Variabili
Gli oggetti su cui opera un programma sono le variabili. Una variabile ha un nome, un tipo e un valore; occupa spazio nella memoria di lavoro (**RAM**).
Nome (identificatore): a, d, x, somma, average, x1
Tipo: “numero intero”, “numero reale”, carattere, sequenza di numeri…
Di norma, lo spazio occupato dipende dal tipo; ogni tipo corrisponde a uno spazio fissato a priori e determina i valori che la variabile può avere e le operazioni che si possono eseguire

Cos’è una funzione?

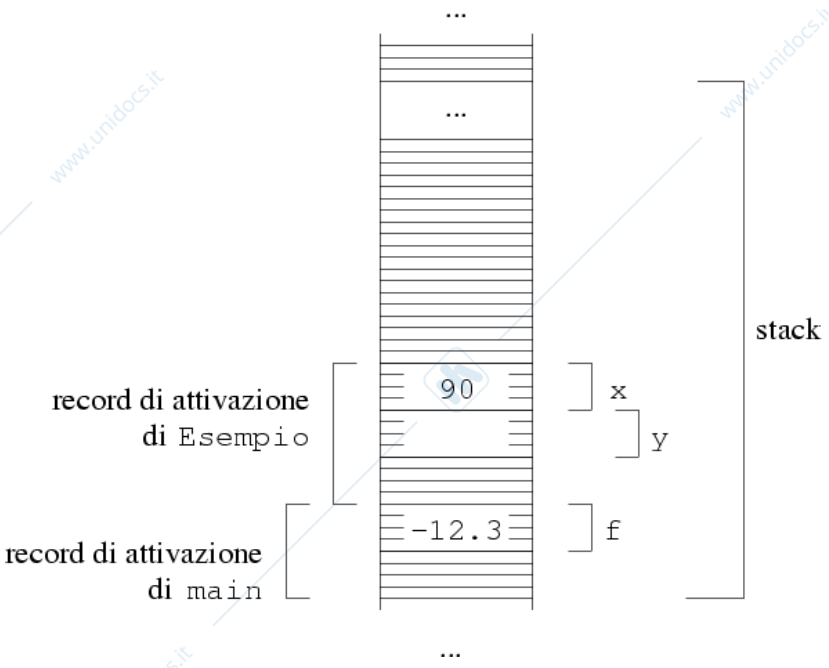
La funzione è una sezione di un programma che svolge un compito specifico, riceve degli argomenti in ingresso e produce un risultato (o un effetto).
Può essere chiamata più volte in un programma, con argomenti diversi, è utile per dare ordine al codice ed evitare di ripetere le stesse istruzioni tante volte.
Una funzione in un programma è simile a una funzione matematica: il tipo degli argomenti definisce il suo dominio; la funzione produce un valore di un tipo definito (analogo al codominio).
La formula che definisce la funzione è sostituita dalle istruzioni che formano il corpo della funzione.

Le funzioni nel linguaggio C

Un programma in C consiste in una serie di definizioni di funzioni (oltre ad altri elementi). Il nome main denota una funzione speciale, che deve essere presente in ogni programma e dalla quale prende avvio l’esecuzione.
Da una funzione è possibile chiamare (o invocare) altre funzioni, specificando i valori degli argomenti; al termine dell’esecuzione della funzione chiamata, la funzione chiamante riceve il valore prodotto dalla chiamata.
Possiamo scrivere funzioni che non producono nessun valore, ma producono effetti (ad esempio, stampare qualcosa sul terminale o modificare il contenuto di una cella di memoria).Le funzioni nel linguaggio C
La definizione di una **funzione in C** si compone di due parti: i**ntestazione e corpo**
Nell’intestazione si devono specificare
-il tipo del valore prodotto dalla funzione
-il nome della funzione
Il corpo della funzione contiene le istruzioni da eseguire quando la funzione viene chiamata. In particolare, deve contenere un’istruzione speciale, chiamata return, che segnala la fine dell’esecuzione e specifica il valore prodotto dalla funzione.
Gli identificatori degli argomenti nell’intestazione di una funzione sono chiamati **argomenti formali**.
I valori specificati nella chiamata di una funzione sono chiamati **argomenti reali**.
Una variabile dichiarata all’interno di una funzione è visibile solo all’interno della stessa funzione.

