



# Costrutti iterativo e condizionale del C

Alessandra Giordani

[agiordani@disi.unitn.it](mailto:agiordani@disi.unitn.it)

Lunedì 16 maggio 2010

<http://disi.unitn.it/~agiordani/>

# Il costrutto if

## Sintassi

```
if (<condizione>) { /* blocco_if ... */ }  
|[ else { /* blocco_else ... */ } ]
```

- Esegui `blocco_if` solo se `<condizione> != 0`
- Se presente, esegui `blocco_else` solo se `<condizione> == 0`

# Il ciclo while

## Sintassi

```
while (<condizione>) { /* istruzioni */ }
```

- 1 Se `<condizione> != 0`, vai a (2)
  - 2 Esegui il blocco `istruzioni`
  - 3 Se `<condizione> != 0`, vai a (2)
- Il blocco viene ripetuto finché la condizione é vera
  - Il blocco non viene mai eseguito se la condizione é falsa quando viene valutata la prima volta
  - Se l'esecuzione delle istruzioni non modifica il valore dalla condizione il programma continua a ripetere le istruzioni all'infinito ⇒ **infinite loop!**

# Esempio: somma dei primi n numeri

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int n, somma = 0;

    printf("Introdurre un intero positivo:\n");
    scanf("%d", &n);
    printf("Ecco la somma dei primi %d numeri: ", n);
    while (n)
    {
        somma+=n;
        n--;
    }
    printf("%d\n", somma);
    exit(0);
}
```



# Esempio: fattoriale di n

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int n;
    unsigned long fatt;

    printf("Introdurre un intero positivo:\n");
    scanf("%d", &n);
    fatt = 1;
    printf("%d! = ", n);
    while (n > 1)
    {
        fatt = fatt * n;
        n--;
    }
    printf("%ld\n", fatt);
    exit(0);
}
```



## Nota:

- Abbiamo dato per scontato che l'utente immettesse correttamente un numero positivo!
- Sarebbe meglio controllare usando un **if**

```
if (n<1) printf("Errore\n");  
    else {...}
```

# Il ciclo do..while

## Sintassi

```
do { /* istruzioni */ } while (<condizione>)
```

- 1 Esegui il blocco `istruzioni`
- 2 Se `<condizione> != 0`, vai a (1)

Analogo a `while`, però:

- Il blocco viene ripetuto finché la condizione é vera
- Il blocco viene comunque eseguito almeno una volta

# Esempio: media tra n numeri

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int n, somma=0, x, i=1;

    printf("Di quanti numeri interi devo calcolare la media? ")
    scanf("%d", &n);
    if (n<1)
        printf("Errore\n");
    else {
        do {
            printf("Dammi il %d° numero: ",i);
            scanf("%d", &x);
            somma=somma+x;
            i++;
        }
        while (i<=n);
        printf("Ecco la media %f\n", (float)somma/n);
    }
    exit(0);
}
```

# Il ciclo for

## Sintassi

```
for (<inizializzazione> ; <controllo>; <incremento>) {  
    /* istruzioni */  
}
```

- ① Esegui <inizializzazione>
  - ② Se <controllo> != 0, vai a (3), altrimenti esci
  - ③ Esegui il blocco di istruzioni
  - ④ Esegui <incremento>
  - ⑤ Vai a (2)
- Prima della prima iterazione, esegue <inizializzazione>
  - Prima di ogni iterazione, verifica che <controllo> != 0
  - Dopo ogni iterazione, esegue <incremento>

# Esempi

```
1  /* Stampa i numeri da 1 a 10 */
2
3  void stampa() {
4      int i;
5      for (i=1; i<=10; i++) {
6          printf("%d\n", i);
7      }
8  }
9
10 /* Trova il primo multiplo di k nell'intervallo [a,b].
11   Ritorna "-1" se l'intervallo non contiene un
12   multiplo di k. */
13
14 int multiplo(int k, int a, int b) {
15     int corrente;
16     for (corrente = a; corrente <= b; corrente++) {
17         if (corrente % k == 0) {
18             return corrente;
19         }
20     }
21     return -1;
22 }
```



# Esempio: fattoriale di n

