UNIVERSITÀ
degli STUDI
di CATANIA

Costrutti di ciclo in C++

Corso di programmazione I AA 2019/20

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella
Web: http://www.dmi.unict.it/farinella
Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

```
while (condition)
    Statement
```

Statement rappresenta una **singola istruzione** o un **blocco di istruzioni**.

```
while (condition)
    Statement
```

`Condition` è una espressione che **produce un valore di verità**.

Essa viene **valutata prima di ogni eventuale iterazione**.

```
while (condition)
    Statement
```

Se il risultato della valutazione della espressione **condition** è uguale o equivalente a **true**, allora viene eseguita una nuova iterazione.

```
while (condition)
    Statement
```

Se e quando la espressione **condition** sarà valutata **false**, allora il flusso di esecuzione seguirà la prima istruzione collocata dopo l'intero costrutto while. In altre parole, il **loop termina**.

Il costrutto `while` in C++. Esempio E11.1

```
1  const double TARGET = 1800.0;
2  const double TASSO_INTERESSE = 0.1;
3  double capitale=1000.0;
4  int anno=0;
5
6  while(capitale<TARGET){ // condizione
7     capitale+=capitale*TASSO_INTERESSE;
8     anno++;
9  }
```

Uso tipico: il numero di iterazioni **non è noto a priori**.

Il costrutto `while` in C++. Esempio E11.2

```
1  const double TASSO_INTERESSE = 0.1;
2  const int N = 5;
3  double capitale=1000.0;
4
5  int anno=0; // inizializzazione
6  while(anno<N){ // condizione
7     capitale+=capitale*TASSO_INTERESSE;
8     anno++; // incremento/aggiornamento
9  }
```

Numero di iterazioni noto a priori (N iterazioni).

Il costrutto `while` in C++. Esempio E11.3

```
1  const double TASSO_INTERESSE = 0.1;
2  const int N = 5;
3  double capitale=1000.0;
4
5  int anno=0;
6  while(anno++<N)
7     capitale+=capitale*TASSO_INTERESSE;
```

Equivalente ad esempio precedente?

Il costrutto `while` in C++. Esempio E11.4

```
1  const double TASSO_INTERESSE = 0.1;
2  const int N = 5;
3  double capitale=1000.0;
4
5  int anno=0;
6  while(++anno<N)
7     capitale+=capitale*TASSO_INTERESSE;
```

In questo caso quante iterazioni?

Il costrutto `while` in C++. Esempio E11.5

```
1  const double TASSO_INTERESSE = 0.1;
2  const int N = 5;
3  double capitale=1000.0;
4
5  int anno=0;
6  while(anno<N);{
7     capitale+=capitale*TASSO_INTERESSE;
8     anno++;
9  }
```

Corretto?

Il costrutto `while` in C++. Esempio E11.6

```
1  const int N = 5;
2  double capitale=1000.0;
3  const double TASSO_INTERESSE = 0.1;
4
5  int anno=0;
6  while(anno<N){
7     capitale+=capitale*TASSO_INTERESSE;
8  }
```

Corretto? (No..loop infinito! Perchè?)

Il costrutto `while` in C++. Esempio E11.7

```
1  const int N = 5;
2  double capitale=1000.0;
3  const double TASSO_INTERESSE = 0.1;
4
5  int anno=0;
6  while(anno<N){
7     capitale+=capitale*TASSO_INTERESSE;
8     anno--;
9  }
```

Corretto?

Sulla base degli esempi precedenti, codificare un ciclo while in linguaggio C++ nel quale:

- il loop si ferma quando il capitale o montante raggiunge o supera la cifra target definita in una costante T;
- tuttavia il periodo di accumulo di interessi sul montante non deve in ogni caso superare il numero di anni definito in una costante N.

Eseguire Hand Tracing del ciclo while codificato nell'esercizio precedente con i seguenti valori (in tutti casi deve essere tasso interesse 10% e capitale iniziale 1000 euro):

1. N=5, TARGET=1500
2. N=3, TARGET=1200
3. N=10, TARGET=1600

Il costrutto while. Homework H11.3

Codificare un programma completo in linguaggio C++ in cui:

- l'utente deve inserire da tastiera il capitale iniziale (C), il tasso di interesse (TI), il target (T) e numero di anni (N);
- se uno tra T ed N è un numero minore o uguale a zero, allora il programma non dovrà tener conto di tale parametro; nel caso in cui sia T che N siano minori o uguali a zero allora il programma terminerà con un messaggio di errore;

Il costrutto while. Homework H11.3

- il programma darà in output il capitale finale ed il numero totale di anni di accumulo; come nello esercizio precedente, l'accumulo degli interessi sul montante si interrompe quando il capitale raggiunge o supera la cifra target T e comunque il numero di anni di accumulo non deve superare N.

```
for(Initialization; Condition; Update)
   Statement
```

`Statement` rappresenta una **singola istruzione** o un **blocco di istruzioni**.

```
for(Initialization; Condition; Update)
   Statement
```

`Initialization` rappresenta una istruzione di inizializzazione

Essa viene eseguita **solo una volta prima dell'inizio del ciclo.**

```
for(Initialization; Condition; Update)
   Statement
```

Condition rappresenta una espressione che produce un risultato uguale o equivalente a **true** o **false**, come per il costrutto `while`.