Che cosa succede quando una funzione viene chiamata?

Nella memoria di lavoro viene creato il **record di attivazione** della funzione, cioè un’area della memoria riservata a questa particolare chiamata. Il record di attivazione contiene lo spazio per gli argomenti e per le variabili dichiarate nella funzione.
I valori degli argomenti reali vengono copiati nelle celle associate agli argomenti formali. Si eseguono le istruzioni del corpo della



funzione fino a incontrare un'istruzione return. Il valore prodotto
dalla funzione viene "consegnato" alla funzione chiamante.
Infine, il record di attivazione viene cancellato.
Ogni volta che viene chiamata una funzione, si crea per essa un nuovo record di attivazione, che viene messo immediatamente sopra a quelli già presenti nello stack.
L'area di memoria in cui vengono allocati i record di attivazione viene gestita come una pila: **STACK**.
Lo stai è una struttura dati gestita a tempo di esecuzione con politica LIFO (Last In, First Out - l'ultimo a entrare è il primo a uscire) nella quale ogni elemento è un record di attivazione.
La gestione dello stack avviene mediante due operazioni:
push: aggiunta di un elemento (in cima alla pila)
pop: prelievo di un elemento (dalla cima della pila)
L'ordine di collocazione dei record di attivazione nello stack indica la cronologia delle chiamate.

Le funzioni main, scanf, printf fanno parte di una collezione che viene chiamata **libreria standard**; per usarle, dobbiamo inserire nel **codice sorgente** le direttive *include*.
Le funzioni scanf e printf non producono un risultato, ma un effetto: assegnare un valore a una variabile o stampare un valore sul terminale.

(**Codice sorgente**: testo di un algoritmo di un programma scritto in un linguaggio di programmazione da parte di un programmatore in fase di programmazione)

Cos'è un vettore?

Un vettore è una sequenza di elementi, tutti dello stesso tipo; ciascun elemento è associato alla sua posizione nel vettore. Nella programmazione, per ragioni legate al modo in cui CPU e RAM si comunicano le posizioni degli elementi di un vettore, le posizioni partono da 0.
Un vettore è comunemente chiamato array. Per dichiarare un vettore in C, si devono specificare il tipo degli elementi che lo compongono, l'identificatore del vettore, e il numero di elementi che lo compongono (che resterà fisso).

int val[18]; float temperature[365]; char sigla[3];

*(**RAM,** memoria di lavoro, è una sorta di magazzino che ospita i programmi in corso di esecuzione e i dati sui quali i programmi operano; la memoria di lavoro è organizzata in celle, ognuna delle quali ha un indirizzo, l'unità di misura della memoria è il byte, sufficiente a memorizzare un carattere o un numero intero piccolo. infine, la memoria di lavoro (RAM) è volatile: se non alimentata perde il suo contenuto.)*
*(**CPU,** unità di elaborazione, è il componente in grado di eseguire effettivamente le istruzioni. Contiene registri, nei quali parcheggia l'istruzione da eseguire e i dati su cui opera quella istruzione, e circuiti in grado di manipolare i dati.)*
Dati e istruzioni transitano tra RAM e CPU lungo il canale di comunicazione (bus).

Ogni elemento di un vettore è una variabile, che si può usare come una qualsiasi variabile dello stesso tipo. Il numero che indica la posizione in un vettore è spesso chiamato indice.

val[5] = x+3;

Possiamo considerare una matrice come un vettore di vettori. Questo punto di vista ci porta alla rappresentazione di matrici in C.
Per dichiarare una matrice, ad esempio, 3 × 5 di numeri reali, scriveremo:
double mat[3][5]

L'elemento alla riga 2 e colonna 3 di questa matrice è la variabile mat[1][2] (ricordando che le posizioni partono da 0).

