

# Laboratorio di programmazione per sistemi ciberfisici

## 5. Costrutti decisionali

*Enrico Martini*  
*November 12, 2025*

## Espressioni logiche

Il tipo `_Bool` rappresenta solo 1 (TRUE) e 0 (FALSE).

Esistono 3 tipi di operatori che lavorano con booleani:

1. Operatori relazionali
2. Operatori di uguaglianza
3. Operatori logici

## Operatori relazionali

Sono maggiore (>), minore (<), minore o uguale (<=), maggiore o uguale (>=).

- ▶ Confrontano due numeri, anche di tipo diverso
- ▶ Restituiscono come valore un booleano
- ▶ La precedenza è inferiore rispetto agli operatori matematici
- ▶ Sono associativi a sinistra:  $0 < i < 15 \rightarrow (10 < i) < 15$

Esempio:

```
1 10 > 5           // 1
2 10.0 < 6          // 0
3 int x = 6; x >= 6 // 1
```

## Operatori di uguaglianza

Sono uguale (==) e diverso (!=).

- ▶ Restituiscono come valore un booleano
- ▶ La precedenza è inferiore rispetto agli operatori matematici ma uguale a quelli relazionali
- ▶ Sono associativi a sinistra

Esempio:

```
1 10 == 5           // 0
2 10.0 != 6         // 1
```

## Operatori logici

Sono negazione (!), and ( && ) e or (||).

- ▶ Restituiscono come valore un booleano
- ▶ La precedenza di and e or è inferiore rispetto agli operatori relazionali e di uguaglianza. L'operatore negazione ha la precedenza.
- ▶ L'operatore negazione è associativo a destra; gli altri sono associativi a sinistra

Esempio:

```
1  !1           // 0
2  1 && 0       // 0
3  4 < 5  && 5 == 6  // 0
4  4 == 5 || 6 == 6  // 1
```

A	B	!A	A && B	A    B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

## Il costrutto if

Una proprietà fondamentale di un programma è la capacità di prendere decisioni.

```
1  if (espressione){  
2      istruzioni...  
3  }
```

Solo se l'espressione è vera allora verranno eseguite le istruzioni contenute tra parentesi graffe.

## Esempio

Scrivere un programma che chieda all'utente un numero e ne stampi il valore assoluto.

```
1  #include <stdio.h>
2  int main(){
3      float n;
4      scanf("%f",&n);
5      if (n < 0){           // Se n e' negativo allora entra e inverte
6          n = -n;
7      }
8      printf("Valore assoluto: %i\n", n);
9      return 0;
10 }
```

## Il costrutto if-else

Utile quando ci sono due possibilità.

```
1  if (espressione){  
2      blocco istruzioni A;  
3  }  
4  else {  
5      blocco istruzioni B;  
6  }
```

Verrà sempre eseguito o il blocco A o il blocco B, mai entrambi.



## Il costrutto if - else if

Può essere utilizzato più volte di seguito.

```
1  if (espressione 1){  
2      blocco istruzioni A;  
3  }  
4  else if (espressione 2){  
5      blocco istruzioni B;  
6  }  
7  else{  
8      blocco istruzioni C;  
9  }
```

## Il costrutto switch

Utile quando si devono prendere decisioni basate solamente sul valore di un'espressione.

```
1 switch (espressione){  
2     case valore1:  
3         blocco_istruzioni_1;  
4         break;  
5     case valore2:  
6         blocco_istruzioni_2;  
7         break;  
8     ...  
9     default:  
10        blocco_istruzioni_N;  
11        break;  
12 }
```

- ▶ Le istruzioni dentro al case non devono essere racchiuse tra parentesi graffe
- ▶ L'istruzione break deve essere presente alla fine di ogni case; In caso contrario andrà ad eseguire anche il case successivo
- ▶ Il default ha la stessa funzione dell'else, catturando tutti gli altri casi non specificati
- ▶ Non possono esserci due case di valore uguale

## L'operatore condizionale

E' un operatore ternario che viene scritto su un'unica riga.

```
1 condizione ? espressione1 : espressione2;
```

Viene utilizzato spesso per assegnare a una variabile valori differenti a seconda della condizione.

Esempio: *Scrivere un programma che stampi 'p' se il numero inserito è maggiore di zero, 'n' altrimenti.*

```
1 int x;  
2 scanf("%i\n",&x);  
3 char s = (x > 0) ? 'p' : 'n';  
4 printf("%c\n",s);
```

## Esercizio 1

Verifica che un numero intero inserito dall'utente sia pari o dispari.

Esempio:

```
1 Inserisci un numero: 9
2 Il numero 9 è dispari.
```

## Esercizio 2

Scrivere un programma che chieda all'utente tre voti, ne faccia la media e scriva che l'esame è superato se la media di questi voti è maggiore o uguale a 18.

