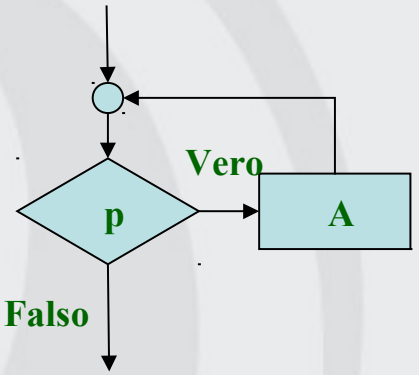
I cicli

Iterazioni
Calcolo della media
Istruzioni break e continue

Ciclo while




```
while(<espressione-booleana>)
    <istruzione>;
```

Esegue una istruzione mentre una condizione è verificata

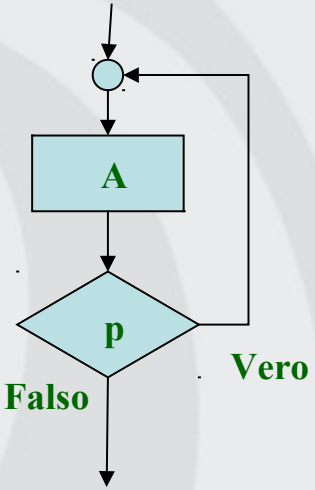
Ciclo while

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv) {
4.    int i=1;
5.    float s=0.0F;
6.
7.    while(i<1000) {
8.      s += 1.0F / (i*i*i); // equivale a: s = s + 1.0F/();
9.      i++;                 // equivale a: i = i + 1;
10.   }
11.   printf("La somma e' %f\n", s);
12. }
```

Il blocco potrebbe non essere mai eseguito

```
La somma e' 1.202051
```

Ciclo do-while




```
do
  <istruzione>
while(<espressione-booleana>);
```

Esegue una istruzione finché una condizione diventa falsa

Ciclo do-while

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv) {
4.    int i=1;
5.    float s=0.0F;
6.
7.    do {
8.      s += 1.0F / (i*i*i);
9.      i++;
10.   } while(i<1000);
11.   printf("La somma e' %f\n", s);
12. }
```

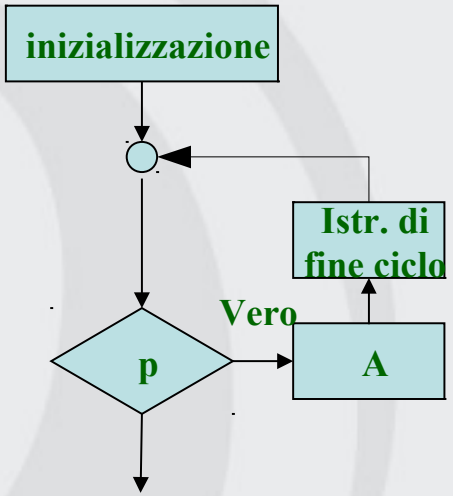
Il blocco viene eseguito sempre almeno una volta

```
La somma e' 1.202051
```

Si poteva anche scrivere in forma compatta

```
do s += 1.0F / (i*i*i); while(++i<1000);
```

Ciclo for




```
for(<espr>; <espr-booleana>; <espr>)
    <istruzione>;
```

Implementa una struttura while con la possibilità di inizializzare dei valori

Ciclo for

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv) {
4.    int i;
5.    float s=0.0F;
6.
7.    for(i=1; i<1000; i++)
8.      s += 1.0F / (i*i*i);
9.
10.   printf("La somma e' %f\n", s);
11. }
```

```
La somma e' 1.202051
```

Ciclo for

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv) {
4.    int i;
5.    float s=0.0F;
6.
7.    for(i=999; i>0; i--)
8.      s += 1.0F / (i*i*i);
9.
10.   printf("La somma e' %f\n", s);
11. }
```

Attenzione il risultato dipende dall'ordine delle operazioni

```
La somma e' 1.202056
```

Ciclo for

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv) {
4.    int i=1;
5.    float s=0.0F;
6.
7.    for(i=1; i<1000; i++)
8.      s += 1 / (i*i*i);
9.
10.   printf("La somma e' %f\n", s);
11. }
```

