Programmazione I

Il Linguaggio C

Strutture di Controllo

Daniel Riccio

Università di Napoli, Federico II

15 ottobre 2021

Sommario

- Argomenti
  - Strutture di controllo iterative
  - Istruzione while
  - Istruzione for
  - Istruzione do-while

Strutture iterative

Si dice **ciclo** (**loop**) una sequenza di istruzioni che deve essere ripetuta più volte consecutivamente.

Si consideri ad esempio il calcolo del fattoriale di un numero **n** > 0:

$$\mathbf{n!} = n \times (n - 1) \times (n - 2) \ldots \times 2$$

con il caso particolare $0! = 1$.

Sembrerebbe che basti una semplice assegnazione, ma se non si conosce a priori il valore di **n**, è impossibile scrivere l'istruzione che esegue il calcolo del fattoriale, poiché la formula contiene tanti fattori quanti ne indica **n**.

Strutture iterative

Proviamo a riscrivere la formula del fattoriale come:

$$\mathbf{n!} = ((\ldots (((1 \times 2) \times 3) \times 4) \ldots \times (n - 1) \times n)$$

Osservando la formula possiamo:

- attribuire ad una variabile **fatt** il valore 1
- moltiplicare **fatt** per **2** ed attribuire il risultato ancora a **fatt**
- poi moltiplicare **fatt** per **3** e così via fino a **n**.

Strutture iterative

L'algoritmo di risoluzione può quindi essere formalizzato mediante un procedimento iterativo:

assegna il valore **1** a **fatt**;
se **n** vale **0**, termina;
con **k** che varia da **1** a **n** con passo unitario esegui:
    moltiplica **fatt** per **k** ed attribuisci il risultato a **fatt**;

Il prodotto **fatt** $\times$ **k** viene eseguito **n** volte com'è necessario.

Strutture iterative

Nel linguaggio C i cicli iterativi sono realizzati da tre costrutti:

while : realizza il costrutto **WHILE - DO**;

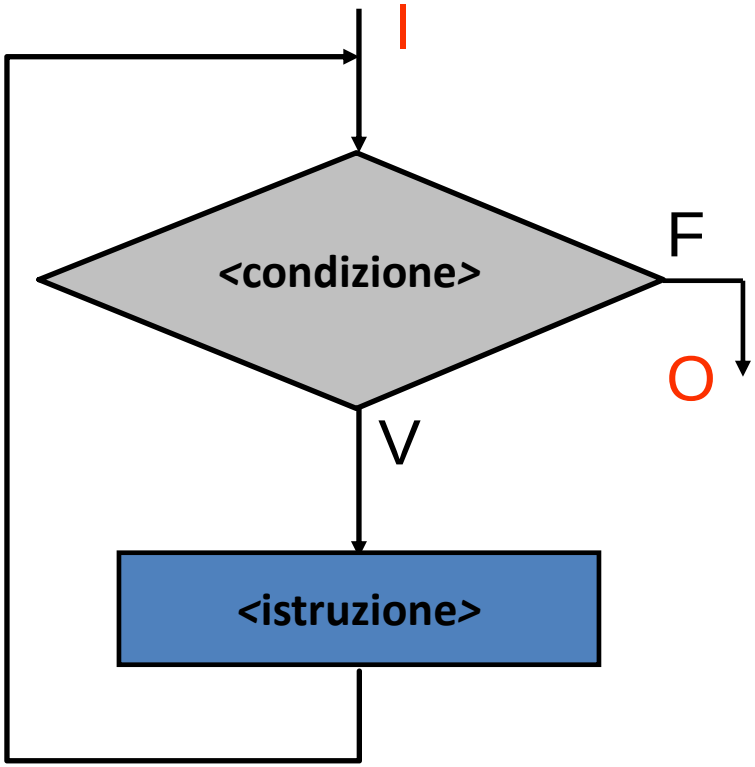
do – while : realizza il costrutto **REPEAT - UNTIL**;

for : realizza il **ciclo a contatore**.

Istruzione while

Sintassi:

while (<**condizione**>)
<**istruzione**>



Finché <**condizione**> è vera esegue <**istruzione**> che può essere semplice o composta.

Istruzione while: osservazioni

Il costrutto **while** realizza il costrutto **while – do** della programmazione strutturata.

<**condizione**> deve essere di tipo logico ed è ricalcolata ad ogni iterazione;

se <**condizione**> risulta falsa già alla prima iterazione <**istruzione**> non viene eseguita neppure una volta;

se <**condizione**> non diventa mai falsa non si esce mai dal loop!

essendo <**istruzione**> una normale istruzione composta può contenere qualsiasi tipo di istruzione, in particolare altri **while**, dando origine (come per l'if) a **while annidati**.

Istruzione while: osservazioni

Siano **a** e **b** due numeri interi i cui valori iniziali sono: **a=5**; **b=10**; (rappresentano le inizializzazioni delle variabili del ciclo necessarie per poter valutare l'espressione booleana; con **b=10** e **a=12** il ciclo non verrebbe eseguito).

Quale sarà il valore di a che verrà stampato alla fine del ciclo?

