```
#include <stdio.h>

int main(){
}
```

Come si dichiara una variabile?

```
type nome;

int x;
float y;
char c;
```

Come si inizializza una variabile?

```
nome = valore;

x = 2;
y = 2.4;
c = 'v';
```

Librerie

<stdio.h>: contiene le funzioni di input e output
- printf(); Stampa a video
- scanf(); Leggere dati in input dalla tastiera
- fprintf(); Scrivere su file
- fscanf(); Leggere da file

```
printf("Testo da stampare");
printf("Testo %d, %f, %c", x, y, c);
```
%d -> int
%f -> float
%c -> char

```
scanf(formato, &var)
scanf("%d %f %c", &x, &y, &c);
```
Legge dalla tastiera un intero, un float e un char e memorizza i valori in x, y, c

```
fprintf(file_in_cui_scrivere, formato, ...)
fprintf(file, "%d %f %c", x, y, c);
```
Scrive sul file, un intero, un float e un char

```
fscanf(file_da_cui_leggere, formato, ...)
fscanf(file, "%d %f %c", &x, &y, &c);
```
Legge dal file un intero, un float e un char e memorizza i valori in x, y, c

Istruzioni di I/O in C

Operatori Logici:
- (| |) : OR - è vero se almeno uno dei due è vero
0 or 0 = 0
0 or 1 = 1
1 or 0 = 1
1 or 1 = 1

- (&&) : AND - è vero se entrambi sono veri
0 and 0 = 0
0 and 1 = 0
1 and 0 = 0
1 and 1 = 1

- (!) : NOT
not 1 = 0
not 0 = 1

Operatori Aritmetici: +, -, /, *
- %: questo operatore restituisce il resto di una divisione

Operatori di Relazione
- > (>=): maggiore / maggiore uguale
- < (<=): minore / minore uguale
- == : uguaglianza
- != : diverso

Operatore di assegnazione
- = è l' operatore che permette di assegnare un valore ad una variabile.

Strutture di selezione (Cicli)

```
if(condizione){
	// blocco di codice che verrà eseguito se soltanto se la condizione è verificata
}else{
	// blocco di codice che verrà eseguito se soltanto se la condizione NON è verificata
}

=========
if(c1){
}else if(c2){
}else{
}
=========

if(c1){
	if(c2){
	}else{
	}
}else{
}
```

Come possiamo eseguire lo stesso blocco di codice, scrivendelo solo una volta?
Usando i cicli.

Stampare i numeri che vanno da 0...50

- for(contatore; condizione; incremento){}

```
for(int i  = 0; i <= 50; i++){
	printf("%d", i);
}
```

Il blocco di codice all'interno del ciclo for, verrà eseguito finché la condizione è verificata

- while(condizione){}

```
int i = 0;
while(i <= 50){
	printf("%d", i);
	i++;
}
```

Il blocco di codice all'interno del ciclo while, verrà eseguito finché la condizione è verificata

Passaggio da ciclo FOR a ciclo WHILE e viceversa

```
for(contatore; condizione; incremento){
      blocco di codice;
}
```

<=>

```
contatore;
while(condizione){
      blocco di codice;
      incremento;
}
```

- do{
  blocco di codice;
  }while(condizione)

A differenza del ciclo while nel quale si può anche non entrare mai se la condizione non è mai verificata, nel ciclo do while, almeno una volta si entra.

Strutture dati

È una collezione di dati omogenea. Al suo interno contiene dati di un solo tipo.

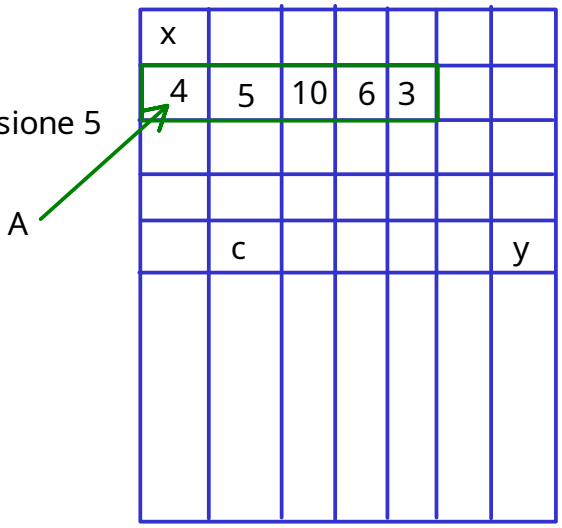
Array

Un' array è uno spazio di memoria contiguo, di dimensione FISSA, che può contenere dati di un unico tipo.

