Le strutture di controllo in C++

Docente: Ing. Edoardo Fusella
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Via Claudio 21, 4° piano – laboratorio SECLAB
Università degli Studi di Napoli Federico II
e-mail: edoardo.fusella@unina.it

Strutture di controllo iterative

Con una struttura di controllo di tipo iterativo si determina la ripetizione dell'esecuzione di un blocco di istruzioni per un numero di volte prefissato o fino al fallimento di una condizione.

- *while*
- *do… while*
- *for*

Il ciclo while

Il ciclo **while** impone che l'esecuzione del blocco di istruzioni sia ripetuta fino a quando la condizione non diventa FALSE.

```
while    (condizione)
        { S;
        }
```

Inizialmente viene valutata la condizione:

- se è FALSE, la sequenza S non viene eseguita;
- Se è TRUE, si esegue S e al suo termine si ricalcola la condizione e si riesegue S se la condizione è ancora VERA
- **Assicurarsi che ci sia una condizione di terminazione!**

Esempio while

- ***cond*** è la variabile booleana di controllo del ciclo che viene inizializzata a true
- Le istruzioni contenute nel corpo del ciclo vengono eseguite finché è vera la condizione ***cond***
- Ad ogni iterazione del ciclo (ogni ripetizione del corpo del ciclo) la variabile di condizione può diventare falsa
- Quando la condizione diventa falsa, si esce dal ciclo

```
1. bool cond = true;
2. while( cond ){
3.     …
4.     …
5.     …
6. }
```

Esempio while 8.1

- Somma di numeri interi positivi. Il programma finisce al primo numero negativo

```
1.  #include<iostream>
2.  #include<cstdlib>
3.  using namespace std;
4.  int main(){
5.    int i=0, somma=0;
6.    bool condizione=true;
7.    while (condizione){
8.      cout << "Inserisci un numero intero: \n";
9.      cin >> i;
10.     if (i < 0){
11.       condizione=false;
12.     }
13.     else{
14.       somma += i;
15.     }
16.   }
17.   cout << "somma: "<<somma<<" \n";
18.   return 0; }
```

Ciclo infinito

- Il while(true) è un ciclo infinito in quanto la condizione è sempre vera.

```
1. while( true ){
2.     …
3.     …
4.     …
5. }
```

Il ciclo do..while

Con il ***do while*** la condizione viene scritta dopo la sequenza S per ottenere che l'esecuzione del blocco avvenga almeno una volta.

```
do
    { S;
    }
    while (condizione)
```

Inizialmente si esegue il blocco di istruzioni e poi viene valutata la condizione:

- se è FALSE, la sequenza S non viene rieseguita;
- Se è TRUE, si riesegue S e al suo termine si ricalcola la condizione
- **Assicurarsi che ci sia una condizione di terminazione!**

Esempio do..while

- ***cond*** è la variabile booleana di controllo del ciclo che viene inizializzata a true
- L'esecuzione avviene almeno una volta anche se la condizione non è verificata
- Le istruzioni contenute nel corpo del ciclo vengono eseguite finché è vera la condizione ***cond***
- Ad ogni iterazione del ciclo (ogni ripetizione del corpo del ciclo) la variabile di condizione può diventare falsa
- Quando la condizione diventa falsa, si esce dal ciclo

```
1. bool cond = true;
2. do {
3.     …
4.     …
5.     …
6. } while( cond );
```

Esempio do..while 8.2

- Somma di numeri interi positivi. Il programma finisce al primo numero negativo

```
1.  #include<iostream>
2.  #include<cstdlib>
3.  using namespace std;
4.  int main(){
5.    int i=0, somma=0;
6.    bool condizione=false;
7.    do{
8.      cout << "Inserisci un numero intero: \n";
9.      cin >> i;
10.     if (i < 0){
11.       condizione=false;
12.     }
13.     else{
14.       condizione=true;
15.       somma += i;
16.     }
17.   } while (condizione);
18.   cout << "somma: "<<somma<<" \n";
19.   return 0;
20. }
```

Osservazione:

La condizione iniziale può essere inizializzata sia a true che a false in quanto si entra nel ciclo almeno una volta.

