Titolo unità didattica: Costrutti di controllo [04]

Titolo modulo : Costrutti di ripetizione [02-T]

Caratteristiche generali dei costrutti di ripetizione

Argomenti trattati:

- ✓ costrutto **for**
- ✓ costrutto **do-while (ripetere-finché)**
- ✓ costrutto **while**
- ✓ costrutti di ripetizione nidificati

Prerequisiti richiesti: P1-02-*-T

un **costrutto di ripetizione** denota la **ripetizione** (**iterazione**) di una squenza di istruzioni

i **costrutti di ripetizione** si distinguono in

- ✓ costrutti in cui il numero di ripetizioni (iterazioni) è noto *a priori*
- ✓ costrutti in cui il numero di ripetizioni (iterazioni) non è noto, ma dipende da una condizione (predicato)

la sequenza di istruzioni che compare in un costrutto di ripetizione è detta

blocco del ciclo

costrutti in cui il numero di ripetizioni è noto *a priori*

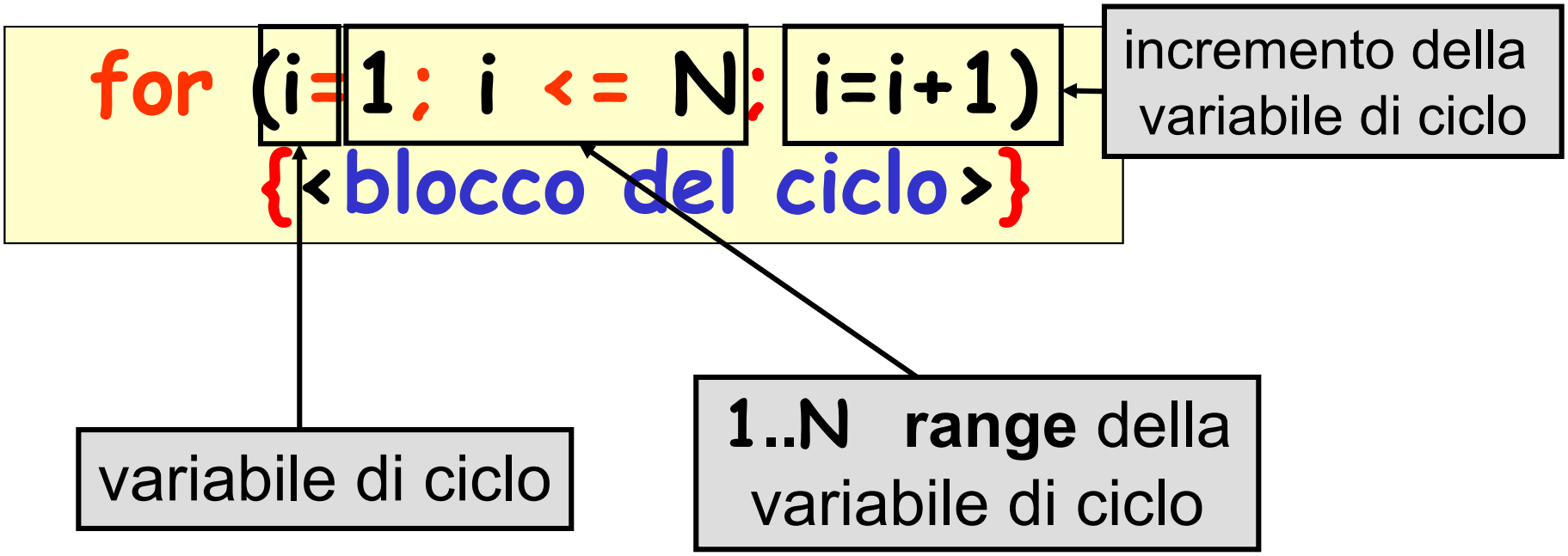
costrutto di ripetizione `for`

costrutti in cui il numero di ripetizioni non è noto, ma dipende da una condizione

costrutti di ripetizione
`while` , `do-while` (ripetere-finché)