```
La somma e' 1.000000
```

Attenzione ai problemi con la divisione

Ciclo controllato da un contatore

Media dei voti:

- **Descrizione del problema:**
  - Una classe di 10 studenti affronta un quiz. I risultati sono interi fra 0 e 100. Calcolare la media complessiva.
  - I voti sono generati in maniera casuale.

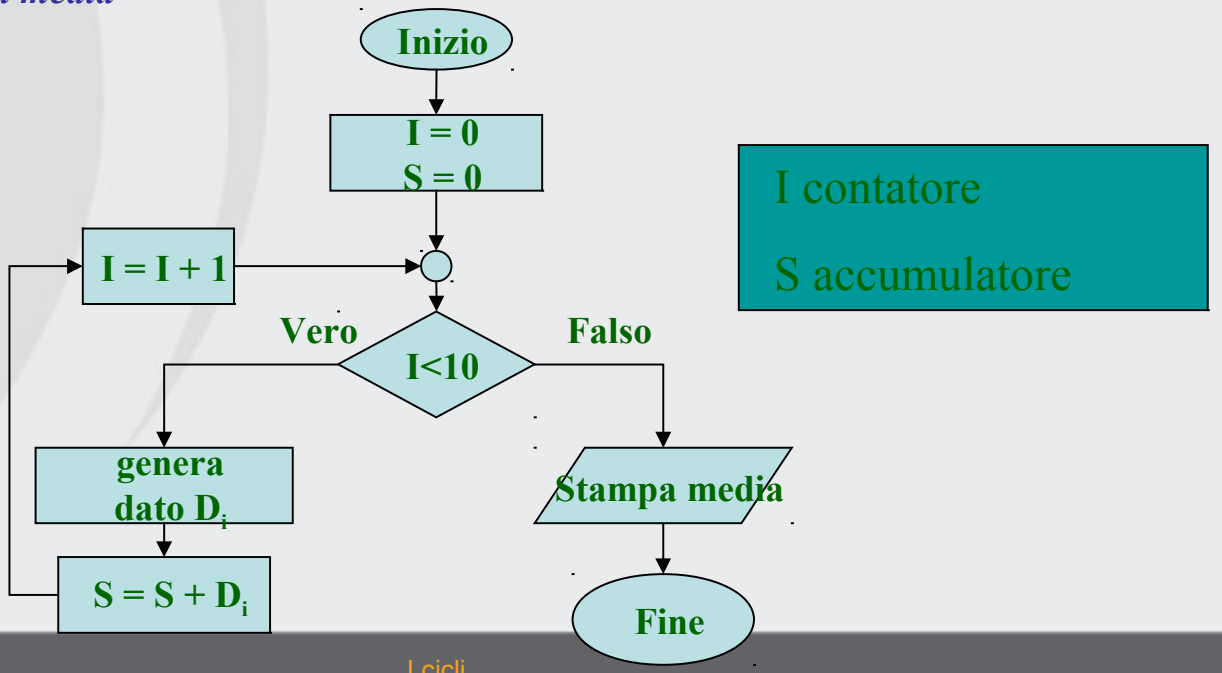
È una iterazione controllata da un contatore

- **Il ciclo viene ripetuto finché un contatore non raggiunge un determinato valore**
- **Il numero di iterazioni è noto: si usa un contatore**

Ciclo controllato da un contatore

Descrizione a parole:

1. *azzera* `totale` *(somma parziale - accumulatore)*
2. *azzera il contatore dei numeri già introdotti (*`contatore`*)*
3. *genera un dato (a caso)*
4. *somma il dato a* `totale`
5. *incrementa il contatore*
6. *se il numero di dati trattati (contatore) è minore di 10 torna a 3, altrimenti continua*
7. *stampa la media*



CalcoloMedia.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>  // vi è la definizione di random()
3.
4.  int main(int argc, char **argv)
5.  {
6.     // inizializzazione del programma
7.     int contatore=0, totale=0;
8.     int numero_studenti = 10;
9.
10.    // esecuzione del programma
11.    while(contatore<numero_studenti) {
12.       // random restituisce un intero a caso
13.       // (generatore di numeri pseudo casuali)
14.       totale += random() % 101; // il risultato è un numero fra 0 e 100 inclusi
15.       contatore++;
16.    }
17.    // presentazione dei risultati
18.    int media = totale / contatore;
19.    printf("La media è: %d\n", media);
20.    return 0;
21. }
```

```
La media è 46
```

Iterazione controllata da sentinella

Generalizzazione del problema

- Scrivere un programma per un numero arbitrario di studenti
- Se il numero degli studenti è sconosciuto come fare??