```
#include <stdio.h>
#include <stdlib.h>

/* Calcola il fattoriale di n */

main()
{
    int n;
    unsigned long fatt;

    printf("Introdurre un intero positivo:\n");
    scanf("%d", &n);
    if (n<1)
        printf("Errore\n");
    else {
        printf("%d! = ", n);
        for (fatt = 1; n > 1; n--)
            fatt = fatt * n;
        printf("%ld\n", fatt);
    }
    exit(0);
}
```

# break

- L'istruzione `break` all'interno di un ciclo causa l'immediata interruzione del loop

```
1 // Continua a comporre un numero telefonico finché non
2 // trovi libero
3 while (1) {
4     componi_numero_telefonico();
5     if (occupato()) {
6         aspetta_1_minuto();
7     } else { break; }
8 }
```

- Equivalente (ma non identico) a:

```
1 int ancora = 1;
2 while (ancora) {
3     componi_numero_telefonico();
4     if (occupato()) {
5         aspetta_1_minuto();
6     } else { ancora = 0; }
7 }
```

# continue

- L'istruzione `continue` all'interno di un ciclo causa l'esecuzione della prossima iterazione, ma prima
  - ciclo `for`: valuta incremento, valuta condizione
  - ciclo `while/do...while`: valuta condizione

```
1 // Fai qualcosa per tutti i numeri compresi tra 100 e 1000.
2 // Se il numero é primo, fai anche qualche altra cosa.
3 k=1001;
4 while (--k >= 100) {
5     fai_qualcosa(k);
6     if (!numero_primo(k)) { continue; }
7     fai_qualche_altra_cosa(k);
8 }
```

- Equivalentemente:

```
1 k=1001;
2 while (--k >= 100) {
3     fai_qualcosa(k);
4     if (numero_primo(k)) { fai_qualche_altra_cosa(k); }
5 }
```



# Considerazioni

- Permettono di definire eccezioni all'esecuzione di un loop
- In caso di annidamento, si riferiscono sempre al loop di maggior profondità
- Rendono piú **opaca** la semantica del loop
- Possono complicare la lettura e il debugging del codice
- Si può vivere senza: la stessa semantica può essere realizzata modificando la struttura dei loop:
  - `if (x) {continue;} y;`  $\Rightarrow$  `if (!x) {y;}`
  - `if (x) {break;} y;`  $\Rightarrow$  `if (x) {cond;} else {y;}`  
dove `cond` é qualche istruzione che modifica il valore di verità del controllo del loop
- Alcuni linguaggi di programmazione ne fanno a meno
- Alcuni programmatori li evitano per principio
- Sono comodi e fanno risparmiare tempo se usati correttamente



# exit VS return

## ■ return

- È un'istruzione (restituire un valore di funz.)

- Anche il main è una funzione

```
int main() { return 0; }
```

## ■ exit()

- È una funzione (della libreria standard)

- Si può usare solo nel main

```
#include <stdlib.h>
```

```
int main() { exit(0); }
```



# Il costrutto switch

- Leggere n numeri e trovare la somma o la media a seconda della scelta dell'utente
- In pratica si può usare il costrutto if, ma in caso di scelte multiple è meglio usare lo switch

# Codice da aggiungere:

```
while (1) {
getchar(); //consuma il carattere "a capo" non letto dalla scanf di prima
printf("Vuoi sapere la [s]omma, la [m]edia o [u]scire? ");
scelta=getchar();
switch(scelta)
{
    case 's':
    case 'S': {
        printf("Ecco la somma %d\n", somma);
        break;
    }
    case 'm':
    case 'M': {
        printf("Ecco la media %f\n", (float)somma/n);
        break;
    }
    case 'u':
    case 'U': exit (0); //con break; uscirebbe dallo switch non dal while!
    default : printf("carattere non accettato!\n");
}
}
```