

LE STRUTTURE DI CONTROLLO IN C

while (*esp*) {*corpo del ciclo*}

Viene verificato che *esp* sia vera, nel qual caso è eseguito *corpo del ciclo*.
Il ciclo si ripete fintantoché *esp* risulta essere vera.

```
/* Programma SommaSequenza */
#include <stdio.h>
main()
{
    int numero, somma;
    somma = 0;
    scanf("%d", &numero);
    while (numero != 0)
    {
        somma = somma + numero;
        scanf("%d", &numero);
    }
    printf("\n somma dei numeri letti %d", somma);
}
```

LE STRUTTURE DI CONTROLLO IN C

Il ciclo enumerativo (o “a conteggio”)

for (*espressione1* ; *espressione2* ; *espressione3*) {*corpo del ciclo*}

Prevede di eseguire il corpo del ciclo un numero di volte determinato da:

- ✓ il **valore iniziale** di una variabile, che funge da **contatore**;
- ✓ la **condizione** che la variabile contatore deve soddisfare;
- ✓ l'**incremento** (o **passo**) da applicare alla variabile contatore al termine di ogni iterazione.
Può essere negativo.

ove

espressione1 assegna al contatore il valore iniziale;

espressione2 esprime la condizione che il contatore deve verificare perché corpo del ciclo venga eseguito;

espressione3 applica al contatore l'incremento; è l'ultima istruzione eseguita dopo il corpo del ciclo.

In C ogni **while** può essere sostituito da un **for**.

Il ciclo a condizione finale

Prevede di eseguire il corpo del ciclo finché è valida la condizione. Quando questa non lo è più, il ciclo viene terminato.

```
do
    {corpo del ciclo}
while (condizione);
```

Il corpo del ciclo viene comunque eseguito almeno una volta.

LE STRUTTURE DI CONTROLLO IN C

Istruzione di selezione a più vie **switch-case**

```
switch (espressione) { case valore1 : case valorek :      sequenza1;  
                        break;  
                        case valoren : case valorem :      sequenza2;  
                        break;  
                        . . . . .  
                        default :                          sequenza;  
                    }
```

L'*espressione* deve fornire un risultato di tipo *Integral Type*.

Le costanti *valorei* rappresentano i possibili valori normalmente assunti da *espressione*.

Quando *espressione* non assume uno dei valori anzidetti, viene eseguita la *sequenza* in corrispondenza della parola-chiave *default*.

Il costrutto *break* porta ad eseguire la istruzione dopo il blocco.

Esempi

```
switch      (CarattereLetto)
{
    case 'A': case 'G': case 'H': printf ("Il carattere letto è A o G o H\n");
                                break;
    case 'F':      printf ("Il carattere letto è F\n");
                                break;
    default:      printf ("il carattere letto è sbagliato\n");
                                break;
}
```

```
switch      (Dipendente.Qualifica)
{
    case      CapoProgetto:  Dipendente.Retribuzione =
                                (Dipendente.Retribuzione * 11) / 10;
                                break;
    case      Venditore:     Dipendente.Retribuzione =
                                (Dipendente.Retribuzione * 12) / 10;
                                break;
    case      Segretario:    Dipendente.Retribuzione =
                                (Dipendente.Retribuzione * 12) / 10;
                                break;
}
```

LE INTERRUZIONI NELLE STRUTTURE DI CONTROLLO

L'istruzione **break** oltre a interrompere l'esecuzione del **case** (cfr. istruzione **switch**), provocando un salto del flusso di esecuzione alla prima istruzione successiva, può forzare la terminazione di un'iterazione **for**, **while** o **do-while**

L'istruzione **continue** come **break** impedisce l'esecuzione delle istruzioni successive di un ciclo (**while**, **for** o **do-while**) e di passare all'iterazione successiva a partire dall'inizio.

L'istruzione **exit** della libreria standard `stdlib.h` provoca l'immediata terminazione del programma e il ritorno al sistema operativo.

L'istruzione **goto** produce l'effetto di trasferire incondizionatamente il flusso di controllo ad una istruzione.

goto *intero positivo;*

L'etichetta *intero positivo:* identifica l'istruzione a cui si vuole saltare.

Esempio

```
        goto    1000;
.....
1000:   istruzione
```

Le ragioni della programmazione strutturata, tra cui leggibilità ed semplicità del codice, **sconsigliano l'uso generalizzato di** **break**, **continue** ed **exit**, e **"proibiscono"** quello di **goto**.

LE INTERRUZIONI NELLE STRUTTURE DI CONTROLLO

Esempi di uso delle istruzioni **break** e **continue**

```
/* ciclo infinito per leggere reali positivi e calcolare sqrt */
    while (true)
    {
        scanf ("%f", &reale);
        if (reale < 0.)    break;
        printf ("%f \n", sqrt (reale));
    }
/* se reale è negativo il break fa saltare a dopo il blocco */
.....
```

```
/* saltare tutte le cifre lette in una sequenza di lunghezza L */
j = 0;
for (i = 0; i < L; i++)
{
    scanf ("%c", &car);
    while (car == '\\n' || car == '\\r') scanf ("%c", &car);
    if (car >= 0 && car <= 9) continue;
    stringa[j] = car;
    j = j + 1;
}
/* il continue fa passare il controllo qui. Viene eseguito l'incremento
dell'indice i del ciclo */
}
```

Esempi di uso delle istruzioni **break** e **continue**

```
/* Codice che elabora tutti i caratteri eccetto le lettere  
minuscole e maiuscole */
```

```
for (i = 0; i < NumDati; i++)  
{  
    scanf ("%c", &Dato);  
    if ((Dato >= 'A' && Dato <= 'Z') || (Dato >= 'a' && Dato <= 'z'))  
        continue;
```

```
/* Istruzioni che elaborano gli altri caratteri */  
    ...
```

```
/* continue trasferisce qui il controllo perché possa iniziare la prossima  
iterazione del ciclo. È importante notare che i++ viene eseguita anche in  
questo caso */  
}
```