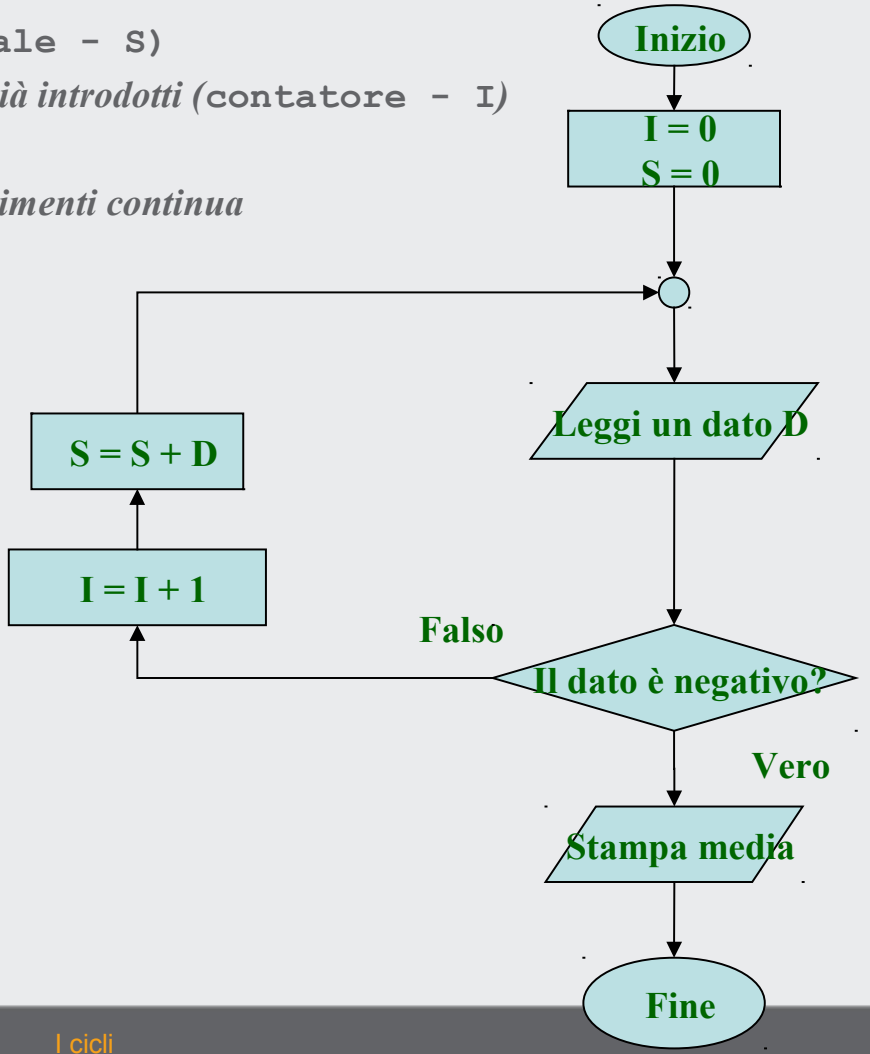
Soluzione uso della sentinella:

- Un valore che indica “la fine dei dati”
- Il ciclo termina quando viene immesso il valore sentinella
- Deve essere un valore non legale
  - Per esempio –1 in questo caso

Ciclo controllato da una “sentinella”

