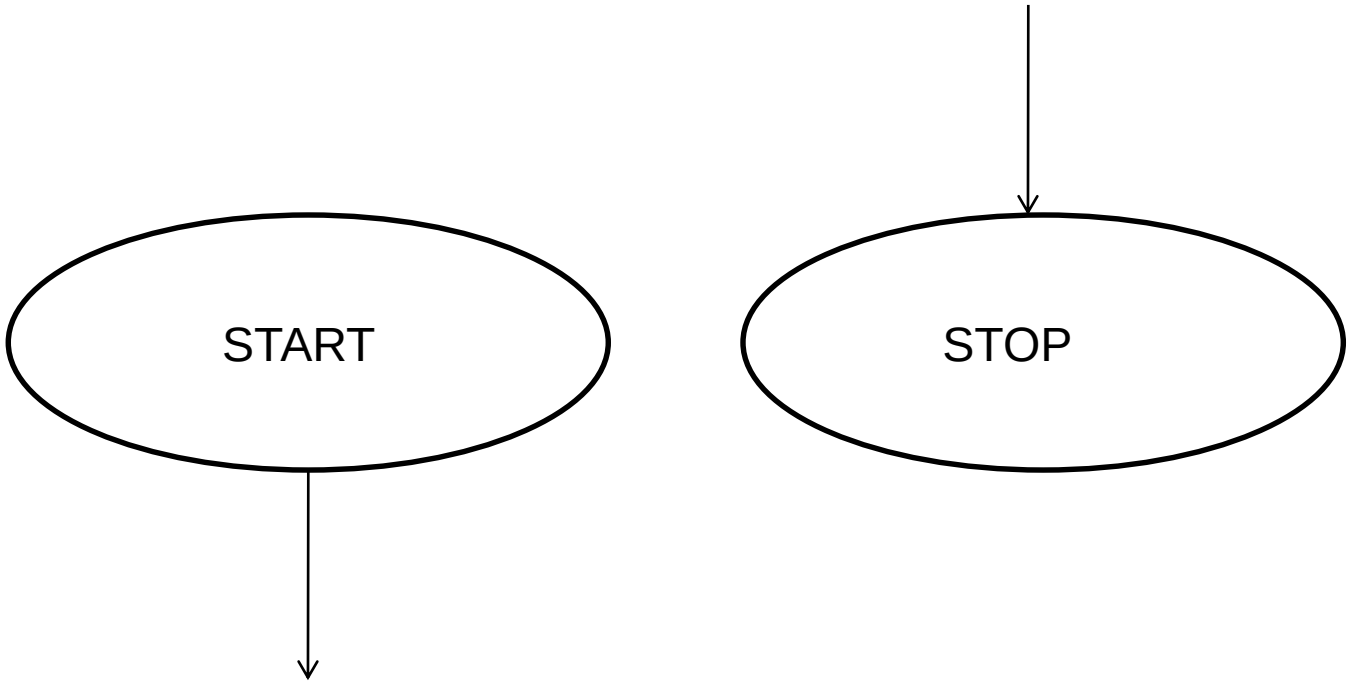
- **Notazione grafica usata per descrivere in modo intuitivo le azioni di cui è fatto un algoritmo.**

- **Viene usata per descrivere i passi salienti di un algoritmo, senza doversi preoccupare dei dettagli sintattici del programma corrispondente**

- **Una volta che l'algoritmo è stato descritto con un diagramma di flusso, deve però essere trasformato nel programma corrispondente.**

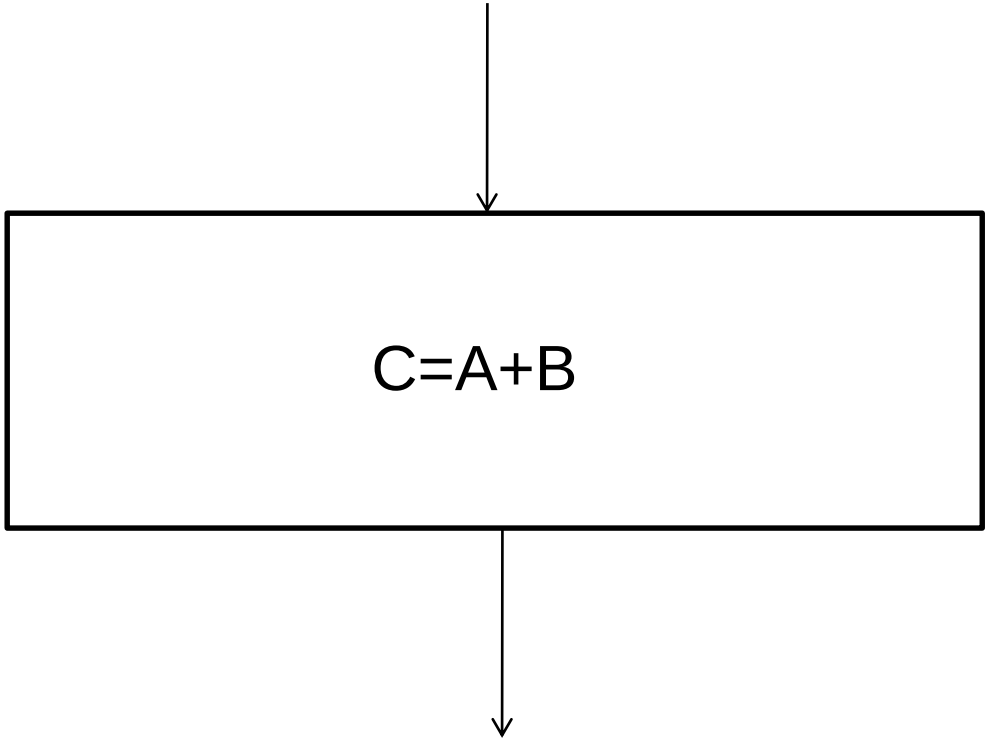
- **Ogni azione è rappresentata da un blocco**