Strutture record

Se dobbiamo gestire dati di tipo diverso, possiamo utilizzare le strutture in C. Queste strutture si chiamano record. Il costruttore di tipo corrispondente si chiama **struct**.
Un record, identificato da un nome, è un insieme di dati, non necessariamente omogenei, identificati dal tipo e dal nome, detti campi.

```
struct alunno
{
 char cognome[30];
 char nome[20];
 int classe;
 char sezione;
}
```

Il campo è un singolo componente di un record che a sua volta può essere strutturato.
Possiamo anche utilizzare un record come tipo elementare di un altro record, definendo strutture dati complesse che prendono nome di record di record.
Una variabile di un tipo struct può essere manipolata come
oggetto unitario e può anche essere passata come argomento di una funzione.
Non si possono confrontare due variabili di uno stesso tipo struct per stabilire se sono uguali o diversi con una espressione semplice. Non possiamo cioè scrivere: "*if (z == w) …"*
Se si vuole confrontare le due variabili, bisogna confrontare esplicitamente i singoli campi.

Allocazione di memoria e puntatori

Problema: la pila è un modello dinamico, non conosciamo a priori la sua dimensione; la dimensione varia durante l'esecuzione.
Non possiamo allocare staticamente la memoria necessaria per la pila (potremmo allocare uno spazio abbondante, ma richiamo di sprecare memoria o di esaurire lo spazio durante l'esecuzione)
Soluzione: **allocazione dinamica della memoria**.

Ogni variabile occupa una o più celle della memoria di lavoro, ogni cella ha un indirizzo e ad ogni variabile possiamo associare un indirizzo

Una variabile può contenere l'indirizzo di un'altra variabile, una variabile che contiene un indirizzo è un **puntatore.** I puntatori possono contenere indirizzi di variabili (o di valori) di un solo tipo.
Dichiarazioni di puntatori:
(L'indirizzo di una variabile: &v)

```
double * pd;
int * z;
struct punto * psp;
```

Un puntatore può essere passato come argomento a una funzione. La funzione riceve un indirizzo e può modificare il valore della cella di memoria corrispondente, anche se non appartiene al suo record di attivazione.

Per chiedere nuovo spazio nella memoria di lavoro durante l'esecuzione di un programma, si usa la funzione **malloc**. La funzione malloc chiede al gestore della memoria di allocare al programma uno spazio di memoria supplementare. L'argomento della funzione specifica il numero di celle di memoria richieste (byte).
La funzione restituisce un puntatore all'area di memoria allocata, cioè restituisce l'indirizzo della prima cella dell'area allocata.

*ptr = malloc(100);

La variabile **ptr* dev'essere stata dichiarata in precedenza, e dev'essere di tipo puntatore.
Caso pratico: abbiamo bisogno di spazio per un nuovo valore di tipo double:

```
double * pt;
...
ptd = malloc (sizeof(double));
```

La dimensione di un valore di tipo double non è specificata dal linguaggio; usiamo allora la funzione sizeof.

Una pila è una sequenza di elementi di lunghezza variabile
Vettore (array): sequenza di elementi di lunghezza fissa.

Liste concatenate
Ogni elemento della catena è un record; un campo del record contiene l'indirizzo dell'elemento successivo.
Il valore speciale NULL segnala la fine della catena.
Un puntatore esterno contiene l'indirizzo del primo elemento della catena (la testa).
La lista concatenata è una struttura dati in cui ogni nodo ha un collegamento (link) ad un altro nodo (linked list). Il collegamento è un puntatore con l'indirizzo di memoria dell'altro nodo

- Si accede alla lista per mezzo di un puntatore al suo primo nodo (head)
- Si accede ai nodi successivi per mezzo del puntatore memorizzato in ogni nodo
- Il puntatore dell'ultimo nodo è impostato a NULL

In una lista i dati sono memorizzati in modo dinamico.

Funzioni ricorsive
Problema: riprodurre la stessa configurazione sul piolo di destra
Mosse lecite: spostare un disco da un piolo a un altro (solo il disco più in alto); non si può spostare un disco sopra un disco più piccolo (le torri di Hanoi).
Osservazione preliminare: il problema è banale se c'è un solo disco. Il problema è più semplice quanto minore è il numero dei dischi. Invento un algoritmo risolutivo..

