

Puntatori

```
int *pointer;
```

Dichiara un puntatore a un intero.

```
char *pointer;
```

Dichiara un puntatore a un char.

```
int *const pointer;
```

Dichiara un puntatore costante. Il contenuto a cui punta può essere modificato.

```
int const* pointer; | | const int* pointer;
```

Dichiara un puntatore variabile. Il contenuto a cui punta non può essere modificato.

```
const int* const pointer;
```

Dichiara un puntatore costante a variabile costante. Il contenuto a cui punta non può essere modificato.

```
pointer = &variabile;
```

Assegna al puntatore l'indirizzo della variabile.

Il puntatore è un valore che indica l'indirizzo del dato a cui punta, non il suo contenuto.

Passare Array e Matrici alle Funzioni

Array	void nomeFunzione (int []);
-------	-----------------------------

Array	void nomeFunzione (int array[]);
-------	----------------------------------

Matrice a 2 dimensioni	void nomeFunzione (int matrice[][DIM2]);
------------------------	--

Matrice a 3 dimensioni	void nomeFunzione (int matrice[][DIM2][DIM3]);
------------------------	--

VLA	void nomeFunzione (int array[], int size);
-----	--

VLA a 2 dimensioni	void nomeFunzione (int array[][*], int size, int size2);
--------------------	--

VLA a 3 dimensioni	void nomeFunzione (int array[][*][*], int size, int size2, int size3);
--------------------	--

1 e 2 sono equivalenti.

Non è necessario passare il nome di un array monodimensionale.

Non è necessario passare il nome delle dimensioni, basta specificare il tipo.

File

```
FILE *filepointer;
```

Definisce il puntatore al file.

```
filepointer = fopen("file.txt", "w");
```

Inizializza il puntatore per aprire il file e ne definisce la modalità di apertura.

```
fclose(filepointer);
```

Chiude lo stream del file.

```
fprintf(filepointer, "testo", variabili...);
```

Stampa nel file il testo e il contenuto delle variabili.

```
fscanf(filepointer, "%formato", &variabili...);
```

Riceve dal file l'input desiderato.

```
fwrite(origine, dimensioneInByte, numElementi, filepointer);
```

Scrive sul file le informazioni presenti nell'origine.

```
fread(destinazione, dimensioneInByte, numElementi, filepointer);
```

Legge e scrive in destinazione le informazioni presenti nel file.

fwrite e fread restituiscono il valore intero di quanti elementi sono stati scritti/letti.

Formato dati		
Data Type	Printf	Scanf
int	%d	%d
unsigned int	%u	%u
long int	%ld	%ld
unsigned long int	%lu	%lu
long long int	%lld	%lld
unsigned long long int	%llu	%llu
double	%f	%lf
float	%f	%f
hex integer	%x	%x
decimal exponent	%e	%e
char	%c	%c
unsigned char	%c	%c
string	%s	%s
%pointer	%p	%p

Sequenze di Escape	
Escape	Azione
\n	Va a capo.
\t	Va alla tabulazione successiva.
\a	Produce suono o allarme visibile senza cambiare posizione del cursore.
\\	Scrive un backlash.
\"	Scrive le doppie virgolette.
\?	Scrive il punto interrogativo.
\'	Scrive un'apostrofo.

Operatori Logici	
Operatore	Simbolo
AND	&&
OR	
NOT	!
NOT è unario	
EX: !a	

Definire Array e Matrici
<pre>int i[10];</pre> Definisce un array di interi di grandezza 10.
<pre>int i[10] = {0};</pre> Definisce un array di interi di grandezza 10 e inizializza tutte le celle a 0.

Definire Array e Matrici (cont)

```
char i[10];
```

Definisce un array di char di grandezza 10: può contenere al massimo 9 caratteri.

```
char i[] = " cia o";
```

Definisce un array di char di grandezza 5: 4 lettere e il terminatore di stringa '\0'.

```
char i[] = {'c', 'i', 'a', 'o', '\0'};
```

Equivalente al precedente.

```
int i[2][3];
```

Definisce una matrice 2*3.

```
int i[2][3] = {{1, 2, 3},
               {4, 5, 6}};
```

Definisce una matrice 2*3 e inizializza ogni riga con valori da 1 a 6.

```
int i[variabile];
```

Definisce un array delle dimensioni conservate in *variabile*. Caso di VLA.

```
int i[SIZE];
```

Definisce un array di dimensioni pari alla costante simbolica *SIZE*.

Un vettore preceduto da **const** non può essere modificato nel suo contenuto.

Definire una Struttura

```
struct dati
{
    int sensore;
    char string a[10];
    float media;
};
```

Una struttura include variabili e array anche di tipi diversi.

Se la struttura è usata da più funzioni, porre la struttura fuori dal corpo della main.

Tipi di Variabili	
Type	Scope
Extern	Prefisso che indica una variabile dichiarata in un modulo (es .h) esterno al modulo chela usa.
Global	Dichiarata esternamente alle funzioni e visibile in ogni funzione
Static	Come locale ma non è distrutta quando la funzione cede il controllo al chiamante
Local	Ha una visibilità limitata ad una funzione o una iterazione (for).