Blocchi di flusso: inizio e fine algoritmo



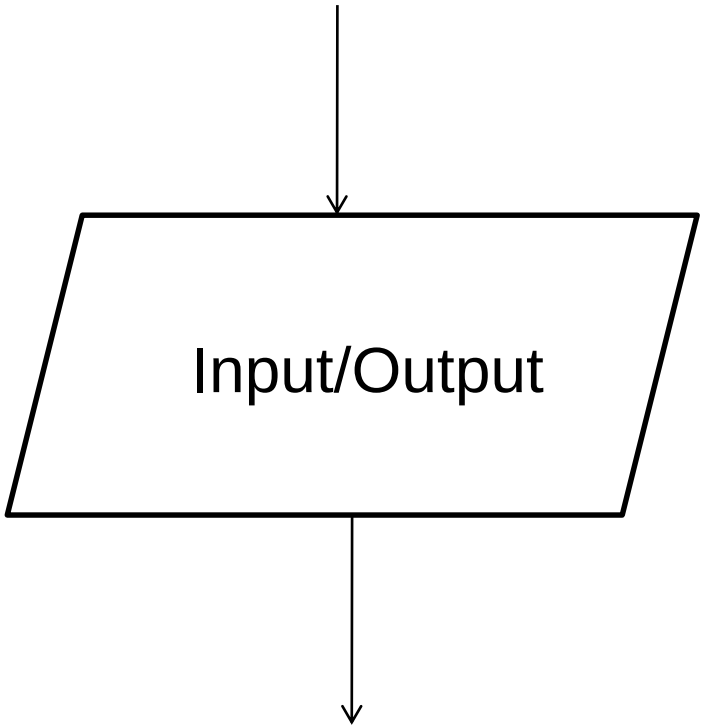
Il blocco START indica l'inizio del diagramma. Deve uscire una sola freccia
Il blocco STOP indica la fine del diagramma. Deve entrare una sola freccia

Blocchi di flusso: operazioni



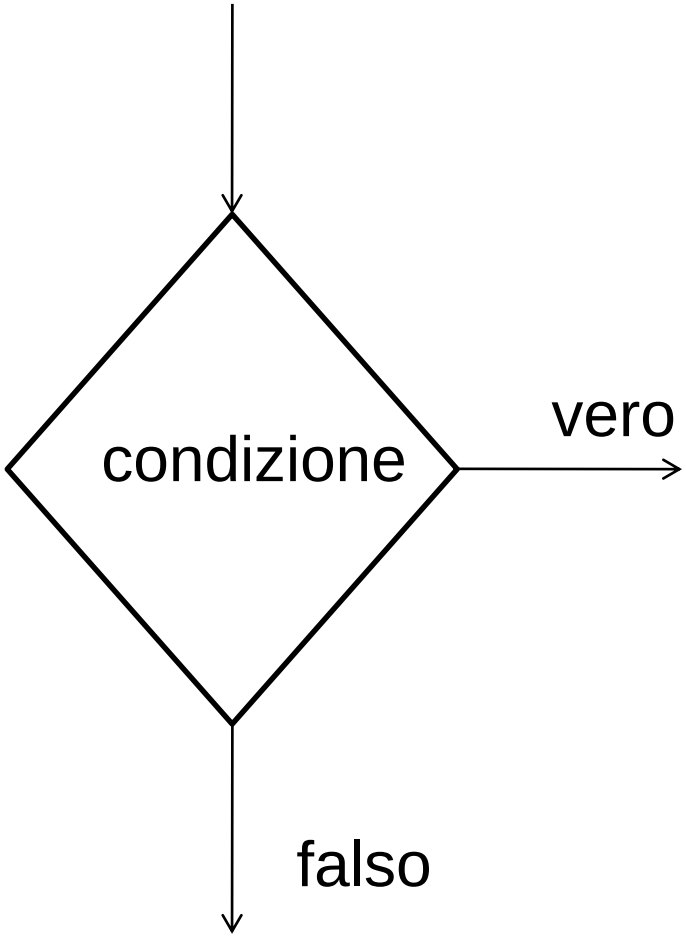
Indica operazioni da eseguire. Deve entrare ed uscire una sola freccia

Blocchi di flusso: input/Output



Indica lo scambio di dati con l'esterno. Gli input sono i dati che il diagramma riceve in ingresso mentre gli output sono i dati che il diagramma trasmette verso un soggetto terzo.
Deve entrare ed uscire una sola freccia

Blocchi di flusso:Blocco condizionale



Implica una scelta tra due possibili percorsi a seconda della valutazione di una certa condizione.
Deve entrare una freccia e devono uscirne due.

