

Tipi di dati e array

int	Intero (-1,0,1,2...)
float	Numero con la virgola
bool	True o False
char	Carattere ('a','b'...)
array	int arr[5] = {1, 2, 3, 4, 5};

Input/Output

using namespace std;	Evita di specificare std::
cin>>variabile;	Inserire a schermo un valore
cout<<variabile;	Stampa a schermo variabile
cout<<"frase";	Mostra a schermo "frase"
cout<<"..."<<endl;	Consente di andare a capo

Struttura ciclo while

```
while(condizione == true) {
// istruzioni eseguite fino a quando
// la condizione è vera
}
```

Struttura ciclo for

```
for (int i = 0; i < n; i++){
// ...
// istruzioni eseguite n volte
}
```

Struttura funzione

```
int funzione(parametri){
//Codice
return 0
}
}
Uso void se la funzione non restituisce
niente
altrimenti uso int se restituisce un intero
oppure bool o char...
```

Struttura if

```
if (condizione) {
//istruzioni
}
```

Condizioni

A==B	Se A è uguale a B, restituisce vero
A!=B	Se A è diverso da B, restituisce vero
A<B	Se A è minore di B, restituisce vero
A>B	Se A è maggiore di B, restituisce vero
A<=B	Se A è minore o uguale di B, restituisce vero
A>=B	Se A è maggiore o uguale di B, restituisce vero
A!B	A not B
A&B	A and B, vero se entrambe le condizioni sono vere
A B	A or B, vero se almeno una condizione è vera

Operatori

+	Addizione	a + b
-	Sottrazione	a - b
*	Moltiplicazione	a * b
/	Divisione	a / b
%	Modulo (Resto)	a % b
++	Incremento	a++
--	Decremento	a--

Random

```
n = rand()%max + min;
numero casuale
n va da min fino a min + max - 1
Senza min va da 0 a max - 1
esempio:
da 30 a 50 faccio rand()%21 + 30
metto come primo numero la differenza e
come secondo numero il minimo
```

fork()

```
int p = fork()    Creo un processo figlio
Restituisce:
0 se sei nel processo figlio
>0 se sono nel padre, p è il PID del figlio
-1 se il figlio non è stato creato
```

robe

exit(n)	Termino il processo in cui mi trovo
	n è il codice di uscita ()
	0 < n < 255
kill(p, SIGKILL);	Faccio terminare immediatamente il processo con PID=p
wait()	Restituisce il PID del figlio terminato
int status;	Qui status prende tutte le informazioni di come è terminato il processo figlio
int pid = wait(&status);	
WIFEXITED(status)	vero se il figlio è terminato normalmente
WEXITSTATUS(status)	codice di uscita passato a exit()
WIFSIGNALED(status)	vero se il figlio è terminato da un segnale
WTERMSIG(status)	segnala quale segnale ha terminato il figlio

i trattini nella tabella non si devono mettere
indica solo che va a capo

Librerie

#include <iostream>	
#include <unistd.h>	per fork() ...
#include <sys/wait.h>	per wait() ...
#include <signal.h>	per kill()...
#include <cstdlib>	per rand()