Esempio:
H è una funzione, con una caratteristica speciale: al suo interno, H chiama sé stessa; una funzione di questo tipo si dice ricorsiva, e diremo ricorsiva anche la auto-chiamata.
Esecuzione
Poiché ogni chiamata "incorporata" riduce di uno il valore del primo argomento, prima o poi si arriverà a chiamare H con il primo argomento uguale a 1; questa chiamata interrompe la serie e si risolve con l'esecuzione di una mossa, in modo che il

programma risale nella catena di chiamate ricorsive, fino a terminare l'esecuzione.

CICLO WHILE

Nel linguaggio C l'istruzione *While* crea una struttura iterativa condizionata che esegue lo stesso blocco di codice fin quando si verifica un particolare evento.
Se l'espressione di controllo è vera il blocco di istruzioni compreso tra le parentesi graffe {} viene eseguito una o più volte. Ogni esecuzione del blocco di istruzioni è detto ciclo o iterazione. In ogni ciclo sono eseguite le stesse istruzioni del blocco.
Il ciclo si interrompe quando l'espressione di controllo diventa falsa.
Generalmente l'istruzione o le istruzioni all'interno del while agiscono sulla condizione che il while aspetta essere falsa per poter uscire dal ciclo, questo perché altrimenti il ciclo non terminerebbe.

Ad esempio per stampare una successione di cifre da 0 a 99, proponiamo il codice seguente:

```
int i = 0;
while (i != 100)
{
printf("%d n", i);
i++;
}
```

Affinché il while possa verificare la condizione associata, è necessario aver dichiarato la variabile prima del ciclo, questo, come nell'esempio, può essere fatto nella riga soprastante o in un altra parte del programma.

CICLO FOR

equivale ad un ciclo while con la seguente struttura:

```
inizializzazione
while (condizione)
istruzione/i
incremento
```

Il for, quindi, potrebbe tranquillamente essere sostituito da un while, se non fosse che è più conciso ed è concettualmente usato quando sappiamo a priori il numero di iterazioni che vogliamo fare. I parametri all'interno del for hanno diversi compiti, e sono separati da un punto e virgola:

Il primo viene eseguito prima di entrare nel ciclo, ed inizializza una variabile, generalmente la variabile è usata come variabile di controllo del ciclo, in poche parole servirà per tenere traccia del numero di iterazioni del ciclo; la variabile deve essere dichiarata fuori, prima del for;
Il secondo è la condizione (che coinvolge anche la variabile di controllo), che se risulta falsa interrompe l'esecuzione del ciclo;
Il terzo parametro è l'istruzione di incremento, che viene eseguita dopo ogni ciclo del for; questa istruzione agisce generalmente sulla variabile di controllo incrementandone (o decrementandone) il valore.
Per chiarire meglio l'uso del for presentiamo un semplice codice che conta da zero a cento, stampando la variabile di controllo:

```
int i;
for (i=0; i<=100; i++)
printf("%d n", i);
```

Funzioni rand() e srand()
La sola funzione **rand** però, ripete sempre la stessa sequenza di numeri. Anche eseguendo un miliardo di volte il programma la sequenza sarà sempre, noi vorremmo invece che la sequenza cambi ogni volta, in modo da avere ad ogni esecuzione sequenze diverse.
Ci viene in aiuto un'altra funzione, la **srand** che accetta come argomento un seme, permettendo alla funzione rand di generare una diversa sequenza di numeri casuali ad ogni esecuzione del programma, a patto che sia fornito ogni volta un seme diverso. Per generare ogni volta un seme diverso si può fornire come argomento alla srand l'orario in quell'istante tramite la funzione time, inclusa nell'header time.h della libreria standard del C.

I parametri formali di main
Differentemente dalle altre funzioni, sono sempre due, convenzionalmente chiamati **argc** e **argv**.