Il concetto di variabile

- **Per eseguire una qualsiasi computazione, abbiamo bisogno di poter immagazzinare i risultati temporanei e finali della computazione stessa.**

- **Ogni linguaggio ad alto livello mette a disposizione le variabili: “contenitori” in cui immagazzinare i dati della computazione**

• **Ogni variabile ha un nome mnemonico, che si usa nel programma per riferirsi alla variabile stessa.**

•**Una variabile contiene un valore che può essere modificato a piacimento**

• **Durante l'esecuzione di un programma, il sistema operativo mantiene una associazione tra il nome di ogni variabile e l'indirizzo della cella di memoria in cui è memorizzato il suo valore**

•**Quando si scrive un programma è necessario dichiarare quali variabili vogliamo usare.**

•**Le variabili possono essere di tipo diverso, per indicare che le usiamo per memorizzare dati di tipo diverso:**

- **Variabile LETTERA, tipo: carattere;**
- **Variabile SOMMA, tipo: intero;**

L’importanza delle variabili

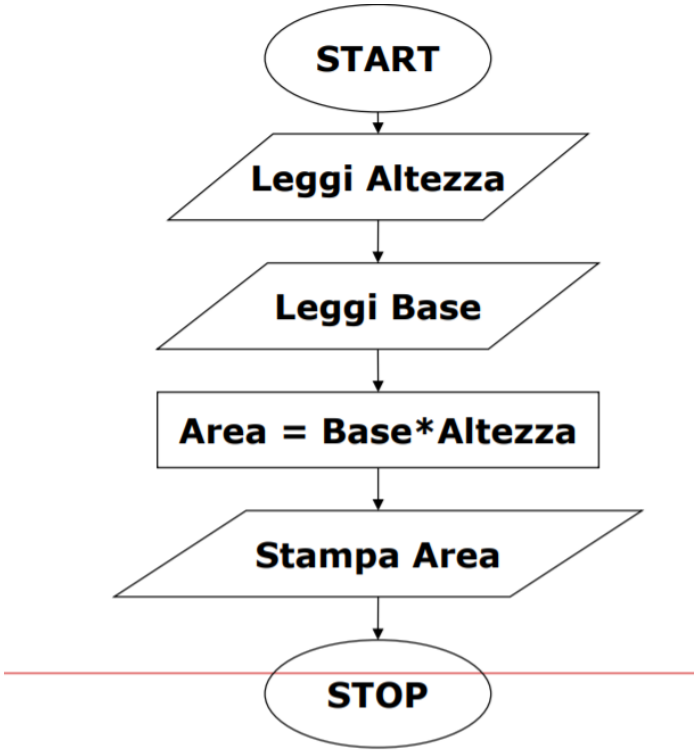
- **Le variabili sono lo strumento fondamentale per assicurare la flessibilità dei programmi.**

- **Lo stesso programma, eseguito con variabili di valore diverso da risultati diversi. Lo stesso programma si adatta cioè alle esigenze del momento, senza dover essere riscritto**

