COSTRUTTI CICLICI

Costituiscono un modo per indicare che un blocco di istruzioni va rieseguito ciclicamente.

Elementi essenziali per un costrutto di iterazione:

- **inizializzazione**
  le variabili interessate, ed in particolare quelle usate nell'espressione della condizione, devono essere inizializzate prima della valutazione della condizione.
- **test**
  deve essere prevista una fase di valutazione della condizione (di permanenza nel ciclo) che determini la ripetizione o la terminazione del ciclo
- **modifica**
  almeno una delle variabili della condizione deve essere modificata all'interno del ciclo, in modo che prima o poi la condizione di ripetizione diventi falsa (terminazione).

Esecuzione dei programmi

Esecuzione «manuale» di un programma

Istruzioni da eseguire

```
.....
int w, y, z;
int sp, ns;

leggi w;
leggi y;

sp=0;
ns=y;

while (ns >0)
 {
  sp=sp+w;
  ns=ns-1;
 }
z=sp;
scrivi z;
```

Nella tabella è riportata la traccia di esecuzione con i valori assunti dalle variabili nei punti significativi *A, B, .. F*
(si supponga che i dati in ingresso siano 7 e 3)

Punti della traccia di esecuzione

	A	*B*	*C*	*D*	*E*	*F*
w	??	7	7	7	7	7
y	??	3	3	3	3	3
sp	??	0	7	14	21	21
ns	??	3	2	1	0	0
z	??	??	??	??	??	21

A: valore dopo la dichiarazione
B: valore dopo l'inizializzazione
C: valore dopo la 1ma iterazione del ciclo
D: valore dopo la 2nda iterazione del ciclo
E: valore dopo l'ultima iterazione del ciclo
F: valore al termine del programma

Istruzioni: «in memoria» scritte e leggibili
Variabili: «in memoria» scrivibili e leggibili

```c
#include <stdio.h>
int main(int argc, char *argv[])
{

int w, y, z;
int sp, ns;

scanf("%d", &w);
scanf("%d", &y);
printf("w = %d    y= %d\n", w,y);
sp=0;
ns=y;
printf("ns = %d    sp= %d\n", ns,sp);
while (ns >0)
{
sp=sp+w;
ns=ns-1;
printf("ns = %d    sp= %d\n", ns,sp);
}
z=sp;
printf("prodotto z=  %d\n", z);


    return 0;
}
```

Iteration

- 2 Basic types of repetition control

- Counter controlled

  loop is done until counter reaches a predetermined ending value

  needs: NAME of counter, INITIAL VALUE, INCREMENT amount, and FINAL VALUE

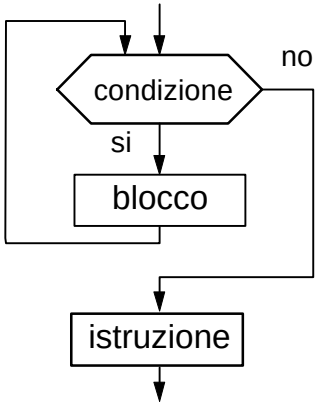
- Sentinel Controlled

  looping continues until some event occurs or some value is encountered

CICLO A CONDIZIONE INIZIALE (Sentinel Controlled)

Sintassi

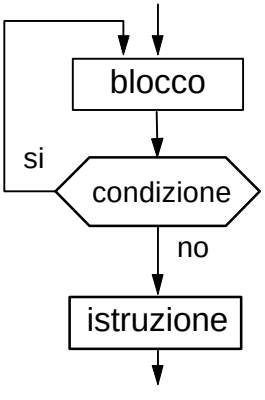
```
while (<condizione>)
{
    <blocco>
}
<istruzione>
```



CICLO A CONDIZIONE FINALE

Sintassi

```
do
{
    blocco
} while (<condizione>);
<istruzione>
```

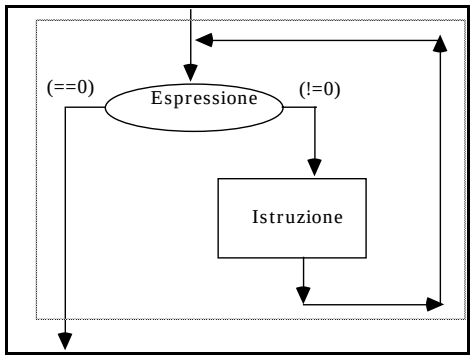


Istruzioni di iterazione: `while`

Consente la ripetizione dell'istruzione componente in modo controllato da una espressione.