ARGOMENTI DELLA LINEA DI COMANDO

int argc: è un parametro di tipo intero. Rappresenta il numero degli argomenti effettivamente passati al programma nella linee di comando con cui si invoca la sua esecuzione. Anche il nome stesso del programma (nell'esempio, prog) è considerato un argomento, quindi argc vale sempre almeno 1.

char **argv: è un puntatore a un puntatore a carattere, ovvero un array di stringhe (La stringa è un array di caratteri e quindi potremmo dire che un array di stringhe è un array di array di caratteri).
Ciascuna stringa nel vettore contiene un diverso argomento. Gli argomenti sono memorizzati nel vettore nell'ordine con cui sono dati dall'utente.
argv[0] contiene il nome del programma stesso.

Costanti simboliche
Nel linguaggio C, le righe di codice che cominciano con il carattere '#' si dicono direttive; non sono vere e proprie istruzioni, ma istruiscono il compilatore a compiere alcune operazioni preliminari, prima della compilazione vera e propria.
Le direttive di tipo define permettono di associare valori fissi a sigle (costanti simboliche). Ad esempio, la direttiva *#define N 100* stabilisce che, da quel punto del programma in poi, ogni occorrenza della sigla N, dev'essere sostituita dalla sequenza 100, che, nei casi opportuni, sarà interpretata come numero.

- int atoi(char * s) – converte la stringa s nel corrispondente valore intero.
- double atof(char * s) – converte la stringa s nel corrispondente valore decimale.

-l'**algoritmo** è un procedimento che risolve una classe di problemi attraverso un numero finito di istruzioni elementari, chiare e non ambigue
-il **vettore** è una sequenza di dati omogenei..

-una **funzione** è una porzione di programma che svolge un compito specifico..
-un **record** è un insieme di dati non necessariamente omogenei..
-un **puntatore** è una variabile che contiene l'indirizzo della cella di memoria di un'altra variabile..
-la **lista concatenata** è una struttura dinamica di dati dove ogni elemento è un record..
-una **variabile** è un oggetto su cui opera un programma..
-la **CPU**, unità centrale di elaborazione, è l'unità che controlla ed esegue effettivamente tutte le istruzioni
-la **RAM**, memoria di lavoro, è una sorta di magazzino che contiene i programmi in corso di avvio e i dati sui quali operano..
-il **BUS** è il canale per mezzo del quale CPU e RAM comunicano
-il **metronomo**, clock, il quale da il tempo a tutti i componenti
-**record di attivazione** è lo spazio temporaneo contenuto nello stack quando una fazione viene chiamata, una volta terminata viene restituito il valore calcolato e cancellato il record di attivazione
-**unità di ingresso e uscita (input/output)** rappresentano l'interfaccia tra il calcolatore e il mondo
-le **memorie di massa** sono tipi di memoria che raccoglie grandi quantità di dati in maniera non volatile, ovvero permanente, però non possiedono una grande velocità di accesso
-il **sistema operativo** è una collezione di programmi che consente di eseguire le operazioni fondamentali del computer, dal punto di vista dell'utente il sistema operativo è un intermediario che consente di utilizzare il calcolatore e le sue parti attraverso interazioni verbali (parole o sigle) o gestuali (mouse, tastiera, menu, icone)
funzioni del sistema operativo
governo dei processi assegnare la **CPU** (unita di elaborazione) ai programmi in corso di esecuzione
governo della memoria di lavoro RAM assegnare aree di memoria di lavoro ai processi
governo delle operazione di ingresso ed uscita (I/O) operazioni che comportano scambi di dati tra CPU e RAM e le altre unità
governo del filesystem assegnare aree di memoria di massa ai file
-**passaggio di parametri per valore** è il normale assegnamento del valore di una variabile a una funzione, mentre il **passaggio di parametri per riferimento** alla funzione viene passato l'indirizzo di una variabile e consente di modificare il valore di variabili esterne alla funzione..
-**compilatore** è un programma informatico che traduce una serie di istruzioni scritte in un determinato linguaggio di programmazione in istruzioni di un'altro linguaggio (linguaggio macchina)
-**calcolatore** RAM, CPU, BUS, CLOCK copia-interpreta-esegue
tutte le informazioni conservate all'interno delle memorie di massa sono raccolte all'interno di file, i file a loro volta sono contenuti all'interno di cartelle che possono essere contenute in ulteriori cartelle.. questa ***struttura gerarchica è detta fillesystem.***

1. Chi era Alan Mathison Turing?
Turing è definito il padre dell'intelligenza artificiale, in base soprattutto alla teorizzazione dell'omonima macchina.
Durante la seconda guerra mondiale, Turing mise le sue capacità matematiche al servizio del Regno Unito per decifrare i codici usati nelle comunicazioni tedesche.