Esempi di algoritmi

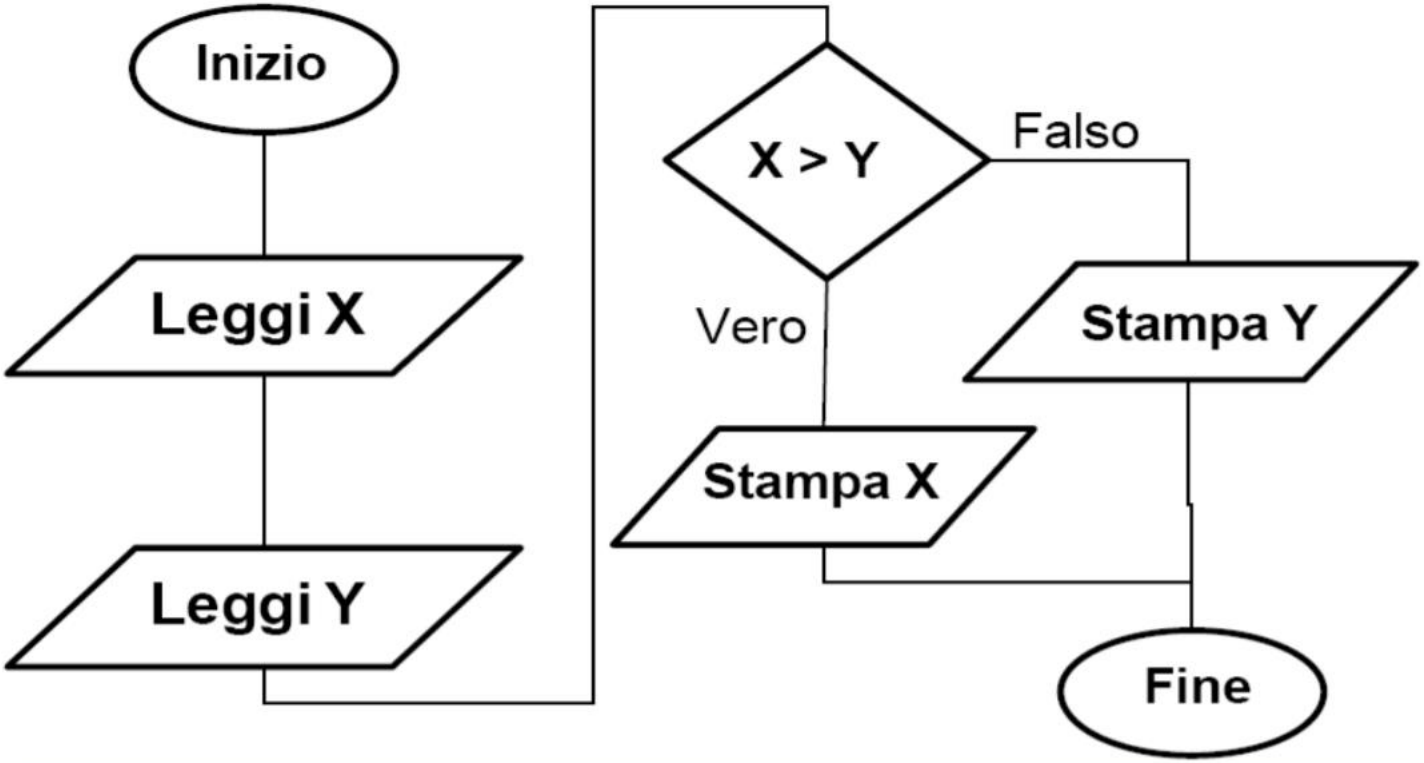
Calcolo dell'area di un rettangolo

- Leggi da input l'altezza (H)
- Leggi da input la base (B)
- Calcola l'area
- Dai in output il risultato



Massimo tra due numeri

Leggi X
Leggi Y
Se X > Y Stampa X
Altrimenti Stampa Y



Linguaggi di Programmazione

Saper programmare - ossia saper creare un codice in grado di far svolgere determinati compiti ad un computer - non è più prerogativa solamente di chi svolge una professione nel campo dell'ingegneria informatica, del web development o della creazione di app.
Competenze di programmazione ormai sono sempre più richieste nel mondo del lavoro.

Generazioni

prima generazione: rientrano in questa categoria i vari linguaggi macchina proprietari, decisamente complessi e fortemente legati all'architettura hardware specifica;
seconda generazione: costituiscono una prima astrazione del linguaggio macchina (assembly), permettendo comunque di migliorare l'attività di sviluppo grazie all'utilizzo di un numero minore di istruzioni. Con essi nascono gli *assembler*, ovvero programmi in grado di tradurre il linguaggio assembly in linguaggio macchina;
terza generazione: rientrano in questa categoria i linguaggi che spostano il focus verso il linguaggio naturale (english-like), risultando decisamente più intuitivi dell'assembly e del linguaggio macchina. Questi linguaggi vengono definiti anche come **linguaggi di alto livello** e tra essi si annoverano *in C, C++, Basic, Java, C#,* ecc…;
quarta generazione: in questa categoria troviamo i linguaggi *dichiarativi*, il cui rappresentate più illustre è SQL.

I LINGUAGGI DICHIARATIVI vengono utilizzati spesso per interrogare delle basi di dati. Hanno una sintassi di facile apprendimento.

PROBLEMA  ALGORITMO  PROGRAMMA

Programmazione. Insieme delle attività e tecniche svolte per creare un programma (codice sorgente) da far eseguire ad un computer.

Che lingua comprende un Computer?

Il linguaggio macchina o codice macchina è dato dall'insieme di "parole" che si possono creare con l'alfabeto binario: comprende due soli simboli, generalmente indicati con 0 e 1 (bit).

Il **linguaggio macchina** o **codice macchina** è il linguaggio in cui sono scritti i programmi eseguibili per computer. Può venire classificato come linguaggio di programmazione, sebbene quest'ultima espressione sia più spesso riservata per indicare i linguaggi di alto livello

Il linguaggio macchina è basato su un alfabeto detto *binario* perché comprende due soli simboli, generalmente indicati con 0 e 1. Un simbolo di questo alfabeto viene detto bit. Il processore o CPU è quella componente hardware di un computer che è in grado di eseguire i programmi scritti in linguaggio macchina.