L'inizializzazione pone a=5 e b=10
Entriamo nel ciclo while, la condizione (a<b) è vera
- otteniamo a=6 e b=9
  Si ritorna alla condizione di controllo del ciclo, (a<b), e poiché 6<9, si esegue un altro ciclo

-a=7 e b=8
  si ritorna ancora al controllo (a <b), si verifica che 7<8

- a=8 e b=7
  la condizione (a <b) è falsa per cui si salta il ciclo e si esegue l'istruzione subito dopo

- Il computer scriverà sul video a=8

```
a=5;
b=10;
while(a<b){
   a++;
   b--;
}
```

passo	a	b
inizio	5	10
N. 1	6	9
N. 2	7	8
N. 3	8	7

Istruzione while: esempi

- inizializza i a 0
- finché i è minore di 0
- stampa Hello <valore di i>
- Incrementa i

Incremento esplicito

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( i < 10 ){
    printf("Hello %d\n", i );
    i = i + 1;
  }
}
```

Incremento +=

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( i < 10 ){
    printf("Hello %d\n", i );
    i += 1;
  }
}
```

Incremento postfisso

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( i < 10 ){
    printf("Hello %d\n", i );
    i++;
  }
}
```

Incremento prefisso

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( i < 10 ){
    printf("Hello %d\n", i );
    ++i;
  }
}
```

Incremento Postfisso simultaneo

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( i++ < 10 )
    printf("Hello %d\n", i );
}
```

Incremento Prefisso simultaneo

```
#include <stdio.h>

int main() {
  int i = 0;

  while ( ++i < 10 )
    printf("Hello %d\n", i );
}
```

Istruzione while: esempio

Leggere un numero intero **N** da tastiera, e calcolare la somma **S** dei primi **N** numeri interi

Ragionamento

Definiamo una variabile **indice** che assuma come valore, di volta in volta, i numeri interi da **1** a **N**

Definiamo una variabile **S** che conterrà le somme parziali:

– **indice** = **1**, **S** = (**0**) + **1** = **1**
– **indice** = **2**, **S** = (**0** + **1**) + **2** = **3**
– **indice** = **3**, **S** = (**0** +**1** +**2**) + **3** = **6**

La condizione di iterazione è **indice** ≤ **N**

I valori che abbiamo posto tra parentesi rappresentano la somma ottenuta al passo precedente;
S = S + indice;
S += indice;

Istruzione while: esempio

Leggere un numero intero **N** da tastiera, e calcolare la somma **S** dei primi **N** numeri interi

Algoritmo

Leggo **N** da tastiera;
Inizializzo l'accumulatore di risultato **S** a **0**;
Inizializza un contatore **indice** a **1**;

finché **indice** <= **N**;
    aggiungi all'accumulatore **S** il valore di **indice**;
    aggiorna **indice** (ovvero incrementalo di **1**);

stampa(valore di **S**).

Istruzione while: esempio

Leggere un numero intero **N** da tastiera, e calcolare la somma **S** dei primi **N** numeri interi

Programma

main.c

```
1  #include <stdio.h>
2
3  main ()
4  {
5    int N, indice = 1, S = 0;
6
7    printf ("Introduci N: ");
8    scanf ("%d", &N);
9
10   while (indice <= N)
11   {
12     S += indice;
13     indice++;
14   }
15
16   printf ("Somma = %d", S);
17 }
```

Esecuzione