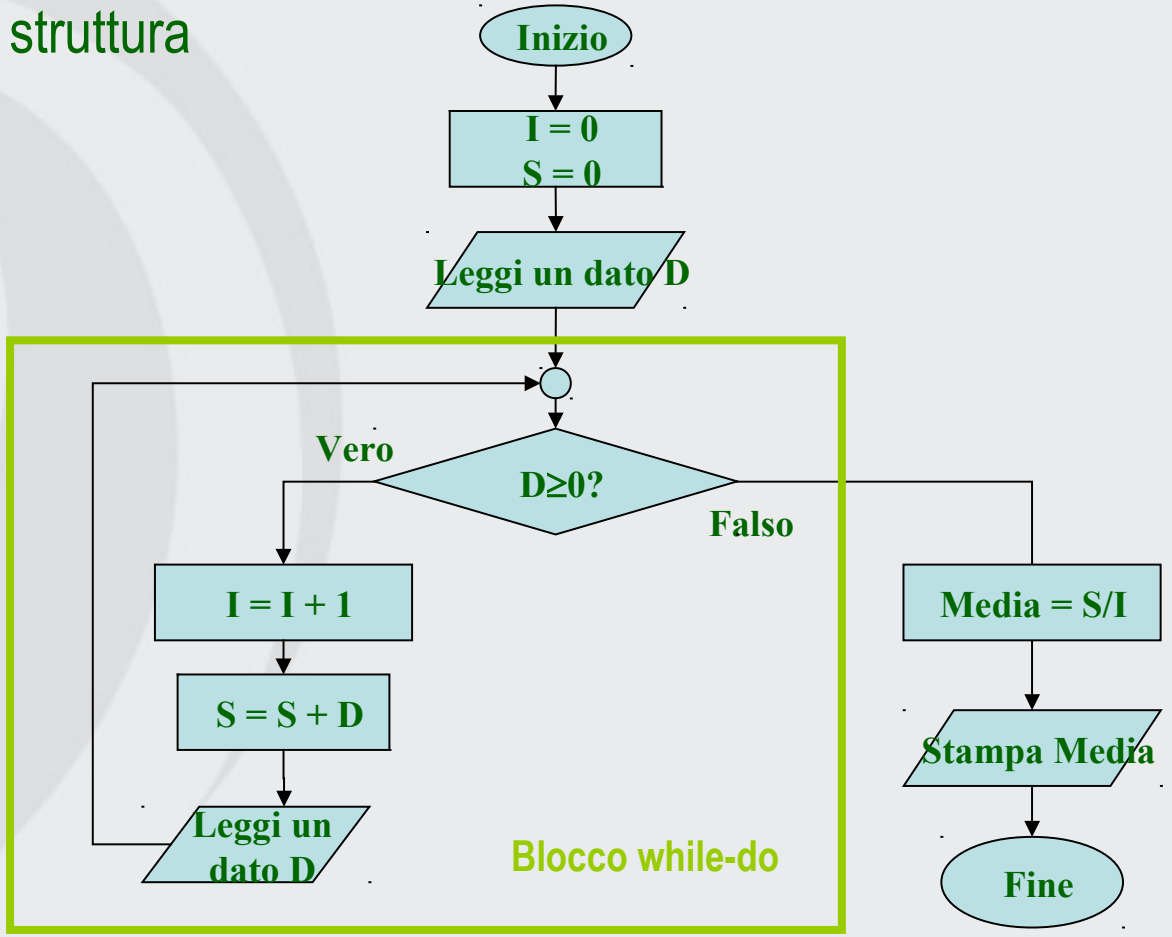
Descrizione a parole:

1. *azzera la somma parziale (`totale - S`)*
2. *azzera il contatore dei numeri già introdotti (`contatore - I`)*
3. *leggi un dato*
4. *se il dato è negativo vai a 8, altrimenti continua*
5. *somma il dato a `totale`*
6. *incrementa il contatore*
7. *torna a 3*
8. *stampa la media*



Ciclo controllato da una “sentinella”

Versione struttura



sentinella.c

```
1.  #include <stdio.h>
2.
3.  int main(int argc, char **argv)
4.  {
5.     int contatore=0;
6.     int totale=0;
7.     printf("Introduci un voto (-1 per terminare): ");
8.     int voto;
9.     scanf("%d", &voto);
10.    while(voto>=0) {
11.       totale = totale + voto;  // somma il voto al totale
12.       contatore = contatore + 1;  // incrementa il contatore
13.       printf("Introduci un voto (-1 per terminare): ");
14.       scanf("%d", &voto);
15.    }
16.    if(contatore>0) {
17.       printf("Sono stati introdotti %d dati", contatore);
18.       double media = (double) totale / contatore; // divisione fra double
19.       printf("La media dei voti è %f\n", media);
20.    } else {
21.        printf("Non sono stati introdotti dati");
22.    }
23.    return 0;
24. }
```

```
Introduci un voto (-1 per terminare): 56
Introduci un voto (-1 per terminare): 72
Introduci un voto (-1 per terminare): 87
Introduci un voto (-1 per terminare): -1
Sono stati introdotti 3 dati
La media dei voti è 71.666667
```