Un **linguaggio di programmazione ad alto livello**, in informatica, è un linguaggio di programmazione caratterizzato da una significativa astrazione dai dettagli del funzionamento di un calcolatore e dalle caratteristiche del linguaggio macchina.

L'idea di un linguaggio automaticamente "traducibile" in linguaggio macchina, ma più vicino alla logica umana fu introdotta in informatica negli a partire dal 1950, soprattutto grazie al lavoro di John Backus (IBM), a cui si deve il primo linguaggio ad alto livello ad avere avuto ampia diffusione, il Fortran. Per questa innovazione Backus ha ricevuto il premio Turing.

Sono progettati per essere facilmente comprensibili dagli esseri umani, fino a includere alcuni elementi del linguaggio naturale.

Sono programmi non direttamente eseguibili, ma che richiedono una traduzione in linguaggio macchina, per esempio per mezzo di un **compilatore.**

Alcuni concetti sono presenti nella gran parte dei linguaggi:

Variabile e costante: un dato o un insieme di dati, noti o ignoti, già memorizzati o da memorizzare; ad una variabile corrisponde sempre, da qualche parte, un certo numero (fisso o variabile) di locazioni di memoria che vengono *allocate*, cioè riservate, per contenere i dati stessi. Molti linguaggi inoltre attribuiscono alle variabili un tipo, con differenti proprietà (stringhe di testo, numeri etc)
Espressione: una combinazione di variabili e costanti, unite da operatori; le espressioni sono state introdotte inizialmente per rappresentare le espressioni matematiche, ma in seguito la loro funzionalità si è estesa.
Strutture dati, meccanismi che permettono di organizzare e gestire dati complessi.
Strutture di controllo, che permettono di governare il flusso di esecuzione del programma, alterandolo in base al risultato o valutazione di una espressione (cicli iterativi quali ad esempio *for*, *do*, *while* e strutture condizionali quali ad esempio *if*, *switch-case*).
Sottoprogramma: un blocco di codice che può essere richiamato da qualsiasi altro punto del programma. In tale ambito quasi tutti linguaggi offrono funzionalità di riuso di codice accorpando cioè sequenze di istruzioni all'interno di funzioni richiamabili secondo necessità all'interno di programmi o all'interno di librerie richiamabili in ogni programma.
Funzionalità di input dati da tastiera e visualizzazione dati in output (stampa a video) attraverso i cosiddetti canali standard (standard input, standard output).
Possibilità di inserire dei commenti sul codice scritto, sintatticamente identificati e delimitati, che ne esplichino le funzionalità a beneficio della leggibilità o intelligibilità.

La compilazione è il processo per cui il programma, scritto in un linguaggio di programmazione ad alto livello, viene tradotto in un codice eseguibile per mezzo di un altro programma detto appunto compilatore.

Valutazione di un linguaggio

Caratteristiche intrinseche

Sono le qualità del linguaggio in sé, determinate dalla sua sintassi e dalla sua architettura interna. Influenzano direttamente il lavoro del programmatore, condizionandolo

Espressività: la facilità e la semplicità con cui si può scrivere un dato algoritmo in un dato linguaggio;

Didattica: la semplicità del linguaggio e la rapidità con cui lo si può imparare. Il BASIC, per esempio, è un linguaggio facile da imparare: poche regole, una sintassi molto chiara e limiti ben definiti fra quello che è permesso e quello che non lo è. Il C non è un linguaggio didattico è molto efficiente ed espressivo ma richiede tempo per essere padroneggiato.

Leggibilità: la facilità con cui, leggendo un codice sorgente, si può capire cosa fa e come funziona. La leggibilità dipende non solo dal linguaggio ma anche dallo stile di programmazione di chi ha creato il programma

Robustezza: è la capacità del linguaggio di prevenire, nei limiti del possibile, gli errori di programmazione

Flessibilità: la possibilità di adattare il linguaggio, estendendolo con la definizione di nuovi comandi e nuovi operatori. I linguaggi classici come il BASIC, il Pascal e il Fortran non hanno questa capacità, che invece è presente nei linguaggi come il C++ e Java.

Generalità: la facilità con cui il linguaggio si presta a codificare algoritmi e soluzioni di problemi in campi diversi. Di solito un linguaggio molto generale, per esempio il C, risulta meno espressivo e meno potente in una certa classe di problemi di quanto non sia un linguaggio specializzato in quella particolare nicchia, che in genere è perciò una scelta migliore finché il problema da risolvere non esce da quei confini.

Efficienza: la velocità di esecuzione e l'uso oculato delle risorse del sistema su cui il programma finito gira. In genere i programmi scritti in linguaggi molto astratti tendono ad essere lenti e voraci di risorse, perché lavorano entro un modello che non riflette la reale struttura dell'hardware ma è una cornice concettuale, che deve essere ricreata artificialmente; in compenso facilitano molto la vita del programmatore poiché lo sollevano dalla gestione di numerosi dettagli, accelerando lo sviluppo di nuovi programmi ed eliminando intere classi di errori di programmazione possibili. Viceversa un linguaggio meno astratto ma più vicino alla reale struttura di un computer genererà programmi molto piccoli e veloci ma a costo di uno sviluppo più lungo e difficoltoso.