Contiguo: le celle sono adiacenti, una dopo l'altra

x, y e c sono variabili.

Sia A un array di dimensione 5 di interi



Sia N la dimensione di un array.
Le celle sono numerotate partendo da 0 a N-1

type nome_array[dimensione];

```
int A[5]; // Sia A un array di interi di dimensione 5 senza elementi
int A[5] = {4, 5, 10, 6, 3};
```

Come leggere un elemento da un array.

Per accedere alla i-esima cella di un array: nome_array[i]

```
int k = A[3];
```

Quanto vale k? k vale 6.

Come leggere un array in modo dinamico.

int A[200] = {1, 2, 3, 4, 5, 6, 7, ... 200}

Stampare tutti gli elementi di A.

```
for(int i = 0; i < 200; i++){
	printf("%d", A[i]); // accedo ad ogni cella di A e stampo il suo contenuto
}

int i = 0;
while(i < 200){
	printf("%d", A[i]);
	i++;
}
```

Da RIFARE!

Per casa: Sia A un array di interi di dimensione 20, definito come segue:
A = {20, 4, 10, 2, 110, 343, 54, 5, 0, 65, 94, 29, 12, 3, 84, 221, 9442, 9567, 32, 78}

Stampare gli elementi di A in questo modo:

A = [20, 4, 10, 2, 110, 343, 54, 5, 0, 65, 94, 29, 12, 3, 84, 221, 9442, 9567, 32, 78]

Calcolare la somma degli elementi di A.
Calcolare la differenza degli elementi di A.
Calcolare il massimo e il minimo di A.

Ricerca in un vettore

```
int len_A = 20;
int A[len_A] = {20, 4, 10, 2, 110, 343, 54, 5, 0, 65, 94, 40, 12, 3, 84, 221, 9442, 9567, 32, 78};
int b;
int i = 0;
scanf("%d", &b);
int trovato = 0;

//Dire se b è un elemento di A, nel caso lo fosse, dire in quale cella si trova, altrimenti -1
while(i < len_A && trovato == 0){
	if(b == A[i]){
		printf("%d si trova nella cella %d\n", b, i);
		trovato = 1;
	}
	i++;
}
if (trovato == 0){
	printf("%d non è presente in A\n", b);
}
```

La variabile "trovato" è usata per capire se durante il ciclo while abbiamo trovato "b" all'interno di "A". Se "b" si trova in "A" allora impostiamo "trovato" a 1 e nella prossima iterazione, non si entrerà più nel while.
Se "b" non si trova in "A" allora significa che non abbiamo mai impostato "trovato" a 1, dunque possiamo uscire dal while solo se "i > len_A" (abbiamo visto tutti gli elementi di A), quindi possiamo concludere grazie al if fuori dal while, che "b" non è presente in "A".

Per casa (oltre al compito precedente):
Implementare questo codice in C e provarlo con il array A

Caricamento di un array

```
int len_A = 20;
int A[len_A];

// caricare il array A con numeri presi da tastiera

for(int i = 0; i < len_A; i++){
      scanf("%d", &A[i]);
}
```