Iterazione controllata da sentinella

– La maggior parte del codice è uguale al precedente


```
9.      scanf("%d", &voto);
10.    while ( voto>=0 ) {
11.       totale = totale + voto;      // somma il voto al totale
12.       contatore = contatore + 1;  // incrementa il contatore
13.       printf("Introduci un voto (-1 per terminare): ");
14.       scanf("%d", &voto);
15.    }
```

Iterazione controllata da sentinella

```
18.    double media = (double) totale / contatore; // divisione fra double
```

La variabile è diventata di tipo double

Così da memorizzare numeri decimali

Le operazioni aritmetiche sono eseguite sempre tra valori dello stesso tipo

In caso di necessità si esegue una promozione automatica verso il tipo che garantisce la maggiore precisione

l'operatore cast `(<tipo>)` forza la conversione di tipo

`(double)` crea una copia di tipo `double` del valore intero in `totale`

`totale` è forzato a double perciò anche `contatore` prima di eseguire la divisione viene promosso a double

La successione di Fibonacci

LA SUCCESSIONE DÌ FIBONACCI

n° di coppie

Il primo mese c'è solo una coppia di conigli, il secondo mese ce ne sono 2 di cui una fertile, quindi il terzo ce ne sono 3 di cui 2 fertili, quindi il quarto mese ce ne sono 5 di cui 3 fertili e così via ...


1


1


2


3


5

I primi numeri di Fibonacci, includendo lo 0, sono: $0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$

Numeri di fibonacci

```
1.  printf("Quanti valori vuoi calcolare? ");
2.  int n, i;
3.  scanf("%d", &n);
4.  printf("0\n1\n");
5.  int prec2 = 0, prec1 = 1;
6.  for (i = 2; i < n; i++) {
7.     int val = prec2 + prec1;
8.     prec2 = prec1;
9.     prec1 = val;
10.    printf("%d\n", val);
11. }
```

I primi 2 valori sono 0 e 1

Devo ricordarmi i due valori precedenti

```
Quanti valori vuoi calcolare? 5
0
1
1
2
3
```

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Soluzione alternativa

Problema

Dato un numero stampare il mese corrispondente:

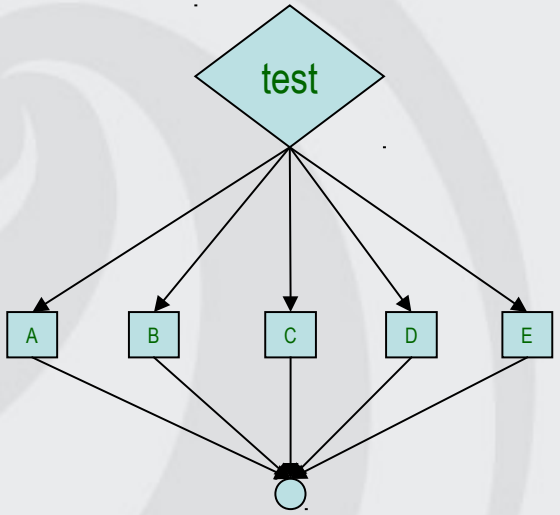
- **1 → gennaio**
- **2 → febbraio**
- **…**
- **12 → dicembre**

