

## Puntatori

Una **VARIABLE PUNTATORE (pointer)** è una variabile che contiene l'indirizzo di una cella di memoria.

L'operatore & (referencing) prefisso al nome di una variabile fornisce l'indirizzo di memoria della variabile.

L'operatore \* (dereferencing) prefisso ad un indirizzo fornisce il contenuto in memoria dell'indirizzo.

Il **TIPO** di un puntatore è il tipo della variabile a cui punta

### Esempi

`int *ptr;` definisce ptr variabile puntatore che conterrà indirizzi di memoria di variabili int

`float *a;` definisce a variabile puntatore che conterrà indirizzi di memoria di variabili float

`int i = 7;`

`int *pippo;`

`pippo = &i;` pippo punta all'indirizzo l'indirizzo di memoria di i

`*pippo = 10;` ora la variabile i vale 10

## Vettori

Esempi di dichiarazione:

`int vet[4] = {5, 7, 1, 24};` definisce e inizializza un vettore di interi

`int vet[] = {5, 7, 1, 24};` definisce e inizializza lo stesso vettore

`float a[100];` definisce un vettore di 100 float, non sempre lo inizializza

`float a[100] = {0};` definisce un vettore di 100 float e li inizializza a 0

sintassi generale:

`tipo nome_array [num.elementi] = {valori iniziali};`

Gli elementi sono memorizzati in modo contiguo uno dopo l'altro, **gli indici partono da 0.**

**N.B. C++ non controlla l'accesso agli array per cui un accesso al di fuori dell'array (indice errato) non verrà segnalato e andrà a "sporcare" altre aree di memoria oppure a recuperare dati da altre aree**

**Alternative nella definizione:**

`const int IMAX = 100;`

`float a[IMAX];` // dimensiono così tutti i vettori e, se devo, cambio solo il valore di IMAX

**oppure**

`#define IMAX 100` // sostituisce tutti gli IMAX con 100

...

`float a[IMAX];`

## Puntatori e vettori (array monodimensionali)

**Il nome di un vettore è un puntatore.** Se definisco

```
float a[10];
```

**a**, il nome dell'array, è l'indirizzo dove l'array, lungo 10, inizia.

### Esempio

Definiti `float x`, e `float a[10]` è equivalente scrivere

```
x = a[0];
```

```
x = *a;
```

in entrambi i casi `x` conterrà il valore del primo elemento del vettore `a`; così come è equivalente scrivere

```
x = a[i];
```

```
x = *(a+i);
```

in tutti i casi `x` conterrà il valore dell' (i+1)-esimo elemento del vettore `a`.

## Vettori e Funzioni

**input:** se un vettore è un parametro di input, nella **definizione** della funzione lo indicherò seguito da parentesi quadre vuote:

```
float effe (... , float a[] , )
```

oppure col segno di **PUNTATORE**

```
float effe (... , float *a , )
```

e metterò tra i parametri di input anche la sua lunghezza. Nella **chiamata** alla funzione lo passerò semplicemente con il suo nome.

**output:** in C++ il **return** può restituire solo uno scalare; se voglio che la funzione mi renda un vettore dovrò inserire tale vettore tra i parametri di input e fare in modo che la funzione modifichi i suoi valori. Una funzione di questo genere, se non deve rendere altro che il vettore, va dichiarata **void**.

(vedi esempio)