Sintassi:

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



Significato

L'espressione (condizione di ripetizione) viene valutata all'**inizio di ogni ciclo**.

- L'<istruzione> viene eseguita finche` il valore dell'<espressione> rimane *vero* (diverso da zero).
- Si esce dal ciclo quando l'espressione restituisce un valore =0 (*falso* logico).
- Se inizialmente <espressione> ha valore zero, il corpo del ciclo non viene **mai** eseguito.

Esempio: somma dei primi dieci interi

```
#include <stdio.h>

main()
{
int somma, j;
somma=0;
j = 1;
while (j <= 10)
  {somma = somma + j;
   j++;
  }
printf("Risultato: %d\n", somma);
}
```

Esempio: scarto dei blank in lettura

```
char car=' ';
while (Car==' ')
  {
   scanf("%c",&Car);
  }
```


Esercizio:

Scrivere un programma che calcoli la media degli N voti riportati da uno studente.

```
/* Media di n voti */
#include <stdio.h>
main()
{
  int voto,N,i;
   float  media, sum;

   printf("Quanti sono i voti ?");
   scanf("%d",&N);
  sum=0;
/* ripeti ...*/
   printf("Dammi un voto:");
  scanf("%d",&voto);
  sum=sum+voto;
/* ... per N volte */
  media=sum/N;
  printf("Risultato: %f",media);
}
```

Esercizio (continua):

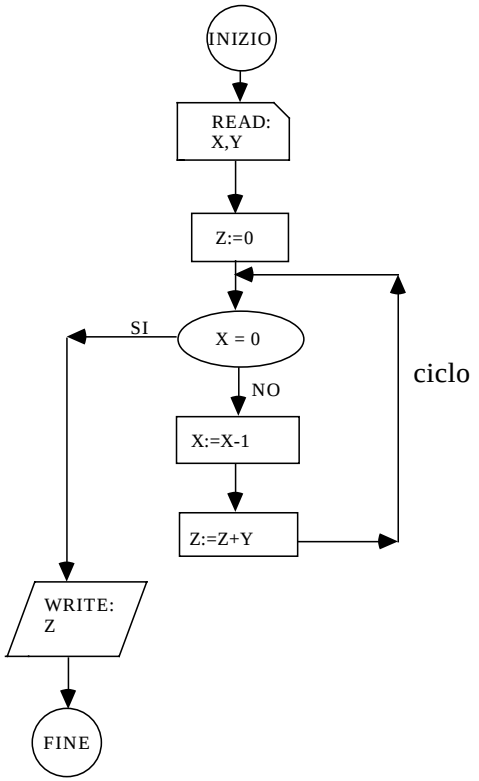
```
/* Media di n voti*/
#include <stdio.h>

main()
{
  int voto,N,i;
  float sum, media;

  printf("Quanti sono i voti ?");
  scanf("%d",&N);
  sum=0;
  i=1;
  while (i <= N)
    { /* corpo ciclo while */
    printf("Dammi il voto n.%d:",sum);
    scanf("%d",&voto);
    sum=sum+voto;
    i=i+1;
    }
  media=sum/N;
  printf("Risultato: %f",media);
}
```


✏ Esercizio:

Programma che calcola il prodotto X*Y come sequenza di somme (si suppone Y>0, X >=0).



Codifica:

```
/* moltiplicazione come sequenza di
   somme */
#include <stdio.h>
main()
{
 int  X,Y,Z;

 printf("Dammi i fattori:");
 scanf("%d%d",&X,&Y);
 Z=0;
 while (X!=0)
     {  /* corpo ciclo while */
          Z=Z+Y;
          X=X-1;
        }
 printf("%d",Z);
}
```