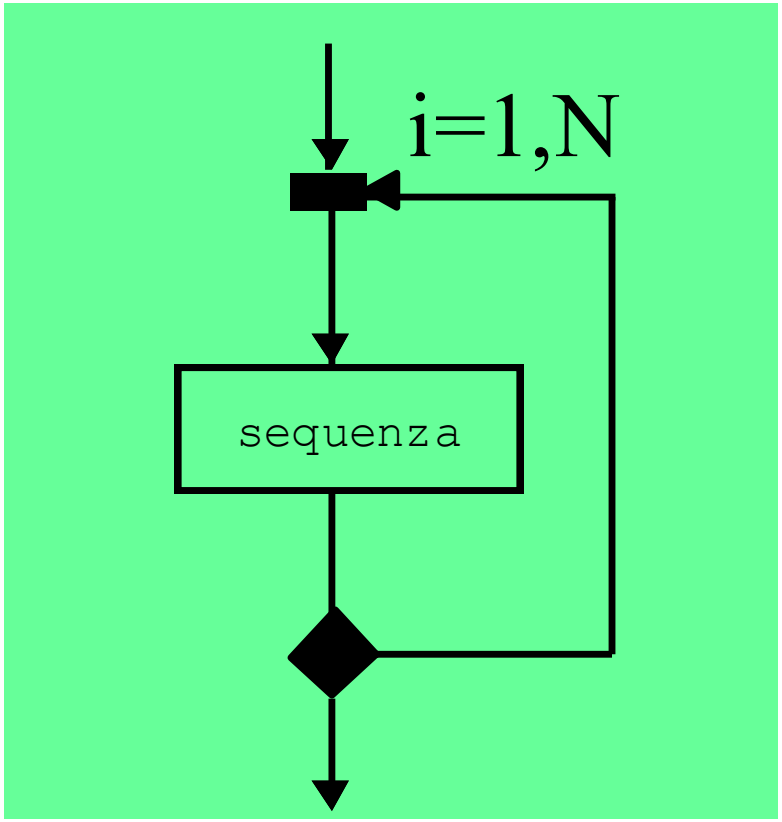
costrutto di ripetizione for
(o ciclo for)

consente di denotare la ripetizione N volte
di una sequenza di operazioni



costrutto di ripetizione for
(o ciclo for)

```
for (i=1; i <= N; i=i+1)
      {<blocco del ciclo>}
```



Esempio:
visualizzare in output i numeri interi da 1 a 10

```
int i;
 for (i = 1; i <= 10; i = i+1)
     {print i ; }
```

output:

```
1   2  ..........  9   10
```

forma generale

```
for (<variabile>=<espr_1>;<espr_2>;
     <variabile>=<espr_3> )
    { <blocco del ciclo>}
```

<espr_1> , < espr_3>
sono espressioni a valori interi

< espr_2> è un **predicato**: se è **vera** si itera

forma generale

```
for (
    valore iniziale della variabile di ciclo;

    espressione logica (se vera si itera,
    se falsa si esce dal ciclo ;

    incremento della variabile di ciclo
    )
        { <blocco del ciclo>}
```

Esempio:
visualizzare in output i numeri interi pari tra -8 e 10

```
int i;
 for (i = -8; i <= 10; i = i+2)
     {print i ; }
```

output:

```
-8   -6  ..........  8   10
```

Esercizio:
dati m e n (numeri pari con m<n), visualizzare in output i numeri pari tra m e n

Esercizio:
dati m e n (numeri dispari con m<n), visualizzare in output i numeri dispari tra m e n

Esercizio:
dato n, visualizzare in output i primi n numeri pari

Esercizio:
dato n, visualizzare in output i primi n numeri dispari

costrutto di ripetizione (o ciclo)
do-while (ripetere-finché)

denota la ripetizione in dipendenza del valore di un predicato (con test finale)

```
do
   { <blocco del ciclo> }
while (<predicato>);
```

predicato di permanenza nel ciclo

costrutto di ripetizione (o ciclo) do-while (ripetere-finché)

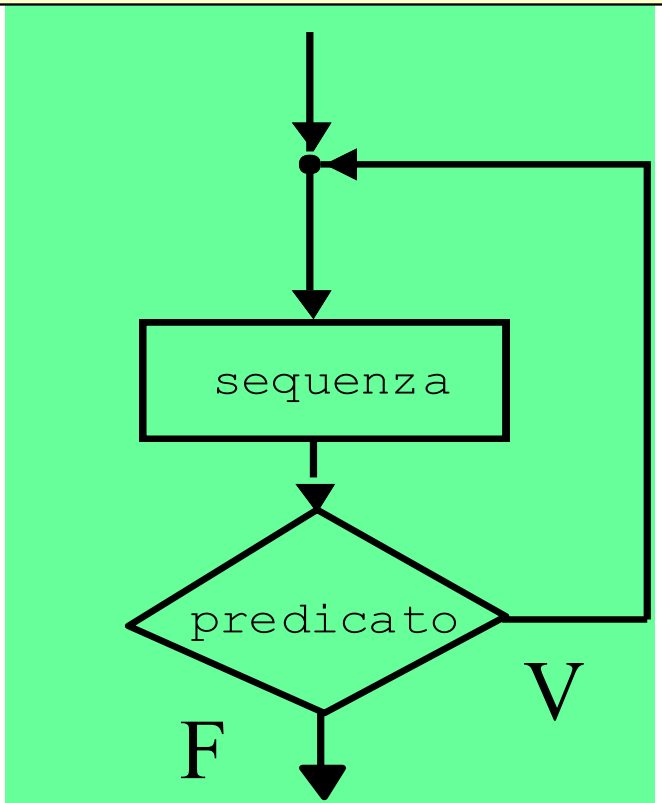
```
do
   { <blocco del ciclo> }
while (<predicato>) ;
```