2. Che cos' la 'macchina di Turing'? Che relazione ha con i calcolatori?
In informatica una macchina di Turing è una macchina ideale che manipola i dati contenuti su un nastro di lunghezza potenzialmente infinita, secondo un insieme prefissato di regole ben definite. In altre parole si tratta di un modello astratto che definisce una macchina in grado di eseguire algoritmi e dotata di un nastro potenzialmente infinito su cui può leggere e/o scrivere dei simboli.

Introdotta nel 1936 da Alan Turing come modello di calcolo, è un potente strumento teorico che viene largamente usato nella teoria della calcolabilità e nello studio della complessità degli algoritmi, in quanto è di notevole aiuto agli studiosi nel comprendere i limiti del calcolo meccanico; la sua importanza è tale che oggi, per definire in modo formalmente preciso la nozione di algoritmo, si tende a ricondurlo alle elaborazioni effettuabili con macchine di Turing.

3. Che cos' un algoritmo? Dare una definizione il più possibile rigorosa.
L'algoritmo è un procedimento che risolve una classe di problemi attraverso un numero finito di istruzioni elementari, chiare e non ambigue.

4. Descrivere la struttura di un calcolatore.
Calcolatore: CPU, RAM, I/O, BUS, CLOCK
-la CPU, unità centrale di elaborazione, è l'unità che controlla ed esegue effettivamente tutte le istruzioni
-la RAM, memoria di lavoro, è una sorta di magazzino che contiene i programmi in corso di avvio e i dati sui quali operano..
-I/O, unità di ingresso e uscita costituiscono l'interfaccia fra calcolatore e mondo:
tastiera, schermo, schede di rete, altri dispositivi strani;
-il BUS è il canale per mezzo del quale CPU e RAM comunicano
-il metronomo, CLOCK, il quale da il tempo a tutti i componenti

5. Che cos' un bit? Che cos' un byte?
-Un **bit** è la più piccola unità di memoria di un computer, il suo nome deriva dalla contrazione di *Binary Digit* ovvero cifra binaria, è in grado di fornire 2 risposte 1 o 0. Ed è proprio questo che si intende per cifra binaria: un numero che può assumere due diversi valori in modo alternato.
-Il **byte** invece è una sequenza di 8 bit, questo vuol dire che può assumere 2^8=256 valori possibili

char=1 byte (8 bit)
int=2 byte (16 bit)
float=4 byte (32 bit)
double=8 byte (64 bit)

ogni cella della RAM vale 1 byte

6. Che cos' un sistema operativo? Quali sono le sue funzioni principali?
Il **sistema operativo** è una collezione di programmi che consente di eseguire le operazioni fondamentali del computer, dal punto di vista dell'utente il sistema operativo è un intermediario che consente di utilizzare il calcolatore e le sue parti attraverso interazioni verbali (parole o sigle) o gestuali (mouse, tastiera, menu, icone)
funzioni del sistema operativo
governo dei processi assegnare la **CPU** (unita di elaborazione) ai programmi in corso di esecuzione
governo della memoria di lavoro RAM assegnare aree di memoria di lavoro ai processi
governo delle operazione di ingresso ed uscita (I/O) operazioni che comportano scambi di dati tra CPU e RAM e le altre unità