Esempio:

```
1 30
2 12
3 18
4 Esame superato.
```

## Esercizio 3

Scrivere un programma che implementi la funzione segno: deve stampare -1 se il numero digitato dall'utente è minore di 0, 0 se uguale a 0, 1 se maggiore di 0.

Esempio:

```
1 Inserisci un numero: 9
```

```
2 1
```

```
1 Inserisci un numero: 0
```

```
2 0
```

```
1 Inserisci un numero: -20
```

```
2 -1
```

## Esercizio 4

Scrivere un programma che, dati due interi in input, scriva qual è il massimo tra i due (usando l'operatore ternario).

## Esercizio 5

Scrivere un programma che verifichi se un anno inserito dall'utente sia bisestile oppure no. Un anno è bisestile se è divisibile per 4; tuttavia se è divisibile per 100 non è bisestile, a meno che non sia anche divisibile per 400.

Esempio:

```
1 Inserisci un anno: 2009
2 L'anno 2009 non e' bisestile.
```

```
1 Inserisci un anno: 2020
2 L'anno 2020 e' bisestile.
```



## Esercizio 6

Scrivere un programma che legga un carattere inserito dall'utente e lo classifichi in carattere alfabetico (a-z, A-Z), cifra (0-9), oppure carattere speciale.

Esempio:

```
1 c
2 carattere alfabetico

1 5
2 cifra

1 !
2 carattere speciale
```

## Esercizio 7

Scrivere una semplice calcolatrice che prenda in input dall'utente, usando lo switch: numero operatore numero e ne stampi il risultato.

Esempio:

```
1 5 * 4
2 20
```

Una volta implementata l'esercizio, lo si estenda aggiungendo un else per gestire il caso dell'operatore non riconosciuto e aggiungendo un controllo per evitare la divisione per 0.

## Esercizio 8

Scrivere un programma che, data una serie di numeri immessi dall'utente, ne trovi il maggiore. Il programma deve chiedere all'utente di immettere i numeri uno alla volta. Quando l'utente immette lo zero, il programma deve visualizzare il più grande numero immesso fino a quel momento.

## Esercizio 9

Scrivere un programma che, dati due interi, dica se il primo è divisibile per il secondo.

## Esercizio 10

Scrivere un programma che chieda un orario nel formato a 24 ore e lo converta in orario in formato 12 ore.

## Esercizio 11

Scrivere un programma che trovi il minimo e il massimo tra quattro numeri inseriti dall'utente.

## Esercizio 12

Scrivere un programma che richiede all'utente tre numeri interi che rappresentano una data(giorno, mese, anno) e verifica se la data inserita è valida.

- ▶ Anno: 4 cifre.
- ▶ Mese: valore compreso tra 1 e 12
- ▶ Giorno: compreso tra 1 e ... 29 o 28 se il mese è Febbraio e l'anno è/non è bisestile
- ▶ Un anno è bisestile se è divisibile per 4 e non è divisibile per 100, oppure è divisibile per 400.
- ▶ 31 se il mese è Gennaio, Marzo, Maggio, Luglio, Agosto, Ottobre, Dicembre
- ▶ 30 se il mese è Aprile, Giugno, Settembre, Novembre.

## Esercizio 13

Scrivere un programma che, date due date dall'utente, indichi quale delle due sia prima in ordine cronologico.



## Esercizio 14

Scrivere un programma che chieda all'utente un numero a due cifre e successivamente scriva la dicitura inglese per quel numero.

Esempio:

```
1 59
2 fifty nine
```

## Esercizio 15

Scrivere un programma che acquisisce tre numeri. Il programma verifica se i tre numeri possono rappresentare le dimensioni dei lati di un triangolo: devono essere valori positivi e la somma di due numeri deve essere maggiore del terzo. In caso il controllo fallisca, il programma deve stampare un apposito messaggio di errore. Se il controllo è stato superato con successo, il programma stabilisce che tipo di triangolo è (isoscele, equilatero o scaleno) e stampa un apposito messaggio a video.

## Esercizio 16

Su una scacchiera 8x8 sono posizionati due pezzi: il Re bianco e la Regina nera. Si scriva un programma che acquisisca le posizioni del Re e della Regina in termini di coordinate (x,y), assumendo che la posizione (1,1) sia situata in basso a sinistra rispetto al giocatore. Il programma controlla prima che le coordinate inserite siano valide; in particolare entrambi i pezzi devono trovarsi all'interno della scacchiera e non possono trovarsi nella stessa posizione. Quindi, il programma determina se la Regina è in posizione tale da poter mangiare il Re e visualizza un apposito messaggio specificando anche in che direzione e per quante posizioni deve muoversi per mangiare.