Che cos'è un programma?
Un programma, in informatica, è un software che può essere eseguito da un elaboratore che riceve in input determinati dati di un problema che ammette una soluzione algoritmica e restituisce in output gli (eventuali) risultati ottenuti a seguito dell'esecuzione delle istruzioni.
Un programma è una lista di istruzioni che insieme dicono al computer che cosa fare.
Deve essere scritto in un modo che il computer possa comprenderlo. Soltanto installarlo non è abbastanza: deve essere eseguito in modo da produrre dei risultati.
Un programma scritto in linguaggio assembly o in un linguaggio di programmazione ad alto livello (codice sorgente), può essere eseguito solo traducendolo in linguaggio macchina che darà vita poi al programma eseguibile, oppure servendosi di un interprete. Un programma scritto direttamente o convertito in linguaggio macchina può essere eseguito direttamente da un computer (inteso come hardware).
L'esecuzione di codice da parte di una macchina hardware nel ciclo di fetch-execute è possibile in virtù della capacità del processore di eseguire una serie di istruzioni base (instruction set), sulla quale il programma è mappato/tradotto a livello di linguaggio macchina, grazie ai circuiti elettronici di base che compongono il processore stesso.
Hardware→ è la parte fisica di un computer, ovvero tutte quelle parti elettroniche, elettriche, meccaniche, magnetiche, ottiche che ne consentono il funzionamento.
CPU→ l'unità di elaborazione centrale (o processore centrale) è un tipo di microprocessore digitale il cui compito è quello di eseguire le istruzioni dei programmi che vengono prelevati dalla memoria di massa, un hard disk per esempio, e aperti nella memoria ad accesso casuale, la RAM. Un valore importante è la frequenza di clock (indica la velocità con la quale la CPU esegue un ciclo di operazioni): maggiore è la frequenza di clock, maggiore sarà la velocità del processore.

Instruction Set of CPU
es. CPU ad 8 bit degli anni '80
-LOAD: prendi un numero dalla RAM alla CPU
-ADD: aggiungi due numeri insieme
-STORE: un numero dalla CPU torna alla RAM
-COMPARE: un numero con un altro
-JUMP ad un altro indirizzo nella RAM
-OUTPUT
-INPUT
Dato che i numeri sono limitati, lo sarà anche il numero di istruzioni di base.

```
$max=1000;
for ($i=1; $i<=$max; $i++) {
   $square= $i*$i;
   echo "$i squared is $square \r";
}
```

es. PHP (simil C)

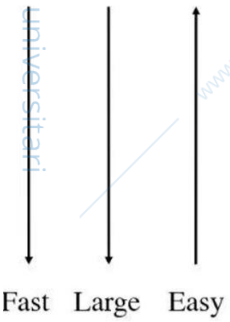
Che cos'è un linguaggio di programmazione?
Un linguaggio di programmazione, in informatica, è un linguaggio formale che specifica un insieme di istruzioni che possono essere usate per produrre dati in output. Esso è utilizzabile per il controllo del comportamento di una macchina o di una implementazione di essa, ovvero in fase di programmazione di questa attraverso la scrittura del codice sorgente di un programma ad opera di un programmatore.
Per linguaggio di programmazione si intende una serie di regole che definiscono come scrivere un codice.

Tutti i linguaggi di programmazione esistenti sono definiti da un lessico, una sintassi ed una semantica e includono:
-variabili (numeri, stringhe) o costanti: 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 di locazioni di memoria che vengono allocate, cioè riservate per contenere i dati stessi.
-espressioni (addizioni, sottrazioni): una combinazione di variabili e costanti, unite da operatori; sono state introdotte inizialmente per rappresentare le espressioni matematiche, ma in seguito la loro funzionalità si è estesa.
-Istruzioni (write, read): un comando oppure una regola descrittiva, ogni volta che un'istruzione viene eseguita, lo stato interno del calcolatore cambia.
-flow control (if, for): strutture di controllo che permettono di governare il flusso di esecuzione del programma, alterandolo in base al risultato o valutazione di un'espressione.