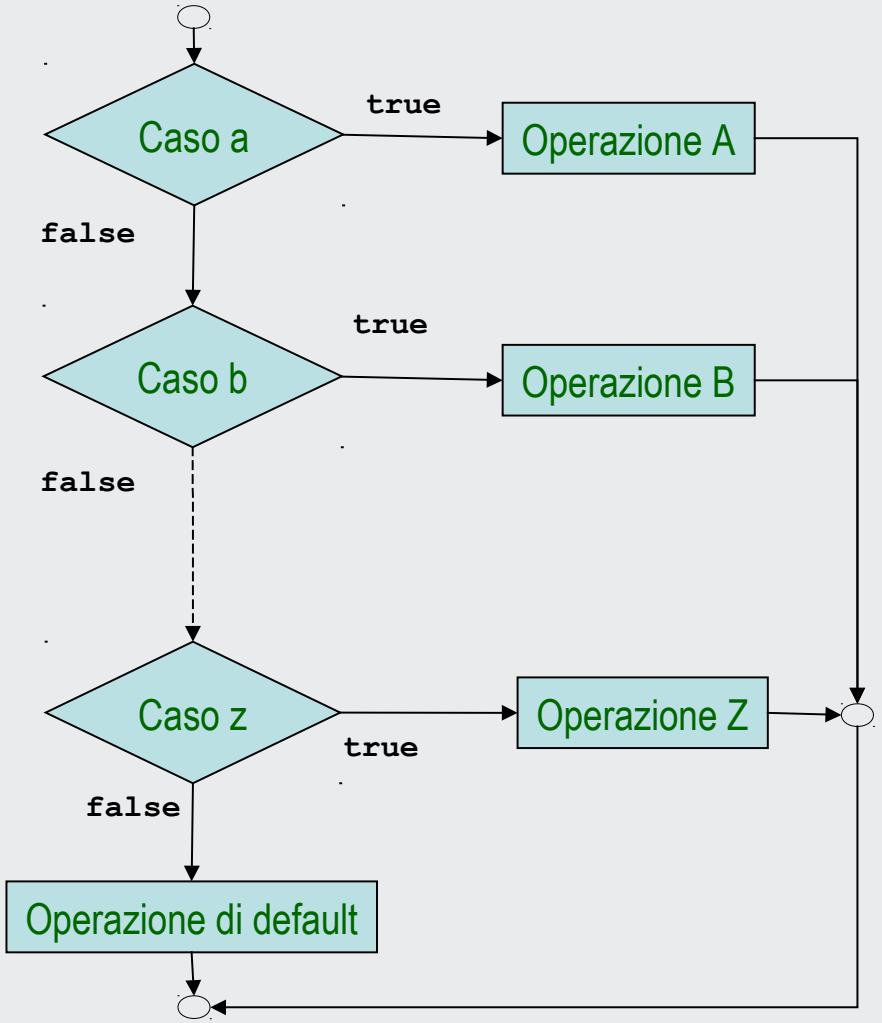
Codifica dei mesi

```
1.  printf("Quale mese vuoi stampare? :");
2.  scanf("%d", &mese);
3.  if(mese == 1) {
4.      printf("Il mese e' gennaio\n");
5.  } else if (mese == 2) {
6.      printf("Il mese e' febbraio\n");
7.  } else if (mese == 3) {
8.      printf("Il mese e' marzo\n");
9.  } else if (mese == 4) {
10.     printf("Il mese e' aprile\n");
11. } else if (mese == 5) {
12.     printf("Il mese e' maggio\n");
13. } else if (mese == 6) {
14.     printf("Il mese e' giugno\n");
15. . . .
16. } else if (mese == 12) {
17.     printf("Il mese e' dicembre\n");
18. } else {
19.     printf("Codice non corretto\n");
20. }
```

```
Quale mese vuoi stampare? 7
Il mese e' luglio
```

Selezione multipla





Istruzione `switch`

Il risultato dell'espressione permette di selezionare la sequenza di istruzioni da eseguire introdotta da una etichetta `case` con lo stesso valore.
Se tale valore non è presente si esegue (se presente) la sequenza introdotta da `default`

La sequenza ha termine quando si incontra l'istruzione `break`

```
switch(espr) {
   case const1:
      istr1;
      break;
   case const2:
      istr2; // viene eseguita anche istr3
   case const3:
      istr3;
      break;
     ....
   default:
      istr;
      break;
}
```

Codifica dei mesi

```
1.      switch(mese) {
2.         case 1:
3.           printf("Il mese e' gennaio\n");   break;
4.         case 2:
5.           printf("Il mese e' febbraio\n");  break;
6.         case 3:
7.           printf("Il mese e' marzo\n");     break;
8.         case 4:
9.           printf("Il mese e' aprile\n");    break;
10.        . . . .
11.        case 11:
12.          printf("Il mese e' novembre\n");  break;
13.        case 12:
14.          printf("Il mese e' dicembre\n");  break;
15.        default:
16.          printf("Codice non corretto\n");
17.     }
```

```
Quale mese vuoi stampare? 7
Il mese e' luglio
```

```
Quale mese vuoi stampare? 0
Codice non corretto
```