Cicli Innestati

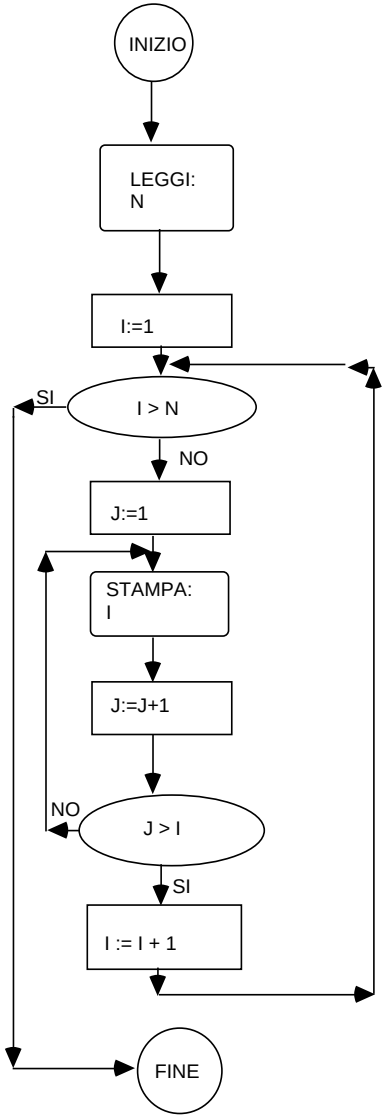
```
while (<espressione>)
        <istruzione>;
```

- <istruzione> puo` essere una qualunque singola istruzione (semplice o strutturata): eventualmente anche una istruzione while ➠ cicli **innestati**

✏ **Esercizio:**

Si legga un numero naturale N. Stampare una volta il numero 1, due volte il numero 2,..., N volte il numero N.

- Dominio di ingresso (0,1,2,...,N)
- Dominio di uscita ((), (1), (1,2,2),(1,2,2,3,3,3)...).



Codifica:

```
#include <stdio.h>
main()
{
  int    N,I,J;

  printf(“dammi N:”);
  scanf("%d",&N);
  I=1;
  while (I<=N)
   {  /* corpo ciclo esterno */
      printf(“Prossimo valore:”);
      printf("%d",I);
      J=1;
      while (J<I)
         {  /* corpo ciclo interno */
            printf("%d",I);
            J=J+1;
         }
      I=I+1;
   }
}
```

✏ **Esercizio:**
Stabilire se un numero naturale N e` primo.
(Un numero naturale N e` primo, se non e` divisibile per alcun numero intero minore di esso.)

Nota: non e` necessario eseguire tutte le divisioni di N per 2, 3, ..., (n-1), ma basta eseguire le divisioni per i naturali minori o uguali alla radice quadrata di N.

```
/* Numero primo */
#include <stdio.h>
#include <math.h>
main()
{
  typedef enum {false,true} boolean;
  int     N, I;
  float   N1;
  boolean primo;

  scanf("%d",&N);
  N1=sqrt(N);
  I=2;primo=true;
  while ((I<=N1) && (primo==true))
   {if (((N / I) * I) == N)
        {primo=false;}
    else  I=I+1; }
  if (primo==true)
      printf("numero primo");
  else printf("numero non primo");
}
```


✏ **Esercizio:**

Calcolo del **fattoriale** di un numero intero non negativo N:

- fattoriale(0) = 1
- fattoriale(N) = N * (N-1)*...*1=fattoriale(N-1)*N

```
/* Calcolo del fattoriale */
#include <stdio.h>
#include <math.h>
main()
{
  int     N, F, I;

  printf("Dammi N:");
  scanf("%d",&N);
  F=1;
  I=2;
  while (I <= N)
        { F=F*I;
          I=I+1; }
  printf("%s%d","Fattoriale: ",F);
}
```

Esempio:

Divisione tra numeri interi positivi attraverso somma e sottrazione.

```
/* divisione tra interi positivi */
#include <stdio.h>
main()
{
  unsigned int  X,Y,Quoz,Resto;

 scanf("%d%d",&X,&Y);
 Resto=X;
 Quoz=0;
 while (Resto >= Y)
       { Quoz=Quoz+1;
         Resto=Resto-Y; }
 printf("%d%s%d%s%d%s%d",
     X," diviso ", Y, " = ", Quoz,
     " resto ", Resto);
}
```

Istruzioni Ripetitive: for

E` una istruzione di ripetizione particolarmente adatta per realizzare *cicli a contatore*.

