Istruzioni di selezione

Sono istruzioni che permettono per l'appunto di "selezionare" il codice da eseguire; infatti, queste consentono di deviare la prosecuzione del programma attraverso l'utilizzo di espressioni logiche chiamate condizioni.

Condizioni

La Treccani dice:

> **condizióne** s. f. […] **1.** In generale, **fatto** il cui intervento è necessario perché un altro fatto possa verificarsi;
>
> Treccani

In particolare:

> […] nella fisica, relazione cui occorre soddisfare perché si verifichi un determinato fatto […]
>
> Treccani

L'ultima definizione la trovo estremamente calzante anche nell'informatica; infatti, noi utilizziamo le condizioni per condizionare il programma e dirigere i dati in modo da raggiungere la soluzione più soddisfacente. Trovo estremamente affascinante pensare che tutti i programmi che utilizziamo quotidianamente si basino su delle domande e delle risposte.

Nelle prossime pagine capiremo meglio cosa sono le istruzioni di selezione e come utilizzare nel linguaggio C++ ed in qualsiasi altro linguaggio.

if()

La dichiarazione if ci permette di porre una condizione sui dati utilizzati, in italiano possiamo tradurlo come “se il dato ha questo valore allora esegui questo codice” e ci permette di porre una domanda alla quale il programma può rispondere vero se la condizione risulta verificata o falso se la condizione non si verifica.

```
if( condizione ){
        //Istruzioni se la condizione è vera
}
```

Cosa significa if?

Per spiegare meglio il funzionamento dell’if prenderò come esempio una situazione reale:

“Io ho una macchina fotografica funzionante, se si rompe dovrò ripararla”

Possiamo riassumere la frase nei seguenti attori:

Fatto	Ho una macchina fotografica	Dati iniziali
Condizione	(se) si rompe	Domanda su una condizione precisa
Conseguenza	dovrò ripararla.	Azioni successive

In informatichese:

```
int main(){
        “Ho una macchina fotografica”                   //Fatto,
        [se] if(“si rompe”){                            //Condizione
                “dovrò ripararla”;                      //Conseguenza
        }
}
```

Infine, in C++ abbiamo:

```
int main(){
        bool fotocamera_integra = true;
        cout << “La macchina fotografica è funzionante?”;
        cin >> fotocamera_integra;
        if( fotocamera_integra == false ){
                ripara_macchina_fotografica();
        }
}
```

else

Nel paragrafo precedente abbiamo visto come muoverci nel caso in cui volessimo porre una condizione specifica, nel caso in cui volessimo specificare le operazioni da eseguire in tutti i casi contrari alla condizione definita utilizziamo l'istruzione else;

Infatti, l'else ci permette di specificare come si deve comportare il codice nel caso in cui la condizione sia falsa.

```
If( a > b ){
        cout << "a è maggiore di b";
}
else{
        cout << "a non è maggiore di b, ma al contrario a è minore o uguale a b (a <= b)";
}
```

Questa istruzione serve nel caso in cui noi volessimo specificare quale codice eseguire nelle condizioni opposte a quella dichiarata nell' if.

Attenzione: proprio per il fatto che l'else permette di specificare una condizione verificata non si può utilizzare questo costrutto a meno che non venga specificato precedentemente un if.

Attenzione: **l'else viene eseguito solo e soltanto se la condizione dell'if non è verificata**, il che è diverso dal dichiarare più if successivi fra loro.

Altri esempi con l'istruzione else

Esempio 1

```
int a;
int b;
cin >> a >> b;

If(a > b){
        cout << "a è maggiore di b \n";
}
else{
        cout << "a non è maggiore di b \n";
}
```

Esempio 2

```
char a;
cin >> a;

if(a > 'a' && a < 'z'){
        cout << "a è una lettera minuscola \n";
}
else{
        cout <<"a non è una lettera minuscola \n";
}
```

Esempio 3

```
int a;
cin >> a;

if(a % 2 == 0){
        cout <<  "a è pari \n";
}
else{
        cout << "a non è pari quindi è dispari \n";
}
```

else if()

Infine, l'else if ci permette di definire delle condizioni alternative all'if più specifiche.