Codifica dei mesi

```
1.      switch(mese) {
2.         case 1:
3.           printf("Il mese e' gennaio\n");
4.         case 2:
5.           printf("Il mese e' febbraio\n");
6.         case 3:
7.           printf("Il mese e' marzo\n");
8.         case 4:
9.           printf("Il mese e' aprile\n");
10.        . . . .
11.        case 11:
12.          printf("Il mese e' novembre\n");
13.        case 12:
14.          printf("Il mese e' dicembre\n");
15.        default:
16.          printf("Codice non corretto\n");
17.     }
```

```
Quale mese vuoi stampare? 7
Il mese e' luglio
Il mese e' agosto
Il mese e' settembre
Il mese e' ottobre
Il mese e' novembre
Il mese e' dicembre
Codice non corretto
```

Codifica dei mesi

```
1.  if(0 < mese < 13) {
2.     switch(mese) {
3.        case 1:
4.          printf("Il mese e' gennaio\n");
5.        . . . .
6.        case 12:
7.          printf("Il mese e' dicembre\n");
8.     }
9.  } else {
10.    printf("Codice non corretto\n");
11. }
```

0 < mese < 13 è una espressione sintatticamente corretta sempre vera:

- 0 < mese dà come risultato o 0 o 1 (comunque minore di 13)

Codifica dei mesi

```
1.  if(0 < mese && mese < 13) { //versione corretta
2.     switch(mese) {
3.        case 1:
4.          printf("Il mese e' gennaio\n");
5.        . . . .
6.        case 12:
7.          printf("Il mese e' dicembre\n");
8.     }
9.  } else {
10.    printf("Codice non corretto\n");
11. }
```

Istruzioni `break` e `continue`

`break`

- Forza l'uscita immediata dalle istruzioni **`while`**, **`for`**, **`do/while`** o **`switch`**
- Il programma prosegue con la prima istruzione dopo la struttura corrente
- Usi più comuni di `break`
  - Uscita da un ciclo
  - Saltare il resto della istruzione **`switch`**

`continue`

- Salta il resto del corpo delle istruzioni **`while`**, **`for`** o `do/while`
  - Si continua con l'iterazione successiva del ciclo
- Istruzioni **`while`** e **`do/while`**
  - Il test di ciclo viene valutato immediatamente dopo l'istruzione `continue`
- Istruzione **`for`**
  - Viene eseguita l'istruzione di fine ciclo e poi eseguito il test

Fattoriale

```
1.   #include <stdio.h>
2.
3.   int main()
4.   {
5.       int i, n, ris;
6.       printf("Quanti valori vuoi calcolare: ");
7.       scanf("%d", &n);
8.       printf("Il fattoriale di %d e' %d\n", 0, 1);
9.       for(ris=1, i=1; i<=n; i++) {
10.          ris *= i;
11.          printf("Il fattoriale di %d e' %d\n", i, ris);
12.      }
13.      return 0;
14.  }
```

Attenzione devo inizializzare il risultato a 1

```
Quanti valori vuoi calcolare: 13
Il fattoriale di 0 e' 1
Il fattoriale di 1 e' 1
Il fattoriale di 2 e' 2
Il fattoriale di 3 e' 6
Il fattoriale di 4 e' 24
Il fattoriale di 5 e' 120
Il fattoriale di 6 e' 720
Il fattoriale di 7 e' 5040
Il fattoriale di 8 e' 40320
Il fattoriale di 9 e' 362880
Il fattoriale di 10 e' 3628800
Il fattoriale di 11 e' 39916800
Il fattoriale di 12 e' 479001600
Il fattoriale di 13 e' 1932053504
```

Il valore è evidentemente scorretto

Come risolvere il problema?