```
Introduci N: 8
Somma = 36
```

Domanda: è possibile risolvere il problema senza ricorrere ad un ciclo?

Nota
La somma dei primi N numeri interi rappresenta una serie notevole e può essere calcolata con una formula chiusa:

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

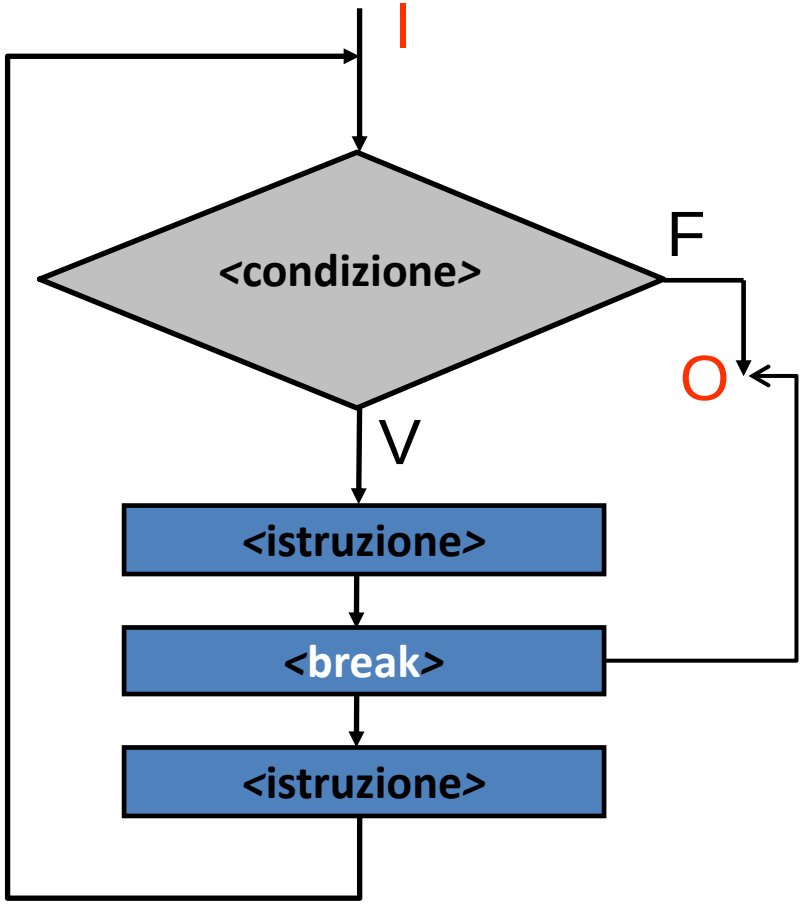
Istruzione while: break e continue

Le strutture di controllo iterative sono corredate dalle istruzioni **break** e **continue**, che consentono rispettivamente di anticipare l'uscita dal ciclo, o di saltare un'iterazione.

break

L'istruzione break nel contesto delle istruzioni di controllo iterative provoca l'immediata terminazione del ciclo e l'uscita dall'ambito di visibilità ad esso connesso.
A seguito di ciò, il controllo di flusso viene quindi rediretto all'istruzione successiva esterna al ciclo.

Solitamente l'esecuzione di break è subordinata alla valutazione di una espressione booleana che determina l'uscita dal ciclo in caso di circostanze "straordinarie" rispetto quelle per cui è prevista la naturale terminazione del ciclo



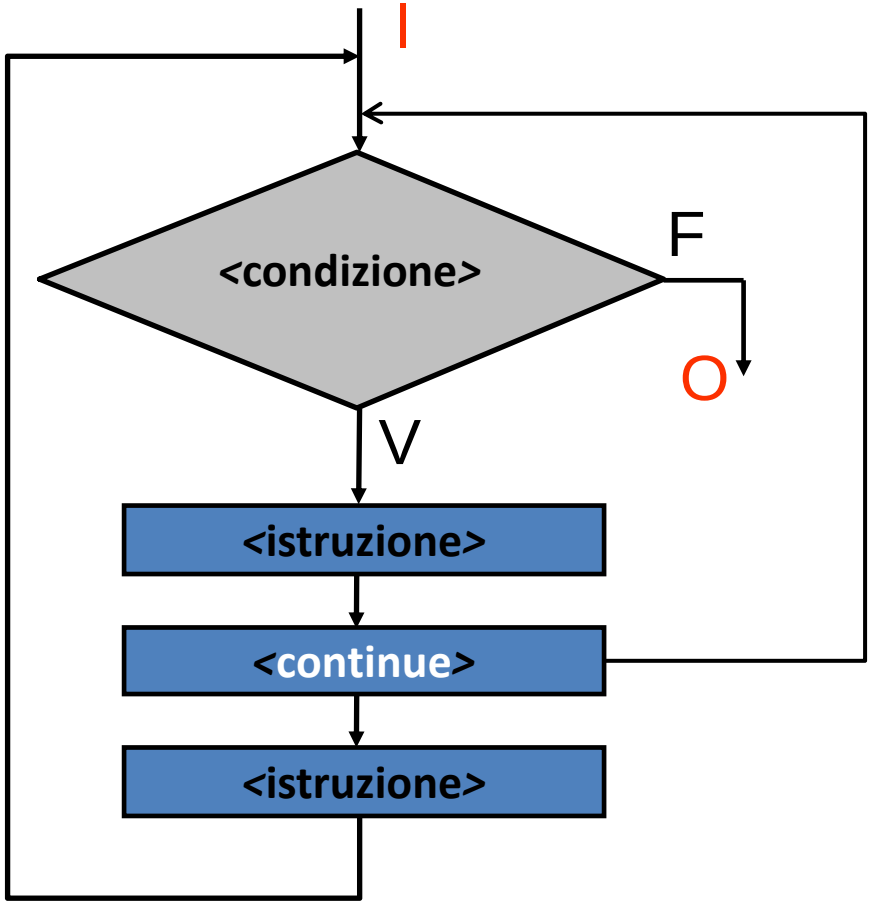
Istruzione while: break e continue

Le strutture di controllo iterative sono corredate dalle istruzioni **break** e **continue**, che consentono rispettivamente di anticipare l'uscita dal ciclo, o di saltare un'iterazione.

continue

L'istruzione continue ha come effetto l'interruzione dell'iterazione corrente. Il controllo di flusso rimane confinato all'interno del ciclo, ma viene reindirizzato all'iterazione successiva in conseguenza di una circostanza inattesa che invalida o rende superflua l'esecuzione di tale iterazione.

Anche l'esecuzione di continue è spesso subordinata alla valutazione di una espressione booleana che determina il ritorno immediato alla valutazione dell'espressione



Istruzione while: break e continue

- inizializza i a 9
- finché i è maggiore di 0
- stampa Hello <valore di i>
- decrementa i

```c
#include <stdio.h>

int main()
{
  int i = 10;

  while ( i > 0 ){
    printf("Hello %d\n", i );
    i = i -1;
  }
}
```

Decremento con uso di break

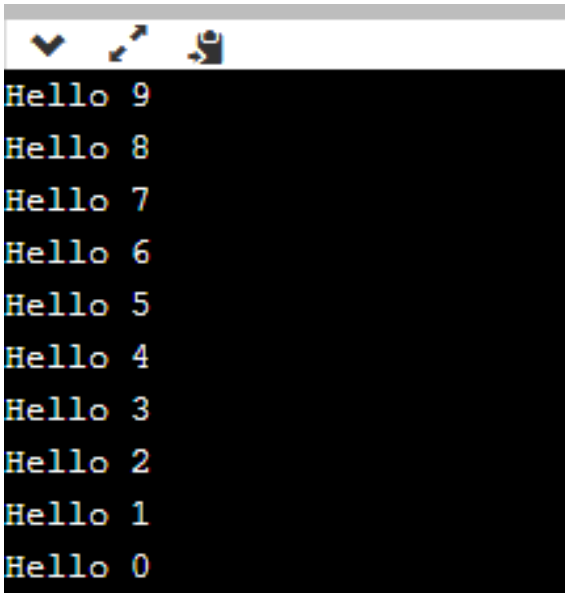
```c
#include <stdio.h>

int main()
{
  int i = 9;

  while ( 1 ){
    printf("Hello %d\n", i );
    i = i -1;

    if(i<0)
      break;
  }
}
```

Esecuzione



Istruzione while: break e continue

Si scriva un programma che prenda un numero di valori interi positivi in input non definito a priori dall'utente, ma interrotta su richiesta dell'utente.

Algoritmo

Int val;

finché **1** è vera:
    stampa("Inserisci un valore intero positivo
        (-1 per interrompere): ");
    leggo un valore **val** da tastiera;
    Se **val** è < **0**
        break;

Esecuzione