Coerenza: l'applicazione dei principi base di un linguaggio in modo uniforme in tutte le sue parti. Un linguaggio coerente è un linguaggio facile da prevedere e da imparare, perché una volta appresi i principi base questi sono validi sempre e senza (o con poche) eccezioni.

Caratteristiche esterne

Oltre alle accennate qualità dei linguaggi, possono essere esaminate quelle degli ambienti in cui operano.

Diffusione: il numero di programmatori nel mondo che usa il tale linguaggio. Ovviamente più è numerosa la comunità dei programmatori tanto più è facile trovare materiale, aiuto, librerie di funzioni, documentazione, consigli. Inoltre ci sono un maggior numero di software house che producono strumenti di sviluppo per quel linguaggio, e di qualità migliore.

Standardizzazione: un produttore di strumenti di sviluppo sente sempre la tentazione di introdurre delle variazioni sintattiche o delle migliorie più o meno grandi ad un linguaggio, originando un *dialetto* del linguaggio in questione e fidelizzando così i programmatori al suo prodotto: ma più dialetti esistono, più la comunità di programmatori si frammenta in sottocomunità più piccole e quindi meno utili. Per questo è importante l'esistenza di uno standard per un dato linguaggio che ne garantisca certe caratteristiche, in modo da evitarne la dispersione. Quando si parla di *Fortran 77*, *Fortran 90*, *C 99* ecc. si intende lo standard sintattico e semantico del tale linguaggio approvato nel tale anno, in genere dall'ANSI o dall'ISO.

Integrabilità: dovendo scrivere programmi di una certa dimensione, è molto facile trovarsi a dover integrare parti di codice precedente scritte in altri linguaggi: se un dato linguaggio di programmazione consente di farlo facilmente, magari attraverso delle procedure standard, questo è decisamente un punto a suo favore.

Portabilità: la possibilità che portando il codice scritto su una certa **piattaforma** su un'altra, questo funzioni subito, senza doverlo modificare.

I programmi devono essere ***corretti*** ed ***efficienti.***

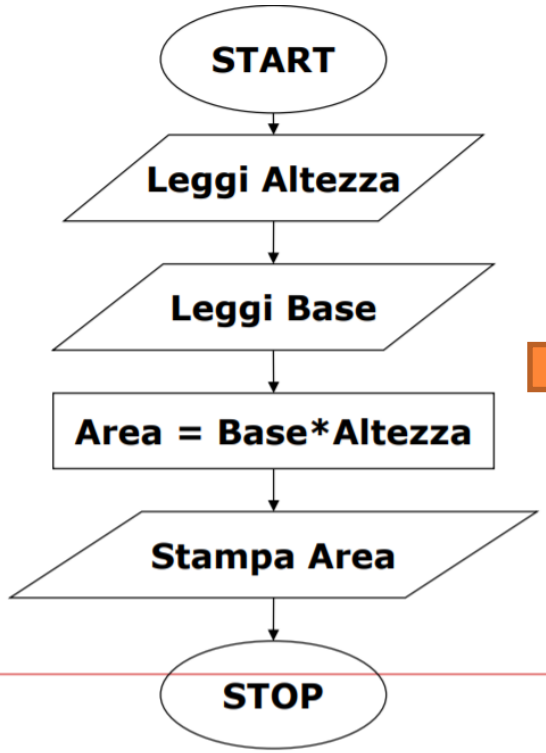
Corretto: validazione del risultato.
Efficiente: tempo, memoria (in termini di costo)

Linguaggio C

Viene definito come un linguaggio di programmazione ad alto livello e integra caratteristiche dei linguaggi di basso livello, Il C è stato concepito per essere snello e performante, si avvale peraltro di numerose librerie per far fronte ad ogni tipo di esigenza

Il linguaggio fu originariamente sviluppato da Dennis Ritchie[5] presso i Bell Labs della AT&T tra il 1969 e il 1973

AREA RETTANGOLO



Calcolo dell'area di un rettangolo

- Leggi da input l'altezza
- Leggi da input la base
- Calcola l'area
- Dai in output il risultato

```c
/* Calcolo area rettangolo */
#include <stdio.h>
int base, altezza, area;
main()
{
 printf("AREA RETTANGOLO\n\n");

 printf("Valore base: ");
 scanf("%d", &base);
 printf("Valore altezza: ");
 scanf("%d", &altezza);
 area = base*altezza;
 printf("Base: %d\n", base);
 printf("Altezza: %d\n", altezza);
 printf("Area: %d\n", area);
}
```

Compilazione

```
C:\Borland\BCC55\Bin>bcc32 calcoloarea.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
calcoloarea.c:
Warning W8070 calcoloarea.c 16: Function should return a value in function main
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
```