7. Come si interagisce con un sistema operativo?
Dal punto di vista dell'utente il sistema operativo è un intermediario che consente di utilizzare il calcolatore e le sue parti. L'interazione tra utente e sistema operativo avviene attraverso due forme principali di interazione: gestuale e verbale.
Parliamo di interazione gestuale quando l'utente impartisce comandi al sistema operativo compiendo gesti che sfruttano dispositivi fisici (mouse, tastiera) ed elementi visivi (figurine, menu, finestre). Questa è la forma oggi largamente più diffusa.
Parliamo invece di interazione verbale quando i comandi sono impartiti nel corso di un dialogo, sotto forma di parole o sigle. In questo caso l'interazione

lascia una traccia su una console, che può occupare l'intero schermo oppure, ed è oggi il caso più frequente, una finestra sullo schermo. Le due forme di interazione possono quindi mescolarsi, e possiamo avere una sessione di interazione verbale nel contesto di un'interazione gestuale.
Una sessione di interazione verbale è governata da un processo, quindi dall'esecuzione di un particolare programma, detto shell, cioè "guscio".

8. Come si rappresentano i numeri, interi e reali, in un calcolatore? Che cosa significa 'virgola mobile'?

Nel linguaggio C, la rappresentazione in virgola mobile di un numero razionale float o double deriva dalla rappresentazione scientifica. Nella rappresentazione scientifica un numero è prodotto in due parti: la seconda, detta fattore di scala, è una potenza di 10, l'altra parte, detta parte frazionaria, è un numero tale che, moltiplicato per il fattore di scala, restituisce il numero che vuole rappresentare. Esistono dunque vari modi di rappresentare uno stesso numero, per esempio:

0,07824×105
0,7824×104
7,824×103
78240×10−1
le quattro notazioni sono equivalenti.

Viene però utilizzata la rappresentazione normalizzata: in essa, si impone che la parte frazionaria sia sempre minore di 1 e la cifra più significativa sia sempre diversa da 0. Quindi, nell'esempio sopra considerato, la notazione corretta è solo la seconda:

0,7824×104

La rappresentazione in virgola mobile è dunque la rappresentazione scientifica normalizzata con l'utilizzo del sistema binario; dunque il fattore di scala è una potenza di 2. La parte frazionaria viene detta mantissa mentre l'esponente della potenza di due è detto esponente. Il numero razionale è dunque così rappresentato:

MANTISSA x 2^ESPONENTE
in cui mantissa ed esponente possono avere segno + o segno −

9. Che cosa si intende per 'pseudocodice'?

In informatica uno **pseudocodice** è una via di mezzo tra il linguaggio naturale e un linguaggio di programmazione di alto livello. E' utilizzato dai programmatori per avere una bozza del programma da sviluppare. In uno pseudocodice si possono trovare le istruzioni tipiche dei linguaggio di programmazione ma in una forma più semplice e leggibile.

10. Che cos' un linguaggio di programmazione?

Un linguaggio di programmazione è una notazione simbolica, più vicina al nostro modo di ragionare, ha una grammatica: morfologia, sintassi e semantica.
Esistono molti linguaggi di programmazione; noi ne studieremo uno (chiamato C)

11. Che cosa si intende per 'strutture dei dati'? Elencate e descrivete qualche struttura di dati dinamica.

Strutture record, esempio di struttura di dati dinamica: liste concatenate
vedi descrizione sopra

12. Quali sono le strutture di controllo necessarie per esprimere un algoritmo?

Un qualsiasi algoritmo può essere espresso in un linguaggio di programmazione che disponga soltanto delle tre strutture di controllo: sequenza, selezione e iterazione. Questi sono, quindi, i mattoni di base con i quali è possibile costruire qualsiasi algoritmo.

13. Che cosa si intende per 'costo computazionale' di un algoritmo? Come si calcola?

Il costo computazionale di una funzione/programma è un costo definito in termini di risorse di calcolo. Le risorse di calcolo fondamentali sono due: quantità di tempo necessario alla computazione (tempo) quantità di memoria utilizzata (spazio)
Una misura del costo computazionale soddisfacente deve basarsi su un modello di calcolo astratto, indipendente dalla particolare macchina, essere una funzione (non un numero) e deve essere asintotica, cioè fornire una idea dell'andamento del costo all'aumentare della dimensione dell'input.

14. L'algoritmo di ordinamento "mergesort" ha un costo computazionale asintotico O(nlog n). Che cosa vuol dire? (a proposito: che cos' un algoritmo di ordinamento?)