```
Inserisci un valore intero positivo (-1 per interrompere): 5
Inserisci un valore intero positivo (-1 per interrompere): 6
Inserisci un valore intero positivo (-1 per interrompere): 7
Inserisci un valore intero positivo (-1 per interrompere): 8
Inserisci un valore intero positivo (-1 per interrompere): -1
```

Programma

```
#include <stdio.h>

int main()
{
  int val;

  while(1){
    printf(" Inserisci un valore intero positivo
                (-1 per interrompere): ");
    scanf("%d", &val);

    if(val<0)
      break;
  }
}
```

Istruzione while: break e continue

Si scriva un programma che prenda un numero di valori interi positivi in input non definito a priori dall'utente, ma interrotta su richiesta dell'utente.

Algoritmo

Int val = 0;

finché **val**>=**0** è vera:

    stampa("Inserisci un valore intero positivo
                (-1 per interrompere): ");
    leggo un valore **val** da tastiera;

Esecuzione

```
Inserisci un valore intero positivo (-1 per interrompere): 5
Inserisci un valore intero positivo (-1 per interrompere): 6
Inserisci un valore intero positivo (-1 per interrompere): 7
Inserisci un valore intero positivo (-1 per interrompere): 8
Inserisci un valore intero positivo (-1 per interrompere): -1
```

Programma

```
#include <stdio.h>

int main()
{
  int val=0;

  while(val>=0){
    printf("Inserisci un valore intero positivo
                (-1 per interrompere): ");
    scanf("%d", &val);
}
}
```

Istruzioni while e continue

Scrivere un programma che legge da input numeri interi positivi minori o uguali a 100, somma solo quelli pari, e termina stampando prima il valore della somma calcolata, quando viene inserito un numero maggiore di 100

Algoritmo

int val = 0;
Int somma = 0;

finché **1** è vera:
    stampa(“Inserisci un valore intero positivo
        <=100 (>100 per interrompere): “);
    leggo un valore **val** da input;
    Se **val** è < **0** o **val>100**
        break;
    Se **val%2 ≠ 0**
        continue;
    Altrimenti
        somma += val;
stampa(somma);

Programma

main.c

```c
#include <stdio.h>

int main()
{
    int val;
    int somma = 0;

    while(1){
        printf("Inserisci un valore intero positivo <100"
        "(>100 per interrompere): ");
        scanf("%d", &val);

        if(val<0 || val>100)
            break;
        if(val%2 != 0)
          continue;
        else
          somma +=val;
    }

    printf("La somma è: %d\n", somma);
}
```

Istruzioni while e continue

Scrivere un programma che legge da input numeri interi positivi minori o uguali a 100, somma solo quelli pari, e termina stampando prima il valore della somma calcolata, quando viene inserito un numero maggiore di 100

Algoritmo

int val = 0;
Int somma = 0;

finché **val**>=**0** && **val**<=**100** è vera:
    stampa(“Inserisci un valore intero positivo
        <=100 (>100 per interrompere): “);
    leggo un valore **val** da input;
    Se **val%2 ≠ 0**
        continue;
    Altrimenti
        somma += val;
stampa(somma);

Programma

main.c

```c
#include <stdio.h>

int main()
{
    int val;
    int somma = 0;

    while(val>=0 && val<=100){
        printf("Inserisci un valore intero positivo <100"
        "(>100 per interrompere): ");
        scanf("%d", &val);