Per capirne il funzionamento prendiamo in considerazione la seguente condizione

"Se il numero inserito è pari lo si indichi in output "

Possiamo scrivere:

```
If(a % 2 == 0){
        cout << "a è pari";
}
```

Trasformando la condizione nel seguente modo:

"Se il numero inserito è pari lo si stampi in output,
se è dispari (cioè tutte le possibilità opposte e rimanenti) lo si stampi in output"

Otteniamo:

```
If(a % 2 == 0){
        cout << "a è pari";
}
else{
        cout << "a è dispari";
}
```

Infine, nel caso in cui la condizione diventi:

"Se il numero inserito è pari lo si stampi in output,
se è dispari (cioè tutte le possibilità opposte e rimanenti) lo si stampi in output,
in particolare se il numero dispari è anche minore di 7 si stampi in output la esclamazione "AH!" "

```
int a;
cin >> a;
If(a % 2 == 0){                                 //Condizione iniziale (pari)
        cout << "a è pari";
        //Esempi di valori in questa condizione: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18
        //Il numero 0 è pari, link
}
else if(a < 7){                                 //Condizione specifica diversa da quella iniziale (dispari minore di 7)
        cout << "AH!";
        //Esempi di valori in questa condizione: ..., -13, -9, -7, -5, -3, -1, 1, 3, 5
}
else                                            //Condizione opposta a quelle prima dichiarate (dispari e maggiore o uguale a 7)
        cout << "a è dispari";
        //Esempi di valori in questa condizione: 7, 9, 11, 13, 15, 17, ...
}
```

Altri esempi con l'istruzione else if()

Esempio 1

```
int a;
int b;
cin >> a >> b;

If(a > b){
        cout << "a è strettamente maggiore di b \n";
}
else if(a == b){
        cout << "a è uguale a b";
}
else{
        cout << "a è strettamente minore di b \n";
}
```

Esempio 2

```
char a;
cin >> a;

if(a > 'a' && a < 'z'){
        cout << "a è una lettera minuscola \n";
}
else if(a > '0' && a < '9'){
        cout << "a non è una lettera minuscola \n";
        cout << "a è una cifra \n";
}
else{
        cout << "a è uno dei simboli restanti \n';
}
```

Esempio 3

```
int a;
cin >> a;

if(a % 2 == 0){
        cout << "a è pari \n";
}
else if(a > 13){
        cout << "a è dispari è dispari ed è maggiore di 13 \n";
}
else if(a < -5){
        cout << "a è dispari e minore di -5 \n"
}
```

switch() case

Questa istruzione permette di condensare le istruzioni if(), else if() ed else in un solo costrutto a mio parere molto elegante, l'unica differenza è che è possibile specificare una condizione soltanto all'interno delle parentesi tonde: a seconda del risultato di quest'ultima si attiveranno i casi specificati.

Confronto fra switch() case e le istruzioni if(), else if(), else

```
int a;
cin >> a;

switch(a){
case 1:
        cout << "a è uguale ad 1 \n"
        break;
case 2:
        cout << "a è uguale ad 2 \n"
        break;
case 13:
        cout << "a è uguale ad 13 \n"
        break;
default:
        cout << "a è diverso da tutti i casi precedenti \n";
        break;
}
```

```
int a;
cin >> a;


if( a == 1){
  cout << "a è uguale ad 1 \n";
}
else if( a == 2){
  cout << "a è uguale a 2 \n";
}
else if( a == 13){
  cout << "a è uguale a 13 \n";
}
else{
        cout << "a è diverso da tutti i casi precedenti \n";

}
```

Notare bene:

- L'istruzione default permette di definire una o più azioni per tutti i casi rimanenti.
- break è un operatore che "rompe" l'esecuzione del programma: quando il processo arriva a questo punto quest'ultimo esce dallo switch e continua con le istruzioni successive.