Tipi di Variabili (cont)

SI

```
#include "myfile.h"

extern int var;

int main(void)
{
    var = 10; return 0;
}
```

NO

```
extern int var;

int main(void)
{
    var = 10;
    return 0;
}
```

Con extern dichiarare dove si trova la variabile.
EX: myfile.h

Incremento e Decremento

Formato Azione

a++	Rende disponibile la variabile, poi la incrementa
++a	Incrementa la variabile, poi la rende disponibile
a--	Rende disponibile la variabile, poi la decrementa
--a	Decrementa la variabile, poi la rende disponibile
a +=	Somma ad a la parte destra, poi assegna ad a
a -=	Sottrae ad a la parte destra, poi assegna ad a
a *=	Moltiplica a per a la parte destra, poi assegna ad a
a /=	Divide a per a la parte destra, poi assegna ad a
a % =	Trova il resto tra a e la parte destra, poi assegna ad a

Manipolazione su bit

Simbolo	Azione
x >> y	x scorre a destra di y bit
x << y	x scorre a sinistra di y bit
x & y	AND bit a bit tra x e y
x y	OR bit a bit tra x e y
x ^ y	XOR bit a bit tra x e y

Manipolazione su bit (cont)

~x Complemento dei bit di x
~ è un operatore unario.

Operatore Ternario

(condizione) ? Espressione 1 : Espressione 2 ;

Se la condizione è **vera** verrà eseguita Espressione 1;
Se la condizione è **falsa** verrà eseguita Espressione 2.

Generare Numeri Random

Genera numeri casuali

```
i = rand();
```

Genera numeri casuali in un intervallo preciso

```
i = minimo + rand() % (massimo - minimo + 1);
```

Scegliere un seme sempre diverso

```
srand(time(NULL));
```

Per utilizzare rand() e srand() è necessario includere <stdlib.h>.
Per utilizzare time(NULL) è necessario includere <time.h>.

Modalità Apertura File

r	Lettura	Se il file non esiste, fopen() restituisce NULL.
r+	Lettura e Scrittura	Se il file non esiste, fopen() restituisce NULL.
w	Scrittura	Se il file non esiste viene creato, se esiste viene sovrascritto.
w+	Lettura e Scrittura	Se il file non esiste viene creato, se esiste viene sovrascritto.
a	Append	Se il file non esiste viene creato.
a+	Lettura e Append	Se il file non esiste viene creato.
rb	Lettura binaria	Se il file non esiste, fopen() restituisce NULL.
rb+	Lettura e Scrittura binaria	Se il file non esiste, fopen() restituisce NULL.
wb	Scrittura binaria	Se il file non esiste viene creato, se esiste viene sovrascritto.



Modalità Apertura File (cont)

wb+	Lettura e Scrittura binaria	Se il file non esiste viene creato, se esiste viene sovrascritto.
ab	Append binario	Se il file non esiste viene creato.
ab+	Lettura e Append	Se il file non esiste viene creato.

Operazioni sui Puntatori

Operazione	Azione
pointer + i	Sposta il puntatore di i posizioni in avanti.
pointer - i	Sposta il puntatore di i posizioni indietro.
pointer1 - pointer2	Quanti dati del tipo puntato possono inserire tra la posizione di pointer1 e pointer2.
pointer++	Sposta di una posizione il puntatore.
Spostare significa aumentare o diminuire di tot. byte (in base alla dimensione del tipo puntato) l'indirizzo del puntatore. 'i' è un valore intero.	

Precedenze

Operatore	Descrizione	Associatività
()	Parentesi/Chiamata a funzione	Sinistra - Destra
[]	Parentesi/Indice Array	Destra
++ --	Postincremento/Postdecremento	
++ --	Preincrementi/Predecrementi	Destra - Sinistra
(type)	Cast	
*	Indirizzione	
&	puntatore/Deferenziazione	
sizeof	Indirizzo puntatore Operatore sizeof	
* / %	Moltiplicazione, Divisione e Modulo	Sinistra - Destra
+ -	Somma e sottrazione	Sinistra - Destra

Precedenze (cont)

<< >>	Shift bit a bit	Sinistra - Destra
< <= > >=	Confronto relazioni	Sinistra - Destra
== !=	Confronto uguaglianza	Sinistra - Destra
&&	AND Logico	Sinistra - Destra
	OR Logico	Sinistra - Destra
? :	Operatore ternario	Destra - Sinistra
=	Assegnazione	Destra - Sinistra
+= -= *= /= %=	Operazione e assegnazione	
,	Operatore virgola	Sinistra - Destra

Pulire Schermo

```
#include <windows.h> ||
#include <process.h> ||
#include <stdlib.h>

| Inserire la libreria necessaria al funzionamento.

system("pause");

| Chiede all'utente di premere un tasto per continuare.

system("cls"); //windows | |
system("clear"); //linux

| Pulisce lo schermo dal testo precedente.
```