       if(val%2 != 0)
          continue;
       else
          somma +=val;
    }

    printf("La somma è: %d\n", somma);
}
```

Istruzione for

In altri linguaggi il costrutto **for** permette di eseguire una istruzione, semplice o composta, per un numero prefissato di volte (ciclo a contatore).

Nel linguaggio C è più generale, al punto da poter essere assimilata ad una particolare riscrittura del costrutto **while**.

Sintassi:

for (<**inizializzazione**>; <**condizione**>; <**aggiornamento**>)
<**istruzione**>;

Istruzione for

<**condizione**> è un'espressione logica;

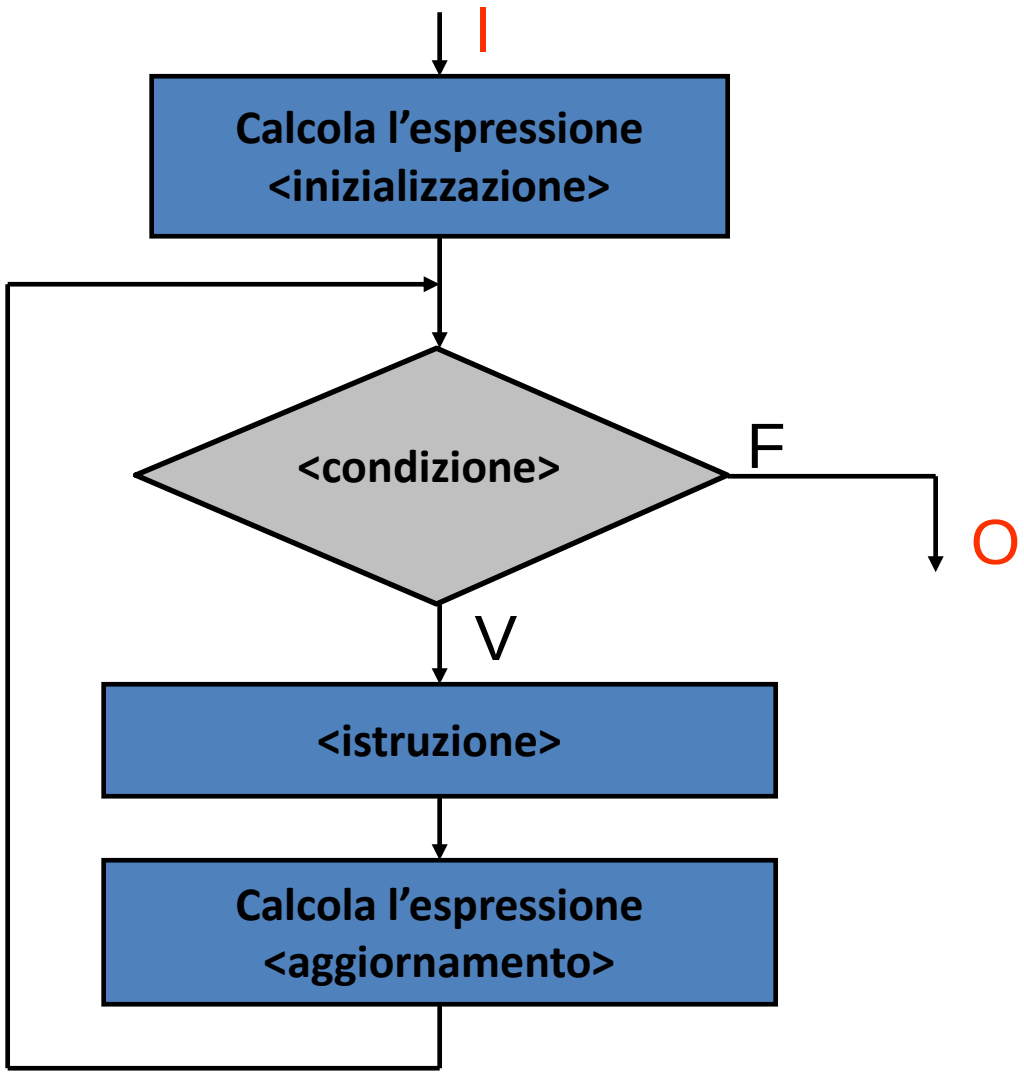
<**inizializzazione**> e <**aggiornamento**> sono invece espressioni di tipo qualsiasi.

L'istruzione **for** opera secondo il seguente algoritmo:

viene calcolata <**inizializzazione**>;
finché <**condizione**> è vera (valore non nullo) ;
viene eseguita <**istruzione**>;
viene calcolato <**aggiornamento**>;

Istruzione for

Diagramma di flusso:



Istruzione for

Di fatto il costrutto **for** è del tutto equivalente al seguente frammento di programma:

```
<inizializzazione>
while (<condizione>)
 {
    <istruzione>
    <aggiornamento>
 }
```

Poiché non vi sono restrizioni sulla <**istruzione**> da eseguire nel corpo del ciclo, questa può contenere a sua volta istruzioni **for** (for annidati), o altri costrutti di controllo (**if**, **while**, **switch**, ecc.).

Istruzione for

Esempi

Per **i** che va da **1** a **100** con passo **1**
  <istruzione>;

```
for( i=1; i<100 ; i++)
  <istruzione>;
```

Per **i** che va da **100** a **0** con passo **1**
  <istruzione>;

```
for( i=100; i>=0 ; i--)
  <istruzione>;
```

Per **i** che va da **25** a **75** con passo **5**
  <istruzione>;

```
for( i=25; i<=75 ; i+=5)
  <istruzione>;
```

Per **i** che va da **25** a **75** con passo **5**
con **j** che va da 0 a 10 con passo 1
  <istruzione>;

```
for( i=25,j=0; i<=75 ; i+=5,j++)
  <istruzione>;
```

```
for( i=25,j=50; i<=75 ; i+=5,j-=1)
  <istruzione>;
```

Per **i** che va da **25** a **75** con passo **5**
con **j** che va da 50 a 40 con passo **1**
  <istruzione>;

Istruzione for

Esempi

Per **i** minore di **100** con passo **1**
  <**istruzione**>;

```
for( ; i<100 ; i++)
  <istruzione>;
```

Per **i** che va da **100** a **infnito** con passo **1**
  <**istruzione**>;

```
for( i=100;  ; i++)
  <istruzione>;
```

Ciclo infinito
  <**istruzione**>;

```
for(; ;)
  <istruzione>;
```

```
i=0;
for( ; ;){
  <istruzione>;

  i+=1;
  if(i==100)
    break;
}
```

```
i=0;
while(1){
  <istruzione>;

  i+=1;
  if(i==100)
    break;
}
```

Sono equivalenti

Istruzione for

Problema: leggere da tastiera un valore intero **N** e un carattere **carat**, e visualizzare una riga di **N** caratteri **carat**

esempio: **N** = 10, **carat** = '*' output **********

soluzione iterativa: **ripeti N volte l'operazione "stampa carat"**

Programma

main.c

```
1  #include <stdio.h>
2
3  main() {
4  int N, indice;
5  char carat;
6
7  printf ("Introduci un carattere: ");
8  scanf("%c", &carat);
9
10 printf ("Introduci un intero: ");
11 scanf("%d", &N);
12
13 for (indice=0; indice<N; indice++)
14         printf ("%c", carat);
15
16 printf ("\n");
17 }
```

Esecuzione