Un algoritmo di ordinamento è un algoritmo che viene utilizzato per elencare gli elementi di un insieme secondo una sequenza stabilita da una relazione d'ordine, in modo che ogni elemento sia minore di quello che lo segue. Il **merge sort** è, appunto, un algoritmo di ordinamento basato su confronti che utilizza un processo di risoluzione ricorsivo, sfruttando una tecnica che consiste nella suddivisione del problema in sottoproblemi della stessa natura di dimensione via via più piccola.

15. Che differenza c'è fra costo computazionale di un algoritmo e costo computazionale di un problema?

Un problema può ammettere diversi algoritmi risolutivi. Intuitivamente, è ragionevole definire il costo del problema come il costo del miglior algoritmo. Ovviamente, in generale possiamo sospettare che esista un algoritmo migliore di quelli conosciuti, ma non averlo ancora scoperto. In alcuni casi, si può dimostrare che un certo algoritmo è il migliore possibile per un certo problema. Ad esempio, per il problema dell'ordinamento di un vettore, è dimostrato che non può esistere nessun algoritmo di costo inferiore (asintoticamente) a O(n log n), con il logaritmo in base 2, sotto l'ipotesi che l'algoritmo si basi su operazioni di confronto fra due elementi.
Quindi, in alcuni casi non conosciamo il costo computazionale di un problema, perché non siamo in gradi dire se il miglior algoritmo esistente sia ottimale; può succedere che il costo computazionale teorico di un problema si possa stabilire in via teorica, pur senza conoscere un algoritmo effettivo.

16. Che cosa si intende per correttezza di un algoritmo? Come possiamo verificare se un algoritmo corretto?

La prima caratteristica di un algoritmo è la correttezza, cioè deve fornire una soluzione corretta del problema e terminate.

Tempo e spazio di memoria sono risorse limitate di un calcolatore, per cui dovendo scegliere fra due algoritmi corretti, si preferirà quello che usa meno risorse.
L'efficienza di un algoritmo indica quanto parsimoniosamente esso utilizza le risorse a disposizione.
Per quantificare l'efficienza di un algoritmo al variare della numerosità dei dati su cui agisce si usa cercare di esprimere la sua complessità computazionale in termini di funzioni della numerosità dei dati n che fungano da limite inferiore, limite superiore, o di un andamento asintotico.

17. In che cosa consiste una 'ricerca binaria'?

L'algoritmo di **ricerca binaria** è un algoritmo che viene utilizzato per trovare elementi in un array ordinato, questo algoritmo viene utilizzato in molte applicazioni in maniera naturale, come ad esempio la ricerca di una parola in un vocabolario o la ricerca di un contatto telefonico in una rubrica. Quindi ad esempio in quest'ultimo caso, si apre la rubrica, verso l'elemento centrale e si confronta il valore trovato con la chiave da cercare.
Se la chiave è uguale abbiamo trovato subito l'elemento.
Altrimenti, se la chiave è maggiore allora si confronta con i valori successivi, scartando quelli precedenti, cioè gli elementi a destra del valore centrale.
Infine se la chiave è minore si confronta con i valori precedenti, scartando i successivi, cioè gli elementi a sinistra del valore centrale.

18. Che cosa sono lo 'standard input' e lo 'standard output' di un programma?

Lo standard input è un canale da cui giunge un flusso di dati.
Il flusso di input proviene dalla tastiera, e il programma trasferisce i dati effettuando operazioni di lettura. Quindi ogni lettura da tastiera viene vista come una lettura dal file standard input.
Quando un programma scrive qualcosa sullo schermo, sta usando una cosa che si chiama standard output . Lo standard output è quello che usano i programmi per scrivere le informazioni all'utente.
(lo **standard output** è quando un programma scrive qualcosa sullo schermo, mentre quando una persona vuole comunicare con il programma ad esempio con la tastiera o il mouse è definito **standard input**)

19. Che cos' una funzione in un programma? Come si definisce una funzione? Come avviene il passaggio dei parametri a una funzione?

vedi sopra definizione di funzione