Interprete→ Il linguaggio di programmazione è un file di testo, deve essere letto da un interprete di un linguaggio (nel caso del linguaggio C lo traduce tutto in una sola volta). Se interpretato, può essere eseguito su qualsiasi macchina. Un interprete, in informatica e nella programmazione, è un programma in grado di eseguire altri programmi a partire direttamente dal relativo codice sorgente. Ha lo scopo di eseguire un programma in un linguaggio di alto livello, senza la previa compilazione dello stesso (codice oggetto) cioè di eseguire le istruzioni nel linguaggio usato, traducendole di volta in volta in istruzioni in linguaggio macchina.

Compilatore→ produce file .exe, direttamente utilizzabili. A differenza di un interprete, un compilatore non esegue il programma che riceve in ingresso, ma lo traduce in linguaggio macchina (memorizzando su file il codice oggetto pronto per l'esecuzione diretta da parte del processore).Per qualunque linguaggio di programmazione si può scrivere sia un interprete che un compilatore, pertanto le espressioni linguaggio interpretato e linguaggio compilato, per quanto comuni, sono improprie, essendo interpretazione e compilazione concetti afferenti alla implementazione di un linguaggio.

I linguaggi di programmazione non sono tutti uguali ma, sono classificabili in high e low level languages. Più ci si avvicina ai linguaggi di basso livello, maggiore sarà la velocità di esecuzione del programma.



- shell scripts, applescript, batch files
- PHP, PERL
- Basic, Java
- C, C++, (Pascal)
- assembler

I linguaggi ad alto livello, hanno istruzioni molto complesse e quindi il file sarà anche più grande.
-shell scripts, applescript, batch files (molto semplici)
-PHP, PERL
-Basic, Java
-C, C++, Pascal
-assembler (la copia in testo dei numeri che l'hardware usa, molto complesso)

Che cos'è un algoritmo?
Un algoritmo è un procedimento che risolve un determinato problema attraverso un numero finito di passi elementari, chiari e non ambigui, in un tempo ragionevole. L'algoritmo è un concetto fondamentale dell'informatica, anzitutto perché è alla base della nozione teorica di calcolabilità: un problema è calcolabile quando è risolvibile mediante un algoritmo.

Inoltre, l'algoritmo è un concetto cardine anche della fase di programmazione dello sviluppo di un software: preso un problema da automatizzare, la programmazione costituisce essenzialmente la traduzione o codifica di un algoritmo per tale problema in programma, scritto in un certo linguaggio, che può essere quindi effettivamente eseguito da un calcolatore rappresentandone la logica di elaborazione.Un insieme di istruzioni che contengono la procedura per ottenere un determinato risultato ma, che a differenza di un programma, è leggibile dall'utente (essere umano): non si riferisce a uno specifico linguaggio di programmazione.
Algorithms are for humans, programs for computers

esempio
-assegnare un valore a max
-per ogni intero da 1 a max
-calcolare il quadrato (moltiplica il numero per sé stesso)
-stampa il numero e il suo quadrato

ATGCTGACTGTGAATTCTGACTGGTCAATGC

AATTCT	**13**
ATGCTG	**1**
...	
GAATTC	**12**
GCTGAC	**3**
...	
TGCTGA	**2**

Better algorithms make faster programs
(es. ricerca dei siti di restrizione: m*n diventa m+1/4m+1/16m+1/64m...
Attraverso un differente approccio, creo un indice della sequenza nella quale ricercare GATTCC, avendo ora 4^6 combinazioni possibili che creano una tabella di 4096 righe, nelle quali si troverà direttamente la posizione della sequenza ricercata, ottimizzando i tempi di ricerca, dato che si effettua un solo spazzolamento della seq.)