```
Introduci un carattere: #
Introduci un intero: 16
################
```

Istruzione for

Leggere da tastiera un numero intero **N**; successivamente leggere da tastiera **N** numeri interi, e calcolarne la **media**.

Inoltre si controlli che ogni numero inserito sia compreso tra **0** e **30**; in caso contrario, il numero deve essere ignorato.

Al termine visualizzare la media dei soli valori validi.

Analisi:

Problema iterativo: si devono leggere **N** numeri da tastiera.
Calcolo della media e controllo di ogni valore inserito (tra **0** e **30**).

Istruzione for

Algoritmo

float totale = 0;
float media;
int cont_validi = 0;
Leggi N da tastiera;
Per **ind** che va da **1** a **N**:
    leggo un valore **num** da tastiera;
    Se **num** è < **0** oppure > **30**
        stampa(**non è un valore valido**),
    altrimenti:
        accumula **num** in **totale**;
        incrementa di 1 **cont_validi**;

Calcola la **media** = **totale**/**cont_validi**

Stampa(**media**)

Programma

Svolgere come esercizio

Istruzione for

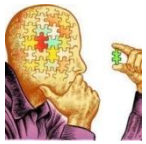
Realizzare un programma in grado di generare un triangolo di "*" sul video, le cui dimensioni (numero di righe su cui si sviluppa il triangolo) siano fornite da tastiera.

```
*
***
*****
*******
*********
***********
```

Poiché il video è composto da 80 colonne, sia 80 la dimensione massima.

Istruzione for

Realizzare un programma in grado di generare un triangolo di "*" sul video, le cui dimensioni (numero di righe su cui si sviluppa il triangolo) siano fornite da tastiera.



Ragionamento

Una possibile organizzazione del programma potrebbe essere costituita dalla seguente struttura:

controlla se il numero di colonne (calcolato in base al numero di righe) è < di 80;
se **nrighe** è il numero di righe da stampare, per **nrighe** volte esegui:

- scrivi un certo numero di spazi;
- scrivi un certo numero di "*".

Istruzione for

Realizzare un programma in grado di generare un triangolo di "*" sul video, le cui dimensioni (numero di righe su cui si sviluppa il triangolo) siano fornite da tastiera.

Idea

Per la relazione tra il numero di spazi ed il numero di '*' si consideri la figura:

```
1ª riga          *
2ª riga        * * *
3ª riga      * * * * *
nª riga    * * * * * * *
```

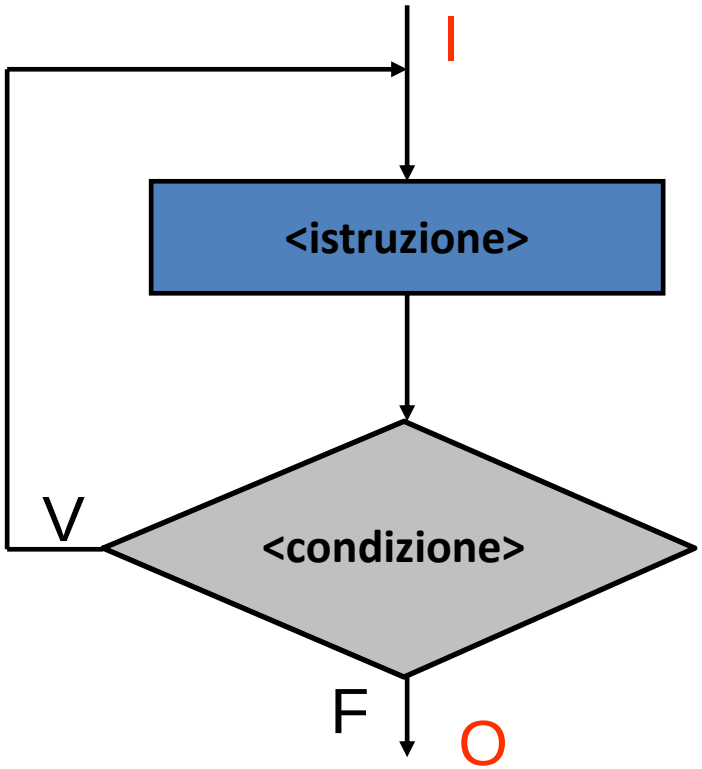
N° spazi - nell'ultima riga se ne devono stampare **0**, nella penultima **1**, nella terzultima **2**, ecc.: in una generica riga **rigacorr** saranno: **nrighe** - **rigacorr**;

N° asterischi - nella prima riga se ne deve stampare 1, nella seconda 3, nella terza 5, ecc.: nella generica riga **rigacorr** saranno: (**rigacorr***2**) - **1**

Istruzione do-while

Sintassi:

