16. Istruzioni di un programma
Un programma consiste in un insieme di funzioni, eventualmente suddivise in più file.
La funzione che costituisce il programma principale si deve necessariamente chiamare **main**. Tutto ciò che il programmatore invoca, parte dal main.

Ogni programma contiene una lista di istruzioni di ogni tipo: istruzioni **semplici** (la base per costruire il programma) o istruzioni **strutturate**.

return 0; //un programma corretto restituisce come valore 0
0 rappresenta il valore di ritorno della funzione main. I valori di ritorno possono essere usati nelle shell per scrivere programmi di più alto livello che indicano operazioni più complesse.

16.1 Istruzioni semplici
Sono alla base delle istruzioni più complesse e terminano con un punto-e-virgola ';'.

Si distinguono in:
- **definizioni/dichiarazioni** (declaration-statement):
  - variabili e costanti (*int x, y; const float pi=3.14*)
- **espressioni** (expression-statement):
  - di input (*cin >> x*)
  - di output (*cout << 3*x*)
  - di assegnamento (*x=2*(3-y)*)
  - matematiche (*(x-3)*sin(x)*)
  - logiche (*x==y && x!=z*)
  - costanti (*3*12.7*)
  - condizionali

! Ogni espressione seguita da un ";" è anche un'istruzione.

16.1.1 L'espressione condizionale
Sintassi:
exp1 ? exp2 : exp3;

Se *exp1* è vera si fa *exp2* altrimenti si fa *exp3*.

Esempi
*prezzo = valore * peso * ((peso>10) ? 0.9 : 1);*

Se peso>10, *prezzo=valore*peso*0.9*.
Se peso <=10, *prezzo=valore*peso*1*.

*valoreass=(a-b) * ((a>b) ? (1) : (-1));*

! Meglio sempre mettere le parentesi.

16.1.2 Esempio di utilizzo dell'espressione condizionale
Prendiamo il caso di un programma che calcola le soluzioni di un'equazione di secondo grado. Esse possono essere 2. Per rendere l'output più completo, anziché scrivere *L'equazione ha 1 soluzione/i* sarebbe meglio scrivere due diversi output: *L'equazione ha 1 soluzione* e *L'equazione ha 2 soluzioni*. Per evitare di rendere il nostro codice troppo elaborato si può semplicemente scrivere così:

cout << "L'equazione ha ";
(x1 == x2) ? cout << "un'unica soluzione" : cout << "due soluzioni" ;

16.2 Le istruzioni strutturate
Le istruzioni strutturate consentono di specificare azioni complesse,
Si distinguono in

- istruzione composta (compound-statement)
- istruzioni condizionali (conditional-statement)
- istruzioni iterative (iteration-statement)
- istruzioni di salto (jump-statement)

16.2.1 Istruzione composta
Trasforma una sequenza di istruzioni in una singola istruzione.
La sequenza viene delimitata per mezzo della coppia di delimitatori '**{**' e '**}**' e viene chiamata **blocco**.

16.2.1.1 Definizioni di variabili nei blocchi
Le definizioni possono comparire in qualunque punto del blocco. Sono visibili solo all'interno del blocco, ma possono accedere ad oggetti definiti esternamente.
In caso di identificatori identici, prevale quello più interno.

Esempio

```
n=5;
n++;   //n=6

{
int n=3;
n+=4;

cout << n;      //n=7
}

cout << n;      //n=6
```

Nel caso in cui nel blocco non avessi definito una nuova variabile n (*int n=3*), nella operazione *n+=4* avrei modificato la *n* del blocco main e avrei avuto 10 (anche nella funzione cout fuori dal blocco).

```
n=5;
n++;   //n=6

{
n+=4;

cout << n;      //n=10
}

cout << n;      //n=10
```

! Se in un blocco viene modificata una variabile creata fuori dal blocco la modifica tocca anche le operazioni fatte con la variabile fuori dal blocco.
! Se viene definita una variabile omonima nel blocco avrà un valore diverso rispetto a quello fuori dal blocco

17. Istruzione condizionale
La più semplice è l'istruzione **if-then**

Sintassi:
if (exp)
  istruzione1;

Se *exp* è vera (=*true* o *1*) viene eseguita istruzione1 altrimenti non viene eseguito nulla. *istruzione1* può essere a sua volta un'istruzione complessa (come un'istruzione composta).

Esempi
```
if (x!=0)
  y=1/x;

if (n%d==0) {
 y=n/d;

 cout << y;
}
```

Se d=0, verrà segnato già nella condizione un errore di runtime.

17.1 L'istruzione if-then-else
Sintassi:
```
if (exp)
  istruzione1;
else
 istruzione2;
```

Se exp è vera vine eseguita *istruzione1*, altrimenti viene eseguita *istruzione2*. Entrambe possono essere istruzioni complesse (un blocco, altri if-then-else).

Esempio
```
if (x<0)
  y=-x;
else
  y=x;
```

17.2 if annidati
Nei costrutti if-then e if-then-else, istruzione1 e istruzione2 possono essere a loro volta istruzioni complesse (un blocco, un'altro if-then-else, …).

L'annidamento di if-then-else e l'uso di operatori logici permettono di costruire strutture decisionali complesse.

N.B. È molto importante indentare bene il codice.

Esempi

```
if (exp)
  istr;
else
  if (exp)
     istr1;
  else
    istr2;
```

OPPURE

```
if (exp)
  istr;
else if (exp)
  istr1;
else
  istr2;
```

! Si usano gli operatori booleani (&& e ||) per concatenare le condizioni degli if.
Con && si esegue l'istruzione se entrambe le condizioni sono vere, con || se almeno sola è vera.

18. L'istruzione condizionale *switch*

Sintassi

```
switch (exp) {
case const-exp1: istruzione1; break;
case const-exp2: istruzione2; break;
…
default: istruzione-default;
}
```

L'esecuzione dell'istruzione switch consiste:

- nel calcolo dell'espressione *exp*
- nell'esecuzione dell'istruzione corrispondente all'alternativa specificata dal valore calcolato
- se nessuna alternativa corrisponde, se esiste, viene eseguita *istruzione-default*

Se *exp==const-exp1* si esegue *istruzione1*, altrimenti si verificano *exp* e *const-exp2*, se *exp==const-exp2* si esegue *istruzione2*, altrimenti si passa al *case* successivo. Se non è mai verificata la condizione si esegue l'istruzione di default.

Esempio:

```
switch (op)                    //op è char
  {
   case '+': cout << op1+op2; break;
   case '-': cout << op1-op2; break;
   case '*': cout << op1*op2; break;
   case '/': cout << op1/op2; break;
   default : cout << "Errore!";
  }
```

18.1 Scelte multiple con *switch*

Se dopo l'ultima istruzione di un'alternativa non c'è un *break*, viene eseguita anche l'alternativa successiva.

Questo comportamento è sconsigliato ma può essere giustificato in alcuni casi (scelte multiple del tipo if-then-else).

Esempio

```
switch (giorno) {
case lun:
case mar:
case mer:
case gio:
case ven: orelavorate+=8; break;
case sab:
case dom: break;
}
```

Se *giorno==lun* or *giorno==mar* … fino a *ven* si incrementa la variabile *orelavoate*, altrimenti se *giorno==sab* or *giorno==dom* si esce dallo *switch*.