
Esercitazione con iterazioni

Filippo Cugini

Aspetti essenziali dell'iterazione

La maggior parte dei programmi si basa sull'iterazione, ovvero su cicli

Un ciclo è un gruppo di istruzioni che il computer esegue ripetutamente finché una qualche condizione di continuazione del ciclo rimane vera

Due modalità di iterazione:

1. Iterazione controllata da contatore
2. Iterazione controllata da sentinella

Aspetti essenziali dell'iterazione

L'iterazione controllata da **contatore** è talvolta chiamata iterazione definita, perché sappiamo in anticipo esattamente quante volte il ciclo sarà eseguito

L'iterazione controllata da **sentinella** è talvolta chiamata iterazione indefinita perché non si sa in anticipo quante volte sarà eseguito il ciclo

Aspetti essenziali dell'iterazione

Nell'iterazione controllata da **contatore** viene usata una variabile di controllo per contare il numero delle iterazioni

La variabile di controllo è incrementata (di solito di 1) ogni volta che viene eseguito il gruppo delle istruzioni

Quando il valore della variabile di controllo indica che è stato eseguito il numero corretto di iterazioni, il ciclo termina e l'esecuzione continua con l'istruzione dopo l'istruzione di iterazione

Aspetti essenziali dell'iterazione

I valori **sentinella** sono usati per controllare l'iterazione quando:

1. non si sa in anticipo il numero preciso di iterazioni
2. il ciclo comprende istruzioni che leggono dati ogni volta che è eseguito il ciclo

Il valore sentinella indica la “fine dei dati”

La sentinella è inserita dopo che tutti i dati regolari sono stati forniti al programma

Le sentinelle devono essere distinte dai dati regolari

Aspetti essenziali dell'iterazione

L'iterazione controllata da **contatore** richiede:

1. Il nome di una variabile di controllo (o contatore del ciclo)
2. Il valore iniziale della variabile di controllo
3. L'incremento (o il decremento) con cui la variabile di controllo è modificata ogni volta nel corso del ciclo
4. La condizione che verifica il valore finale della variabile di controllo (cioè se il ciclo deve continuare)

Aspetti essenziali dell'iterazione

Programma per contare fino a 10

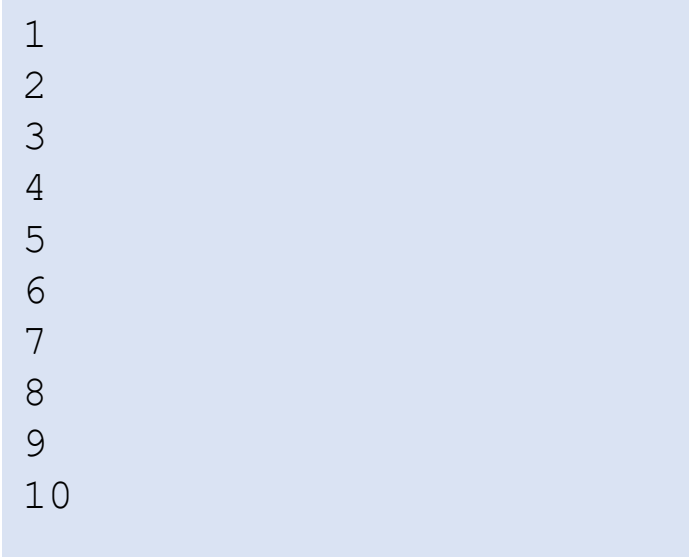
```
// Iterazione controllata da contatore
#include <stdio.h>

int main( void )
{
    unsigned int counter = 1; // contatore

    while (counter <= 10 ) {
        printf( "%u\n", counter );
        ++counter; // incremento
    }
}
```

Aspetti essenziali dell'iterazione

Output all'esecuzione del programma:



1
2
3
4
5
6
7
8
9
10

Aspetti essenziali dell'iterazione

```
unsigned int counter = 1;
```

La definizione e l'inizializzazione del contatore si possono scrivere anche come

```
unsigned int counter;  
counter = 1;
```

La definizione non è un'istruzione eseguibile, mentre lo è l'assegnazione

Si usano entrambi i metodi di impostazione dei valori delle variabili

Aspetti essenziali dell'iterazione

```
++counter;
```

incrementa di 1 il contatore del ciclo ogni volta che il ciclo è eseguito

La condizione di continuazione del ciclo nell'istruzione `while` verifica se il valore della variabile di controllo è minore o uguale a 10 (l'ultimo valore per il quale la condizione è vera)

Aspetti essenziali dell'iterazione

```
while (counter <= 10 ) {  
    printf( "%u\n", counter );  
    ++counter; // incremento  
}
```

Il corpo di questo `while` è eseguito anche quando la variabile di controllo ha valore 10

Il ciclo termina quando la variabile di controllo supera il valore 10 (cioè il contatore diventa 11)

Aspetti essenziali dell'iterazione

E' possibile rendere il programma più conciso
inizializzando counter a 0 e sostituendo l'istruzione
`while` precedente con

```
while (++counter <= 10 ) {  
    printf( "%u\n", counter );  
}
```

Aspetti essenziali dell'iterazione

Questo codice risparmia un'istruzione, perché l'incremento è fatto direttamente nella condizione `while` prima che la condizione sia verificata

```
while (++counter <= 10 ) {  
    printf( "%u\n", counter );  
}
```