```
do
  <istruzione>
while (<condizione>)
```



Ripeti <**istruzione**>,
che può essere semplice
o composta,
finché <**condizione**>
è vera.

Istruzione do-while

L'istruzione **do** ... **while** realizza il costrutto **repeat - until** della programmazione strutturata.

<**condizione**> deve essere di tipo logico ed è calcolata ad ogni iterazione;

<**istruzione**> pertanto è sempre eseguita almeno una volta (anche se <**condizione**> è subito falsa);

se <**condizione**> non diventa mai falsa non si esce mai dal loop!

come per gli altri costrutti <**istruzione**> è una normale istruzione composta e può contenere qualsiasi tipo di istruzione o costrutto (altri **while**, **if**, **switch**, ecc.), dando origine a strutture annidate.

Esempio: calcolo del seno

- Calcolare la funzione **seno x** mediante lo sviluppo in serie

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

sino a quando i fattori sono > 0.00005.

N=1 $\quad \frac{x^3}{3!} = x\frac{-x \cdot x}{1 \cdot 2 \cdot 3}$

N=2 $\quad \frac{x^5}{5!} = \frac{x \cdot x \cdot x \cdot x \cdot x}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} = \frac{x^3 \cdot x^2}{(1 \cdot 2 \cdot 3)(4 \cdot 5)} = \frac{x^3}{(1 \cdot 2 \cdot 3)}\frac{x^2}{(4 \cdot 5)} = \boxed{\frac{x^3}{3!}}\frac{-x^2}{(4 \cdot 5)}$

$$fatt_N = fatt_{N-1} \cdot \frac{-x \cdot x}{(N+1)(N+2)}$$

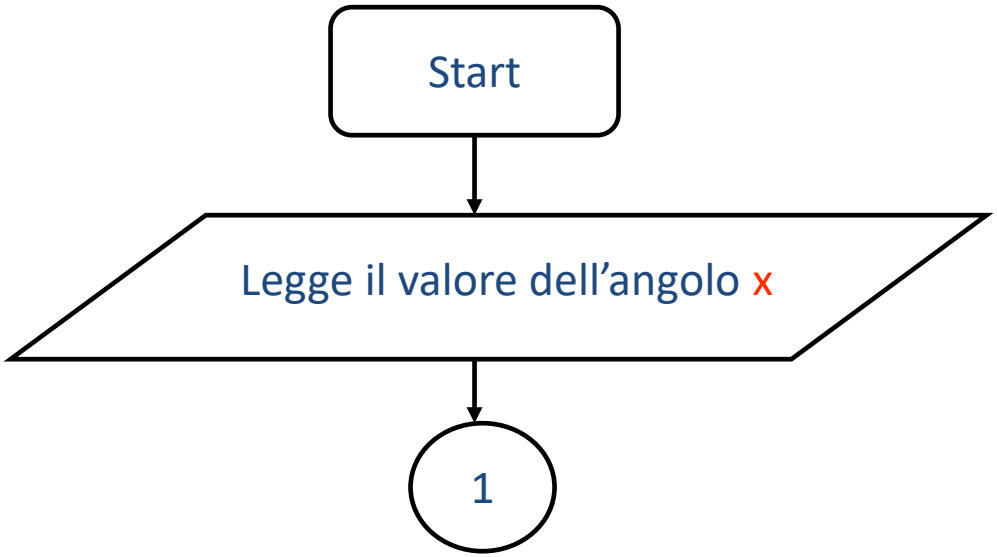
con **N** che varia a passi di 2.

Esempio: calcolo del seno

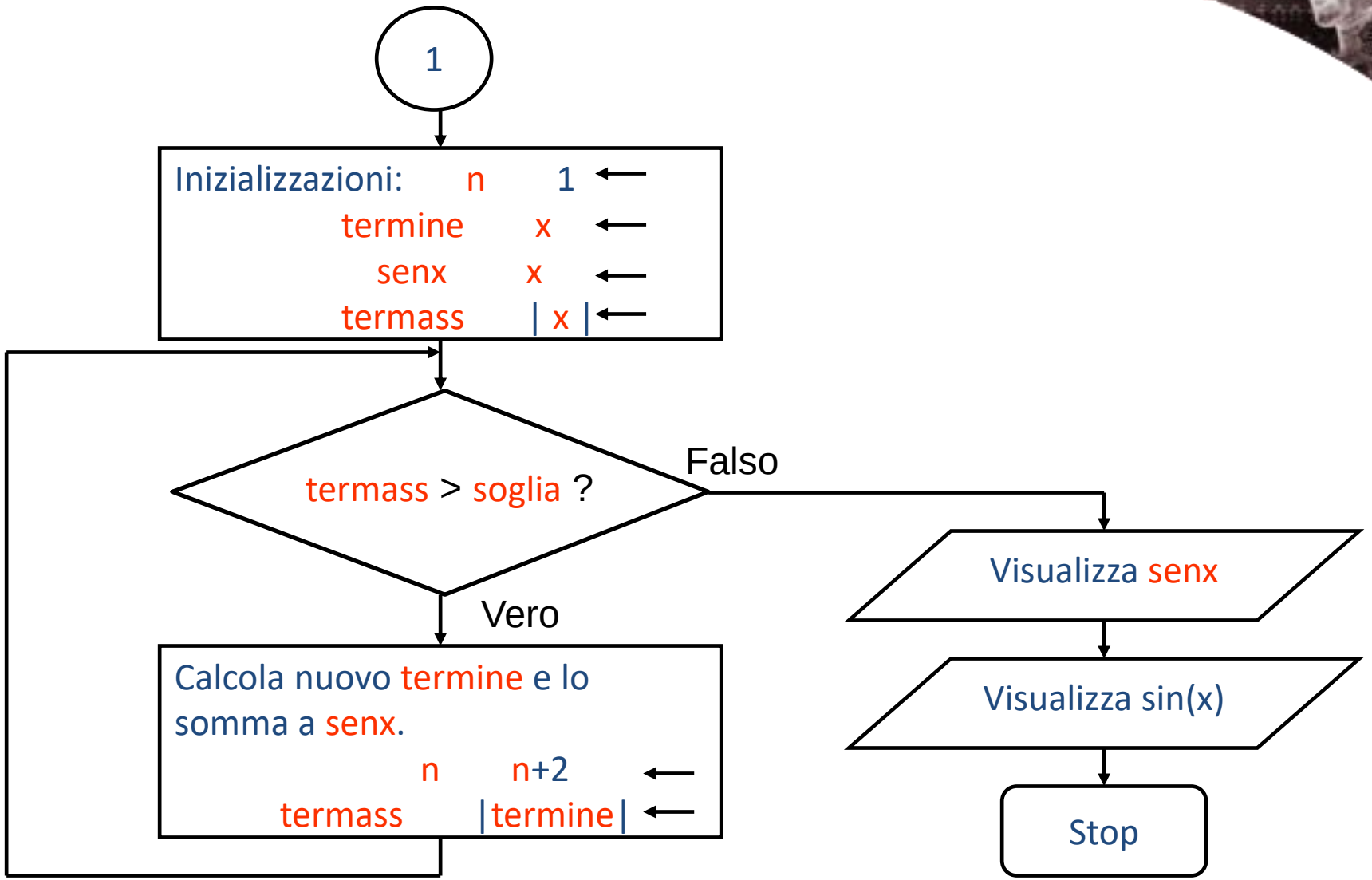
Il primo termine lo conosciamo all'atto della lettura da tastiera del valore in radianti dell'angolo.

Tutti gli altri termini possono essere ricavati in successione a partire dal primo applicando la relazione che abbiamo trovato che lega un termine al successivo.

Diagramma di flusso



Esempio: calcolo del seno



Esempio: calcolo del seno