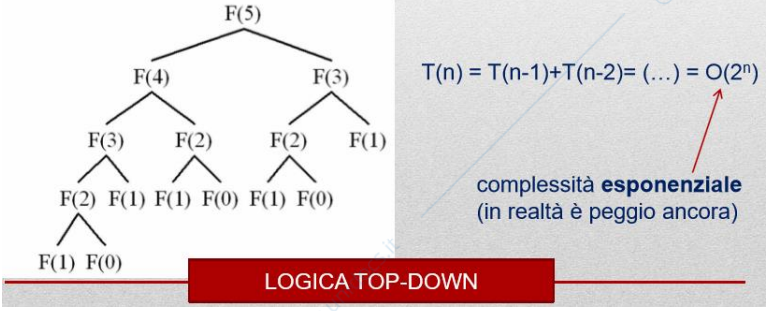
Programmazione dinamica

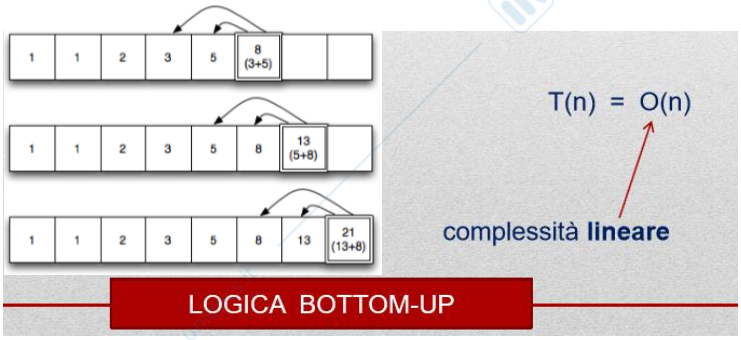
La successione di Fibonacci

La successione di Fibonacci (detta anche successione aurea), indicata con F_n o con Fib(n), in matematica indica una successione di numeri interi positivi in cui ciascun numero a cominciare dal terzo è la somma dei due precedenti, dove i primi due sono, per definizione: $F_0=0$ e $F_1=1$. Questa successione è definita da:

$$F_0 = 0,$$
$$F_1 = 1,$$
$$F_n = F_{n-1} + F_{n-2} \text{ (per ogni n>1)}$$

```
algoritmo Fibonacci2(int n) → int
    if ( n==1 || n==2 ) then
        return 1;
    else
        return Fibonacci2(n-1)+Fibonacci2(n-2);
    endif
```





CS Cosa ci insegna l'algoritmo iterativo per il calcolo dell'ennesimo numero di Fibonacci? (progr. dinamica)

fib_table	
F[1]	1
F[2]	1
F[3]	2
F[4]	3
F[5]	5
F[6]	8
F[7]	13
F[8]	21
F[9]	34
F[10]	55
F[11]	89
F[12]	?

- **Cosa fa la soluzione iterativa?**
  - rivela sottoproblemi identici
  - ordina le operazioni in modo da favorirne il riutilizzo
  - riempie una tabella di risultati (intermedi)
  - esprime problemi complessi come insiemi di sottoproblemi più semplici
- L'ordine delle operazioni **conta**
  - approccio top-down molto lento (risultati intermedi non disponibili)
  - approccio bottom-up efficace (risolve sottoproblemi, salva risultati, non ricalcola la soluzione ma la costruisce da valori già disponibili)

>Il problema può essere rappresentato come insieme di sottoproblemi.
> In problemi di ottimizzazione reali:
- scelte ottimali a livello locale
- score sommato rispetto a tutto lo spazio dei sottoproblemi
- soluzione: traceback per trovare il "percorso migliore" tra le soluzioni dei sottoproblemi

Needleman e Wunsch è un algoritmo veloce perché tutti i percorsi possibili sono calcolati una sola volta.
-calcolare insieme molti path
-conservare il risultato intermedio
E' un esempio di programmazione dinamica.