Fattoriale

```
9.      for(ris=1, i=1; i<=n; i++) {
```

Inizializzo sia i che ris a 1

- in questo caso si poteva anche scrivere ris=i=1

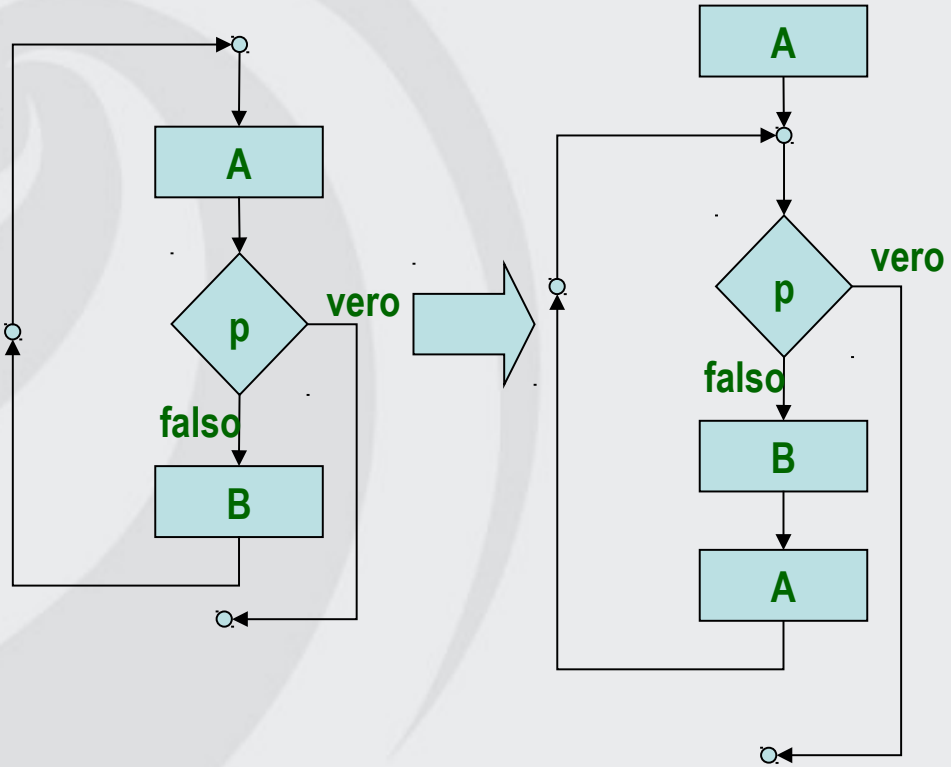
Esiste anche l'operatore ,

- Sintassi: `espr1 , espr2`
- Il risultato è espr2, ma espr1 viene comunque calcolata
- esempio:

```
b = (a=1, 2);
```

- b assume il valore 2, ma a viene inizializzato a 1

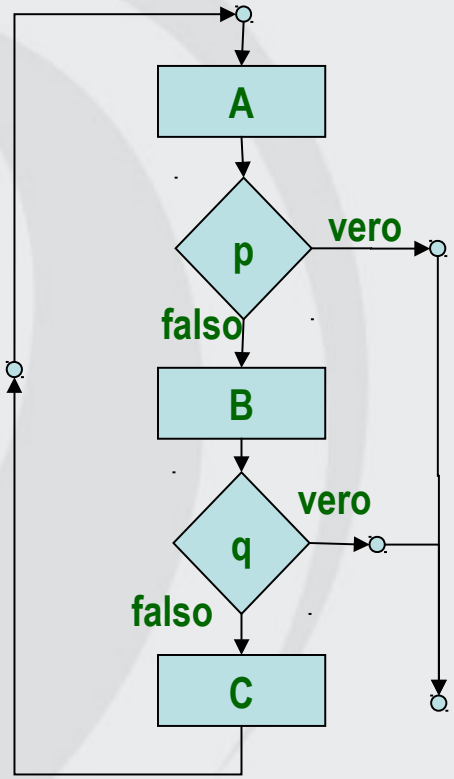
Ω_1




```
1.  while(1) {
2.    A();
3.    if(p) break;
4.    B();
5.  }
```

```
1.  A();
2.  while(!p) {
3.    B();
4.    A();
5.  }
```

Ω_2




```
1.  for(;;) {
2.      A();
3.      if(p) break;
4.      B();
5.      if(q) break;
6.      C();
7.  }
```

Istruzione goto

Sintassi: `goto <etichetta>;`

L'uso è normalmente deprecato, rende I programmi poco leggibili

Si usa a volte in caso di condizioni di errore per forzare l'uscita immediata da più cicli annidati

```
1.  for(;p;) {
2.     C();
3.     for(errore=0;q;) {
4.        A();
5.        if(errore) goto tratta_errore;
6.        B();
7.     }
8.     D();
9.  }
10. tratta_errore:
```

if annidati (massimo di 3 valori)

```
int a, b, c, max;
printf("Scrivi 3 valori: ");
scanf("%d %d %d");
if(a>b) {  // il massimo è a o c
    if(a>c) {
            max = a;
    } else {
            max = c;
    }
} else {  // il massimo è b o c
    if(b>c) {
            max = b;
    } else {
            max = c;
    }
}
```

Come esercizio calcolare contemporaneamente anche il minimo
Il codice è corretto anche se vi sono valori uguali?