```
#include <stdio.h>
#include <math.h>
const double soglia = 0.00005;

Int main()
{
    double x, senx, termine, termass, n;

    printf ("Angolo in radianti: ");
    scanf ("%lf ", &x);
    n = 1;
    termine = x;
    senx = x;
    if (x < 0)
        termass = - x;
    else
        termass = x;
```

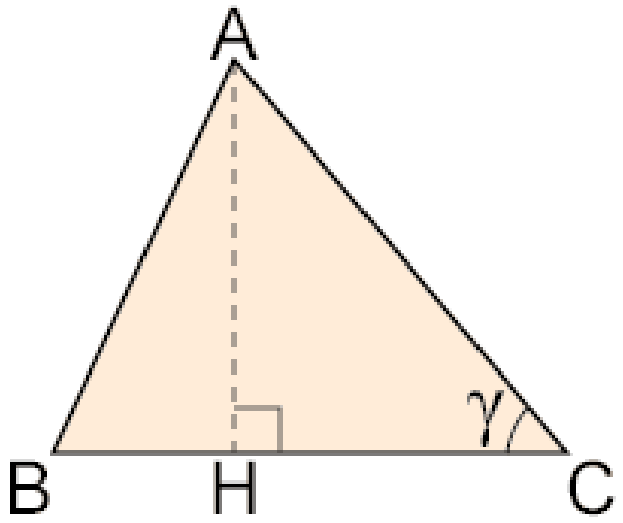
Inizializzazioni

Esempio: calcolo del seno

```
/*          Ad ogni ciclo calcola un nuovo contributo          */
while(termass > soglia)
{
    termine = - termine * (x * x) / ((n+1) * (n+2));  /* nuovo termine */
    senx += termine;                                  /* accumula in senx */
    n += 2;                                           /* aggiorna n */
    if (termine < 0)          /* aggiorna il valore assoluto di termine  */
        termass = -termine;
    else
        termass = termine;
  }
  printf("\nIl seno di %lf e' %lf\n", x, senx);
  printf("\nValore fornito dalla funzione di libreria:  %lf\n", sin(x));
}
```

Esercizio: angoli di un triangolo

In geometria, il **teorema del coseno** esprime una relazione tra la lunghezza dei lati di un triangolo e il coseno di uno dei suoi angoli. Può essere considerato una generalizzazione del **teorema di Pitagora** al caso di triangoli non rettangoli.



$$\overline{AB}^2 = \overline{AC}^2 + \overline{BC}^2 - 2 \cdot \overline{AC} \cdot \overline{BC} \cos\gamma$$

Esempio: calcolo del seno

Programma

main.c

```c
#include <stdio.h>
#include <math.h>

int main()
{
    float a, b, c;
    float alpha, beta, gamma;

    printf("inserisci il lato a: ");
    scanf("%f", &a);
    printf("inserisci il lato b: ");
    scanf("%f", &b);
    printf("inserisci il lato c: ");
    scanf("%f", &c);

    if(a<0 || b<0 || c<0){
      printf("I lati del triangolo devono"
      " essere valori positivi\n");
    }
    else{
      alpha = acos((c*c + b*b - a*a)/(2*c*b));
      beta = acos((a*a + c*c - b*b)/(2*a*c));
      gamma = acos((a*a + b*b - c*c)/(2*a*b));

      printf("Gli angoli sono\nalpha: %f\nbeta: %f\ngamma: %f\n", alpha, beta, gamma);
      printf("Verifichiamo che la somma sia pi-greco: %f", alpha+beta+gamma);
    }

    return 0;
}
```

Esecuzione

```
inserisci il lato a: 5
inserisci il lato b: 8
inserisci il lato c: 12
Gli angoli sono
alpha: 0.307395
beta: 0.505360
gamma: 2.328837
Verifichiamo che la somma sia pi-greco: 3.141593
```

Esercizi

1. Calcolare il valore della serie armonica

$$y(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

terminando il calcolo quando un fattore contribuisce per meno di 0.00005.

La serie armonica si trova in numerose applicazioni quale, ad esempio, il calcolo degli interessi composti.

2. Calcolo della Conversione da decimale a Binario