il blocco del ciclo viene eseguito se il valore del <predicato> è vero;

se il valore del <predicato> è falso la ripetizione termina

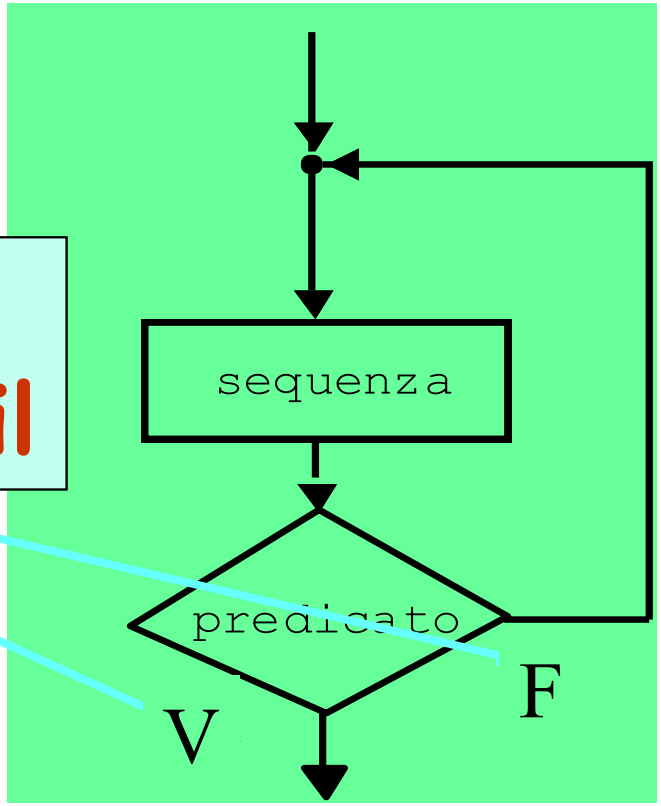
costrutto di ripetizione (o ciclo) do-while (ripetere-finché)

```
do
   { <blocco del ciclo> }
while (<predicato>) ;
```



il costrutto do-while può essere usato per ottenere il costrutto repeat-until

costrutto
repeat-until




```
do
   { <blocco del ciclo> }
while (<predicato negato>) ;
```

Esempio: visualizzazione interi da 1 a 10

```
int i;
i=0;
do
   { i = i+1 ;
     printf (i); }
while (i < 10) ;
```

output:

```
1   2  . . . . . .  9   10
```

Esempio: visualizzazione interi da 1 a 10

```
int i;
i=1;
do
   { printf (i);
     i = i+1 ; }
while (i <= 10) ;
```

Versione alternativa

output:

```
1   2  ..........  9   10
```

osservazione importante

in generale, il valore del predicato **deve cambiare** durante l'esecuzione del ciclo, per evitare un loop (sequenza infinita di operazioni)

almeno una delle variabili che compaiono nel predicato di uscita deve avere il proprio valore **modificato** da una istruzione del blocco del ciclo

```
int i;
i=0;
do
   { i = i+1 ;
     printf (i); }
while (i < 10) ;
```

```
int i;
i=1;
do
    { printf (i);
      i = i+1 ; }
while (i <= 10) ;
```

costrutto di ripetizione (o ciclo)
while

denota la ripetizione in dipendenza del valore di un predicato (con test iniziale)

```
while (<predicato>)
   { <blocco del ciclo>}
```