Essa viene valutata **prima di ogni iterazione**.

```
for(Initialization; Condition; Update)
   Statement
```

`Update` è una istruzione finalizzata ad aggiornare una o più variabili. Viene eseguita **dopo ogni iterazione**.

```
for(Initialization; Condition; Update)
    Statement
```

Se e quando il risultato della valutazione della espressione **condition** sarà uguale o equivalente a **false**, allora il flusso di esecuzione seguirà la prima istruzione collocata dopo l'intero costrutto `for`.

In altre parole il **loop termina**.

```
1  const double TASSO_INTERESSE = 0.1;
2  cont int N = 5;
3  double capitale = 1000.0;
4  for(int anno=0; anno<N; anno++)
5     capitale+=capitale*TASSO_INTERESSE;
```

Uso tipico: N iterazioni note a priori

```
1   int anno=0;
2   while(anno<N){
3       capitale+=capitale*TASSO_INTERESSE;
4       anno++;
5   }
```

```
1   for(int anno=0; anno<N; anno++)
2       capitale+=capitale*TASSO_INTERESSE;
```

Più immediata la lettura/comprensione della struttura del ciclo (aggregare inizializzazione, condizione e incremento)

```
1   int anno=0;
2   while(anno<N){
3       capitale+=capitale*TASSO_INTERESSE;
4       anno++;
5   }
```

```
1   for(int anno=0; anno<N; anno++)
2       capitale+=capitale*TASSO_INTERESSE;
```

Scope (portata/visibilità) della variabile anno?

```
1   for(int anno=0; anno<N; anno++)
2     capitale+=capitale*TASSO_INTERESSE;
3
4   //il tasso di interesse e' cambiato!
5   for(int anno=0; anno<N; anno++)
6     capitale+=capitale*TASSO_INTERESSE_2;
```

Scope variabile `anno` limitato al blocco di istruzioni del ciclo $\Rightarrow$ si può usare lo stesso nome in un altra istanza del costrutto `for` senza dover controllare se era già stata dichiarata in un for precedente.

```
1  double capitale=1000.0;
2  while(capitale<TARGET){
3     capitale+=capitale*TASSO_INTERESSE;
4  }
```

```
1  double capitale =1000.0;
2  for(; capitale<TARGET;){
3     capitale+=capitale*TASSO_INTERESSE;
4  }
```

`for` poco adatto (codice poco leggibile) quando il numero di iterazioni non è fissato...

```
1  double capitale=1000.0;
2  while(capitale<TARGET){
3     capitale+=capitale*TASSO_INTERESSE;
4  }
```

```
1  double capitale =1000.0;
2  for(; capitale<TARGET; \
3           capitale+=capitale*TASSO_INTERESSE);
```

NB: Istruzione di aggiornamento del capitale!

```
1  double capitale=1000.0;
2  while(capitale<TARGET){
3     capitale+=capitale*TASSO_INTERESSE;
4  }
```

```
1  double capitale;
2  for(capitale=1000.0; capitale<TARGET; \
3              capitale+=capitale*TASSO_INTERESSE);
```

NB: Inizializzazione del capitale!

for vs while

```
1  double capitale=1000.0;
2  int anno=0;
3  while(capitale<TARGET){
4     capitale+=capitale*TASSO_INTERESSE;
5     anno++;
6  }
```

```
1  double capitale=1000.0; int anno;
2  for(anno=0; capitale<TARGET; anno++)
3     capitale+=capitale*TASSO_INTERESSE;
```

Condizione su TARGET e incremento variabile anno.

NB: Ma le tre espressioni non sono del tutto correlate..

```
1   float y=0.1;
2   for(y=0.1; y!=0.8; y+=0.1)
3       cout << y << endl;
```

Posso usare i numeri in virgola mobile per controllare un ciclo?

(Attenzione agli errori di approssimazione o rappresentazione!..)