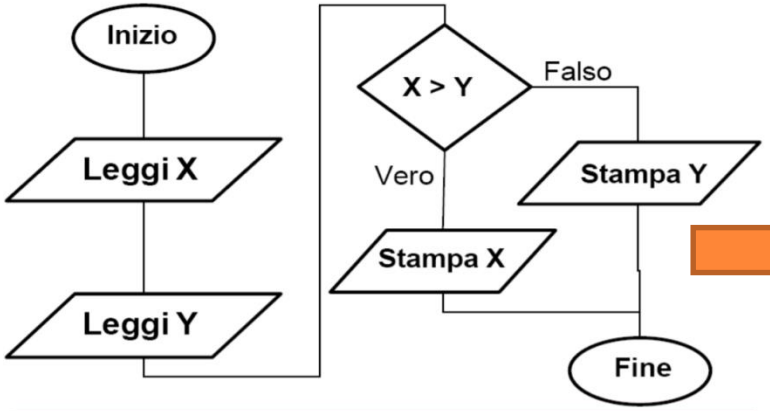
```
C:\Borland\BCC55\Bin>calcoloarea
AREA RETTANGOLO

Valore base: 2
Valore altezza: 3
Base: 2
Altezza: 3
Area: 6

C:\Borland\BCC55\Bin>
```

Massimo tra due numeri

Leggi x
Leggi y
Se x > y Stampa x
Altrimenti Stampa y




```c
/* Utilizzo if-else */
#include <stdio.h>
main()
{
int x;
int y;
printf("x: ");
scanf("%d", &x);
printf("y: ");
scanf("%d", &y);
if(x>y)
 printf("Il valore maggiore e' x: %d\n",x);
else
 printf("Il valore maggiore e' y: %d\n",y);
}
```

Compilazione

```
C:\Borland\BCC55\Bin>bcc32 utilizzoif.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
utilizzoif.c:
Warning W8070 utilizzoif.c 15: Function should return a value in function main
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
```

```
C:\Borland\BCC55\Bin> utilizzoif
x: 23
y: 20
Il valore maggiore e' x: 23
```

Altri linguaggi di programmazione

Creato nel 1991 inizialmente come linguaggio di programmazione per smart TV. Oggi è il linguaggio di programmazione più popolare al mondo, la sua posizione di prestigio in classifica è accreditata dal fatto che Java è cruciale per lo sviluppo delle applicazioni Android e per molti software di business.

python

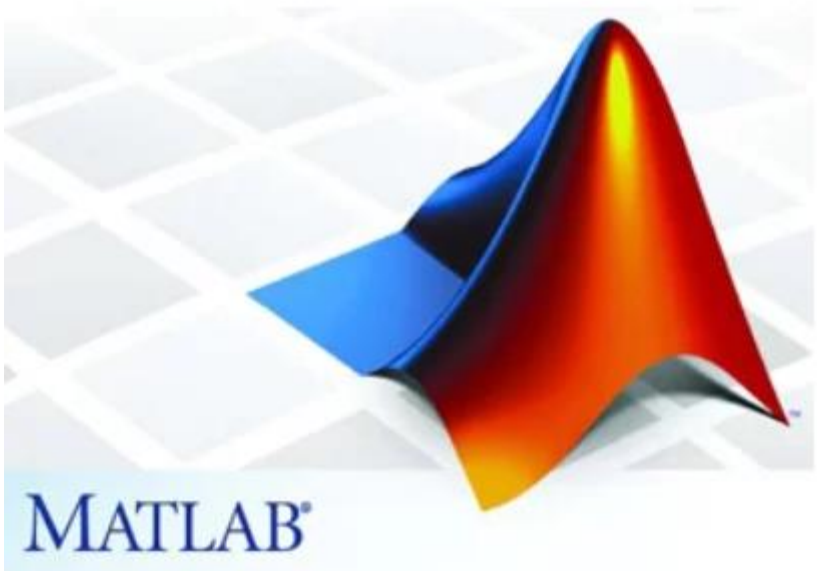
Creato nel 1989, è amatissimo dai suoi utilizzatori grazie al suo codice facilmente leggibile. Molti programmatori ritengono che sia il linguaggio più semplice per approcciare al mondo del linguaggio informatico.

Assembly Language

```
..B1.3:         # 100           # Preds ..B1.2
        cmpl    $4, %eax                                #11.19
        je      ..B1.8          # Prob 25%              #11.19
                                # LOE rbp r12 r13 r14 r15 eax ebx
..B1.4:         # 75            # Preds ..B1.3
        movl    $2, %edx                                #17.9
        xorl    %esi, %esi                              #17.9
        cmpl    $8, %eax                                #17.9
        cmove   %edx, %esi                              #17.9
                                # LOE rbp r12 r13 r14 r15 ebx esi
..B1.5:         # 100           # Preds ..B1.4 ..B1.8
        movl    $_2__STRING.0.0, %edi                   #19.5
        xorl    %eax, %eax                              #19.5
        call    printf
..B1.8:         # 25            # Preds ..B1.3    # Infreq
        movl    $5, %esi                                #13.9
        jmp     ..B1.5          # Prob 100%             #13.9
```

Creato nel 1949, il linguaggio Assembly è tra i linguaggi di programmazione, quello più vicino al linguaggio macchina vero e proprio. E' utile per la costruzione di software su macchine a bassa potenza come smart application e computer indossabili (ad esempio: smart watch), infatti negli ultimi tempi è ritornato di nuovo in auge.