Sintassi:

```
for(<espressione1>; <espressione2>;  <espressione3>)
  <istruzione>;
```

Significato:

- <espressione1> e` l'espressione di inizializzazione: viene eseguita una volta sola, prima di entrare nel ciclo
- <espressione2> rappresenta la condizione di permanenza nel ciclo (valutata all'inizio di ogni iterazione)
- <espressione3> e` l'espressione di passaggio al ciclo successivo (valutata alla fine di ogni iterazione).

for è equivalente a:

```
<espressione1>;            /* Inizializzazione  */
while (<espressione2>)     /* Condizione di
                                 Ripetizione  */
{
      <istruzione>;
      <espressione3>;      /* InizioNuovoCiclo  */
}
```

Esempio while/for:

```
somma = 0;  /* while */
j = 1;
while (j <= n)
  {
   somma=somma+j;
    j++;
  }
```

con il for:

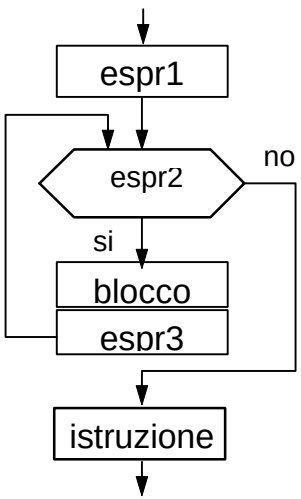
```
somma = 0; /* for */
for(j=1;j<=n;j++)
  somma = somma + j;
```

CICLO A CONTEGGIO: CICLO FOR

```
var_cont = val_iniz;
while (var_cont <= val_fin)
{
    <istruzioni del corpo del ciclo>
    var_cont = var_cont +1;
}
```

Il linguaggio C prevede anche un costrutto di ciclo che racchiude **inizializzazione**, **test** e **modifica** di una variabile.

```
for (espr1; espr2; espr3)
   {
    <istr del ciclo>
   }
   <istruzione>
```



espressione1: deve definire il valore iniziale della variabile di conteggio

espressione2: deve definire la condizione sul valore finale della variabile di conteggio

espressione3: deve definire la modifica della variabile di conteggio

```
for (var_cont=val_iniz; var_cont<=val_fin;var_cont++)
   {
    <istruzioni del corpo del ciclo>
   }
   <istruzione>
```

for()

```
for ( expr1 ; expr2 ; expr3 )
{
      statement1;
      statement2;
}
```

```
for ( ctr=1 ; ctr<10 ; ++ctr )
      sum += ctr;
```

expr1 - initializes the loop; done only once

expr2 - tests the condition; done at TOP of loop

expr3 - updates the loop; done AFTER statements in body of for()

Braces not needed if only one statement in body

Any or ALL expressions can be null statements

```
for ( ; ; );
```

Body may not be needed!

for()

Logic of for() is like that of while()

Compare the code:

```
int ctr;

ctr = 1;

while ( ctr < 10 )
{
      printf("Ctr = %d\n", ctr);
      ++ctr;
}
```

```
int ctr;

for( ctr = 1; ctr < 10; ++ctr )
      printf("Ctr = %d\n", ctr);
```

a for() loop

- initialize variables i, j, and k each to 1
- test to **end loop** when i > 10 **or** j > 3 * k
- increment i by 1, j by 2, and k by 2 * i
- print i, j, and k in the body of the loop

```
for ( i = 1, k = 1, j = 1;  i <= 10 && j <= 3 * k ; ++i, j += 2, k += 2 * i )
    printf("i = %d, j = %d, k = %d\n", i, j, k);
```

ESAGERATO

For:

```
for(<espressione1>; <espressione2>;  <espressione3>)
   <istruzione>;
```

- Ognuna delle espressioni **può essere omessa** (il punto e virgola deve rimanere)
- Se manca espressione2, si ha un ciclo infinito.

Cosa eseguono i seguenti **for** ?

```
for (i = 1;  i <= n;  i++)  printf("%d ",i);
for (;;)  { ... }
```

Esempio: fattoriale con istruzione for.

```
/* Calcolo del fattoriale */
#include <stdio.h>
#include <math.h>
main()
{
  int     N, F, I;

  printf(“Dammi N:”);
  scanf("%d",&N);
  F=1;
  I=2;
  for (I=2,F=1;I<=N;I++)
      F=F*I;
  printf("%s%d","Fattoriale: ",F);
}
```