predicato di permanenza nel ciclo

costrutto di ripetizione (o ciclo)
while

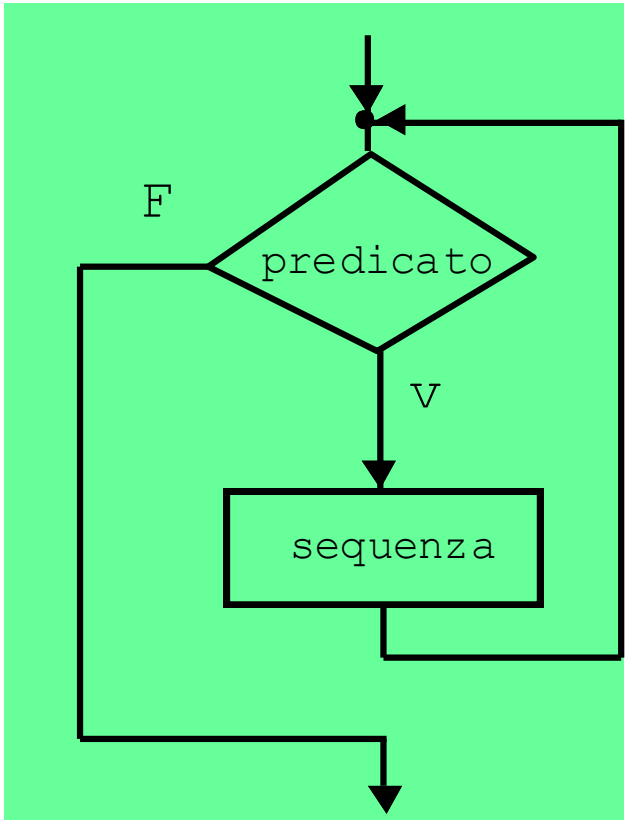
```
while (<predicato>)
   { <blocco del ciclo>}
```

il blocco del ciclo viene eseguito se il valore del <predicato> è vero;

se il valore del <predicato> è falso la ripetizione termina

costrutto di ripetizione (o ciclo)
while

```
while (<predicato>)
   { <blocco del ciclo>}
```



Esempio: visualizzazione interi da 1 a 10

```
int i ;
 i = 0;
 while (i < 10)
   { i = i+1 ;
     print i ; }
```

output:

```
1   2  .........  9   10
```

Esempio: visualizzazione interi da 1 a 10

```
int i ;
 i = 1;
 while (i <= 10)
   { print i ;
   i = i+1 ; }
```

Versione alternativa

output:

```
1   2  . . . . . .  9   10
```

osservazione importante

in generale, il valore del predicato **deve** cambiare durante l'esecuzione del ciclo, per evitare un loop (sequenza infinita di operazioni)

almeno una delle variabili che compaiono nel predicato di uscita deve avere il proprio valore **modificato** da una istruzione del blocco del ciclo

- ✓ il blocco di un ciclo **while** **può non** essere eseguito (ciò accede se il predicato di uscita ha valore falso);
- ✓ il blocco di un ciclo **do-while** è **sempre** eseguito almeno una volta

Esempio:
algoritmo di controllo di correttezza della risposta (inserita da tastiera) a una domanda

dato di input: la risposta (variabile **risposta**)

dato di output: nessuno: si visualizza (in alternativa): "risposta corretta", "risposta sbagliata, ripetere"

dati di input: dal dispositivo di input

output: sul dispositivo di output

```
{
int risposta;
const int risposta_corretta = 10 ;
 do
  { printf (“qual è il log in base 2 di 1024? ”) ;
   read (risposta) ;
   if (risposta == risposta_corretta)
        { printf (“risposta corretta”) ; }
      else
       {  printf (“risposta sbagliata”) ; }
   }
 while (risposta != risposta_corretta) ;
}
```

```
int risposta;
const int risposta_corretta = 10 ;
 do
  { printf (“qual è il log in base 2 di 1024? ”) ;
   read (risposta) ;
   if (risposta == risposta_corretta)
        { printf (“risposta corretta”) ; }
      else
       {  printf (“risposta sbagliata”) ; }
   }
 while (risposta != risposta_corretta) ;
```

modificare l’algoritmo per consentire al più 10 tentativi e visualizzare a ogni passo (dopo la domanda) il numero di tentativi rimanenti

un ciclo può trovarsi all'interno di un altro ciclo (**cicli innestati**)

```
.....
for (i=1;i<=2;i=i+1)
  { for (j=1;j<=3;j=j+1)
    {
      printf ( '(' ,i, ' ' ,j, ')' ) ;
    }
  }
```

ciclo esterno

ciclo interno

quante volte è eseguita la **printf** ?

output:

```
(1 1)(1 2)(1 3)(2 1)(2 2)(2 3)
```

inversione dei cicli

```
.....
for (j=1;j<=3;j=j+1)
  { for (i=1;i<=2;i=i+1)
     {
      printf ( '(' ,i, ' ' ,j, ')' ) ;
     }
  }
```

ciclo esterno

ciclo interno

quante volte è eseguita la **printf** ?

output:

```
(1 1)(2 1)(1 2)(2 2)(1 3)(2 3)
```