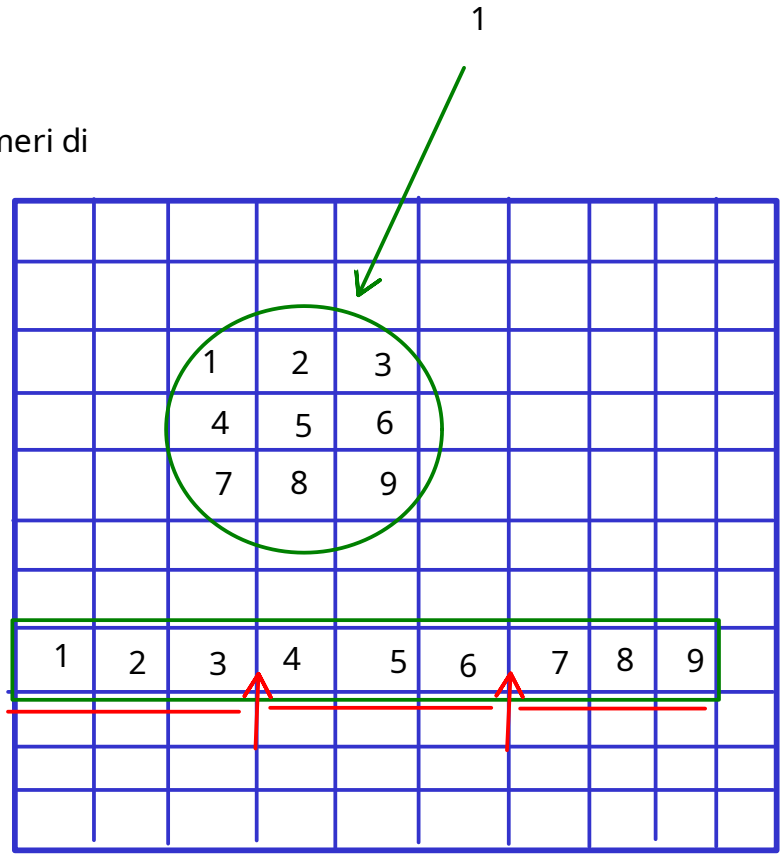
Per casa:
Implementare il caricamento di un array.
Stampare gli elementi dell'array in questo modo:
A = [n1, n2, n3, ...]

MATRICI

M(m x n): Una tabella di numeri di m righe e n colonne

M(3 x 3)

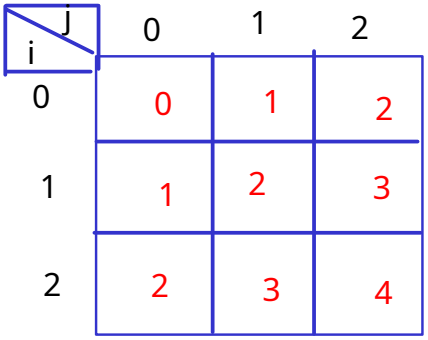
|1 2 3|
|4 5 6|
|7 8 9|



```
int M[righe][colonne]; // solo dichiarazione, senza elementi

int M[2][2] = {{1, 2}, {3, 4}}; // dichiarazione e inserimento elementi
|1 2|
|3 4|
```

Caricamento Matrice




```
int M[3][3];
// cicli for annidati
for(int i = 0; i < 3; i++){
      for(int j = 0; j < 3; j++){
            M[i][j] = i + j;
      }
}
```

```
for(int i = 0; i < 3; i++){
      for(int j = 0; j < 3; j++){
            printf("%d ", M[i][j]);
      }
      printf("\n");
}
```

```
0 1 2
1 2 3
2 3 4
```

Ricerca all'interno di una matrice

```
int M[3][3] = {{0 1 2},
               {1 2 3},
               {2 3 4}};

int n;
printf("Quale numero bisogna cercare?\n");
scanf("%d", &n);

for(int i = 0; i < 3; i++){
      for(int j = 0; j < 3; j++){
            if(n == M[i][j]){
                  printf("%d si trova nella cella (%d, %d)\n", n, i, j);
            }
      }
}
```

Ordinamento di un Array

A = {54, 2, 12, 65, 5, 21};
Ordinare A in ordine crescente

A = {2, 5, 12, 21, 54, 65};

len_A = 6;

BubbleSort

```
int temp;
for(int i = 0; i < len_A; i++){
    for(int j = 0; j < len_A - i - 1; j++){
        if(A[j] > A[j + 1]){
            temp = A[j];
            A[j] = A[j + 1];
            A[j + 1] = A[j];
        }
    }
}
```

Per casa: Implementare il bubbleSort in C.
Ordinare e stampare i seguenti array:

A = {54, 2, 12, 65, 5, 21};
B = {20, 4, 10, 2, 110, 343, 54, 5, 0, 65, 94, 29, 12, 3, 84, 221, 9442, 9567, 32, 78}

Implementare:
Caricamento e stampa e ricerca all'interno di una matrice