Codificare in un modo così conciso richiede molta pratica

Alcuni programmatori pensano che questo renda il codice troppo criptico e soggetto a errori

Aspetti essenziali dell'iterazione

Errore comune di programmazione



I valori in virgola mobile possono essere approssimati

Il controllo con contatore dei cicli con variabili in virgola mobile può portare a valori del contatore e a test di chiusura imprecisi



Controllate i cicli con contatore con valori interi

Aspetti essenziali dell'iterazione

Buona pratica di programmazione



Troppi livelli di annidamento possono rendere un programma difficile da comprendere

Di regola, cercate di evitare l'uso di più di tre livelli di annidamento

Aspetti essenziali dell'iterazione

Buona pratica di programmazione



La spaziatura verticale prima e dopo le istruzioni di controllo e l'indentazione dei corpi delle istruzioni di controllo rispetto alle intestazioni di queste ultime conferiscono ai programmi un'apparenza bidimensionale che ne migliora moltissimo la leggibilità

Esercizi con operatori

Scrivete quattro differenti istruzioni in C che sommino
1 alla variabile intera x

Esercizi con operatori

Scrivete quattro differenti istruzioni in C che sommino 1 alla variabile intera `x`

```
x = x + 1;  
x += 1;  
++x;  
x++;
```

Scrivete singole istruzioni in C per:

- a) Acquisire il valore della variabile intera senza segno \times
- b) Scrivere il valore della variabile senza segno \times decrementato di 1

Scrivete singole istruzioni in C per:

- a) Acquisire il valore della variabile intera senza segno `x`
- b) Scrivere il valore della variabile senza segno `x` decrementato di 1

```
scanf( "%u", &x );  
  
printf( "%u\n", --x );
```

Esercizi con operatori

Determinate l'output del seguente programma:

```
int x = 4, y = 5;

if ( ++x >= 5 ) {
    if (y-- >= 5)
        puts ( "*" );
    else
        puts ( "+" );
        puts ( "#" );
}
```

Esercizi con operatori

Determinate l'output del seguente programma:

```
int x = 4, y = 5;

if ( ++x >= 5 ) {
    if (y-- >= 5)
        puts ( "*" );
    else
        puts( "+" );
    puts( "#" );
}
```

```
*
#
```

Esercizio con operatori di uguaglianza:

```
int main( void ) {  
  
    int x = 4, y = 4;  
  
    printf ( "%d\n", x==y) ;  
    printf ( "%d\n", x!=y) ;  
    x++;  
    printf ( "%d\n", x==y) ;  
    printf ( "%d\n", x!=y) ;  
}
```

Esercizi con operatori

Esercizio con operatori di uguaglianza:

```
int main( void ) {  
  
    int x = 4, y = 4;  
  
    printf ( "%d\n", x==y) ;  
    printf ( "%d\n", x!=y) ;  
    x++;  
    printf ( "%d\n", x==y) ;  
    printf ( "%d\n", x!=y) ;  
}
```

1
0
0
1

Esercizi con iterazioni `while`

Scrivete un programma in C che calcoli x elevato alla potenza y

Il programma dovrà avere un'istruzione di controllo dell'iterazione `while`

Esercizi con iterazioni **while**

```
// eleva x alla potenza y
#include <stdio.h>

int main( void )
{
    printf( "%s", "First integer: " );
    unsigned int x;
    scanf( "%u", &x ); // leggi x

    printf( "%s", "Second integer: " );
    unsigned int y;
    scanf( "%u", &y ); // leggi y
```

Esercizi con iterazioni **while**

```
unsigned int i = 1;
unsigned int power = 1;

while ( i <= y ) {
    power *= x;
    ++i;
} // fine di while

printf( "%u\n", power );
} // fine della funzione main
```

Esercizi con iterazioni **while**

Che cosa stampa il seguente programma?

```
#include <stdio.h>

int main( void )
{
    unsigned int c = 1;
    while (c <= 10 ) {
        puts( c % 2 ? "*" : "+" );
        c++;
    }
}
```

Esercizi con iterazioni **while**

Che cosa stampa il seguente programma?

```
#include <stdio.h>

int main( void )
{
    unsigned int c = 1;
    while (c <= 10 ) {
        puts( c % 2 ? "*" : "+" );
        c++;
    }
}
```


*
+
*
+
*
+
*
+
*
+

Esercizi con iterazioni **while**

Che cosa stampa questa versione?

```
#include <stdio.h>

int main( void )
{
    unsigned int c = 1;
    while (c <= 10 ) {
        puts( c % 2 + 1 ? "*" : "+" );
        c++;
    }
}
```



Esercizi con iterazioni **while**

Che cosa stampa questa versione?

```
#include <stdio.h>

int main( void )
{
    unsigned int c = 1;
    while (c <= 10 ) {
        puts( c % 2 + 1 ? "*" : "+" );
        c++;
    }
}
```

*
*
*
*
*
*
*
*
*
*

Mai zero

Esercizi con iterazioni `while`

Scrivete un programma che stampi le tabelline

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Esercizi con iterazioni **while**

Scrivete un programma che stampi le tabelline

```
unsigned int x = 1;
while (x <= 10 ) {
    unsigned int y = 1;
    while (y <= 10 ) {
        printf( "%d\t", x*y );
        y++;
    }
    printf( "\n");
    x++;
}
```