E' un software statistico, definito come un ambiente, ovvero un insieme di macro, librerie, oggetti che possono essere utilizzati per la gestione, l'analisi dei dati e la produzione di grafici. R è molto utilizzato nei centri di ricerca Google.



Creato come un linguaggio di programmazione matematica per contribuire ad insegnare agli studenti universitari di algebra avanzata e di elaborazione delle immagini, è oggi ampiamente utilizzato da scienziati, ingegneri e programmatori che lavorano nel campo dell'elaborazione delle immagini e di altre applicazioni di intelligenza artificiale. MATLAB è usato da milioni di persone nell'industria e nelle università per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio applicati e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, GNU/Linux e Unix.

AREA RETTANGOLO C

```c
/* Utilizzo if-else */
#include <stdio.h>
main()
{
int x;
int y;
printf("x: ");
scanf("%d", &x);
printf("y: ");
scanf("%d", &y);
if(x>y)
 printf("Il valore maggiore e' x: %d\n",x);
else
 printf("Il valore maggiore e' y: %d\n",y);
}
```

```
C:\Borland\BCC55\Bin>bcc32 calcoloarea.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
calcoloarea.c:
Warning W8070 calcoloarea.c 16: Function should return a value in function main
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
```

```
C:\Borland\BCC55\Bin>calcoloarea
AREA RETTANGOLO

Valore base: 2
Valore altezza: 3
Base: 2
Altezza: 3
Area: 6

C:\Borland\BCC55\Bin>
```

Linguaggi come il C, C++, Delphi, Visual Basic sono **linguaggi compilati** e seguono questi passi: si scrive il codice in un editor, al più utilizzando un ambiente di sviluppo IDE che ne facilita la creazione, questo codice viene poi controllato per verificare che non ci siano errori e poi viene compilato, ovvero ogni istruzione viene trasformata nel corrispondente codice in linguaggio macchina che può essere, così, eseguito dal processore;

AREA RETTANGOLO Matlab

```
1   %calcolo area rettangolo
2
3   disp('AREA RETTANGOLO')
4   base = input('Valore base ')
5   altezza= input('Valore altezza ')
6   area=base*altezza
7   sprintf ('Base: %d',base)
8   sprintf ('Altezza: %d',altezza)
9   sprintf ('Area: %d',area)
10
```

altezza	3
ans	'Area: 6'
area	6
base	2

```
Command Window

ans =

    'Area: 6'
```

I linguaggi interpretati, invece, seguono una strada diversa, il codice sorgente viene, appunto, interpretato al volo e vengono, quindi, eseguite le istruzioni così come descritte nel codice sorgente. La potenza di questo genere di linguaggi è, di fatto, l'alta portabilità e l'immediatezza tra quello che scriviamo e quello che viene presentato all'esecuzione del programma, ma rimangono dei problemi come la ricerca di errori nel codice sorgente o il carico di lavoro maggiore per il processore

Massimo tra due numeri C

```c
/* Utilizzo if-else */
#include <stdio.h>
main()
{
int x;
int y;
printf("x: ");
scanf("%d", &x);
printf("y: ");
scanf("%d", &y);
if(x>y)
 printf("Il valore maggiore e' x: %d\n",x);
else
 printf("Il valore maggiore e' y: %d\n",y);
}
```

```
C:\Borland\BCC55\Bin>bcc32 utilizzoif.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
utilizzoif.c:
Warning W8070 utilizzoif.c 15: Function should return a value in function main
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
```

```
C:\Borland\BCC55\Bin> utilizzoif
x: 23
y: 20
Il valore maggiore e' x: 23
```

Massimo tra due numeri Matlab

```
1   %massimo tra due numeri
2
3   disp('MASSIMO TRA DUE NUMERI')
4   x = input('X: ')
5   y = input('Y: ')
6   if x>y
7   sprintf ('Il valore maggiore è x: %d',x)
8   else
9   sprintf ('Il valore maggiore è y: %d',y)
10  end
11
```

altezza	5
ans	'Il valore maggio...
area	30
base	6
x	23
y	20

Command Window

```
ans =

    'Il valore maggiore è x: 23'
```

Una ricerca effettuata su un campione di 26 milioni di offerte di lavoro da Burning Glass e Oracle Academy, ha rivelato che metà dei posti di lavoro al top per retribuzione (dai 57mila dollari in su l'anno) richiede almeno le nozioni di base dei linguaggi di programmazione.

I lavori in questione non erano legati strettamente alla tecnologia digitale. L'informatica è oggi presente in ogni tipo di industria, e questo significa che saper programmare è un tipo di competenza richiesto in molteplici campi. Chimica, fisica, biologia, marketing, ingegneria, comunicazione, banking, assicurazioni