Il costrutto `do/while` in C++. Definizione

```
do{
    Statement;
}while(Condition);
```

Identico al costrutto `while`, ma **la prima iterazione viene eseguita incondizionatamente!!**

Usi del costrutto do/while.

```
1  int input;
2  do{
3     cout << "Inserire un numero minore di 100 \
4       e maggiore o uguale a 50: " << endl;
5     cin >> input;
6  }while(input >= 100 || input <50);
```

Uso tipico. La prima iterazione differisce dalle altre, che sono eseguite a causa di una particolare condizione (utente inserisce numero non valido) quindi a seguito di una condizione.

Contare le iterazioni in un ciclo

a e b numeri interi, con $a < b$.

b − a iterazioni

```
1   for(int i=a; i<b; i++)
2       //do something
```

b − a + 1 iterazioni!

```
1   for(int i=a; i<=b; i++)
2       //do something
```

Homework

H11.4

Codificare in linguaggio C++ un algoritmo che produca la somma dei numeri interi dispari da 1 a 99, facendo uso del costrutto for. Dalla somma vanno esclusi i numeri divisibili per tre.

H11.5

Codificare in linguaggio C++ un algoritmo che stampi a ritroso i numeri pari minori o uguali a mille; Il loop deve comunque terminare se la somma dei numeri precedentemente stampati è maggiore o uguale a centomila.

Homework

H11.6

Codificare in linguaggio C++ un algoritmo che stampi a ritroso la sequenza di caratteri da 'a' a 'z', ma non le vocali. Codificare due versioni differenti nelle quali si fa uso del costrutto for e del costrutto while rispettivamente.

H11.7

Estendere l'esercizio precedente in modo da conservare tutti i caratteri stampati in un oggetto `stringstream`. Infine, stampare la lunghezza della stringa contenuta in esso.

Homework

H11.8

Codificare in linguaggio C++ un algoritmo che stampi le prime N potenze di 2, dove N è un parametro scelto dall'utente (input da tastiera).

Homework

H11.9

Codificare in C++ un programma che chiede all'utente di inserire due numeri decimali maggiori di zero e diversi tra loro (ES: 10.2 e 24.7), ed un numero intero N.

- il programma calcola lo arrotondamento per eccesso o per difetto di entrambi i numeri, a seconda che la parte decimale sia maggiore o uguale a 0.5 oppure minore di 0.5 rispettivamente. Siano a e b i due numeri ottenuti, con $a < b$.
- per ogni numero $a \leq p \leq b$, calcola e stampa la somma degli N-1 numeri minori di p e la somma dei $2N$ numeri maggiori di p;

La istruzione `break` interrompe il flusso di controllo all'interno di un costrutto `switch`.

> Inoltre, se inserita all'interno di un ciclo `for` o `while` oppure ancora `do/while`, **interrompe l'esecuzione del ciclo nel punto esatto in cui è presente la istruzione**.

NB: mai indispensabile! **Esiste sempre una forma di controllo iterativo equivalente** senza break.

```
1  double capitale=1000.0;
2  int anno=0;
3  while(true){
4     capitale+=capitale*TASSO_INTERESSE;
5     anno++;
6     if(capitale>=TARGET)
7        break;
8  }
```

Potrebbe peggiorare la comprensibilità della struttura del ciclo.

```
1  double capitale=1000.0;
2  for(int anno=0; anno<N; anno++){
3     capitale+=capitale*TASSO_INTERESSE;
4     if(capitale>TARGET)
5        break;
6  }
```

Potrebbe peggiorare la comprensibilità della struttura del ciclo.

L'istruzione `continue`, se inserita all'interno di un ciclo, **salta le istruzioni restanti della iterazione** per iniziare un altro ciclo.

```
1  const int N = 20;
2  //stampa solo i numeri dispari
3  for(int i=0; i<N; i++){
4       if(i%2==0)
5          continue;
6       cout << i << endl;
7  }
```

```
1  const int N = 20;
2  //stampa solo i numeri dispari
3  for(int i=0; i<N; i++){
4       if(i%2==0)
5          continue;
6       cout << i << endl;
7  }
```

NB: mai indispensabile! **Esiste sempre una forma di controllo iterativo equivalente** senza continue.

Esempi svolti

```
01_while_vs_for.cpp
02_doublinv.cpp
03_doublinv_whilefor.cpp
03_invtable_modified_for_years.cpp
04_loop_vars.cpp
05_fence.cpp
06_io.cpp
06_rafter.cpp
```

Esempi svolti

Esempi svolti

07_2_loop_and_half_B.cpp

07_2_loop_and_half.cpp

07_loop_and_half.cpp

07_sentinel.cpp

08_square_matrix.cpp

09_random.cpp

10_dice.cpp

11_powtable.cpp

12_montecarlo.cpp

Esempi svolti

B4_07.cpp

B4_08.cpp

B4_09.cpp

B4_10.cpp

B4_11.cpp

B4_12.cpp

B4_13.cpp

FINE