Esempio do..while 8.3

- Somma dei numeri inseriti dall'utente. Ad ogni iterazione l'utente decide se terminare l'esecuzione

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5.    int i=0, somma=0;
6.    bool condizione=true;
7.    do{
8.       cout << "Inserisci un numero : \n";
9.       cin >> i;
10.      somma += i;
11.      cout << "Terminare (0=si,1=no): \n";
12.      cin >> condizione;
13.   } while (condizione);
14.   cout << "somma: "<<somma<<" \n";
15.   return 0;
16. }
```

Il ciclo for

Il ciclo ***for*** opera come il while con una iterazione della sequenza che continua quando la condizione espressa risulta vera: appena la condizione è falsa la iterazione termina.

```
for (inizializzazioni; condizione; variazioni)
                        { S;
                         }
```

Il ciclo for prescrive:

1. l'esecuzione delle istruzioni indicate come *inizializzazione* (non obbligatorie);
2. il calcolo della *condizione*;
3. la esecuzione della sequenza S se risulta verificata la condizione (assume valore TRUE); in caso contrario l'iterazione termina;
4. l'esecuzione delle istruzioni di *variazione* (non obbligatorie) al termine di S;
5. la rivalutazione della condizione con il ripetersi dei passi precedenti fino alla determinazione della falsità della condizione.

Esempio for crescente

```
1. for(int i = 0; i<10; i++ ){
2.     …
3.     …
4.     …
5. }
```

- ***i*** è la variabile contatore di ciclo che viene inizializzata a ***0***
- Le istruzioni contenute nel corpo del ciclo vengono eseguite finché è vera la condizione **i<10**
- Ad ogni iterazione del ciclo (ogni ripetizione del corpo del ciclo) la variabile contatore viene incrementata
- Alla prima iterazione i=0, alla seconda i=1, alla terza i=2, … , all'ultima i=9.

Esempio for decrescente

```
1. for(int i = n; i>0; i-- ){
2.     …
3.     …
4.     …
5. }
```

- ***i*** è la variabile contatore di ciclo che viene inizializzata a ***n***
- Le istruzioni contenute nel corpo del ciclo vengono eseguite finché è vera la condizione **i>0**
- Ad ogni iterazione del ciclo (ogni ripetizione del corpo del ciclo) la variabile contatore viene decrementata
- Alla prima iterazione i=n, alla seconda i=n-1, alla terza i=n-2, … , all'ultima i=1.

Esempio for 8.4

- Somma dei primi n numeri interi (for crescente)

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5.    int n=0, somma=0;
6.    cout << “Quanti numeri: \n";
7.    cin >> n;
8.    for(int i=0;i<=n;i++){
9.       somma += i;
10.     }
11.   cout << "somma: "<<somma<<" \n";
12.   return 0;
13.  }
```

Esempio for 8.5

- Somma dei primi n numeri interi (for decrescente)

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5.    int n=0, somma=0;
6.    cout << “Quanti numeri: \n";
7.    cin >> n;
8.    for(;n>0; n--){
9.       somma += n;
10.     }
11.   cout << "somma: "<<somma<<" \n";
12.   return 0;
13.  }
```

Equivalente a:

```
for(int i=n;i>0;i--){
      somma+=i;
}
```

Attenzione se si usa la variabile i, la somma va fatta con *i* e non *n*

Le istruzioni goto, break e continue

Il linguaggio prevede tre istruzioni di salto che vanno usate in casi eccezionali quando il tempo di esecuzione è particolarmente critico:

- **break;** *comporta l'uscita da un while, do-while, for e dallo switch.*
- **continue;** *può essere usata solo all'interno di un ciclo while, do-while e for. La sua esecuzione comporta il passaggio alla iterazione successiva.*
- **goto** <label>; *salta all'istruzione avente etichetta <label>*
  - L'uso dell'istruzione goto è sconsigliato perché il flusso di esecuzione dei programmi diventa poco strutturato

Esempio break 8.6

- Somma di numeri interi positivi. Il programma termina al primo numero negativo

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5. int i=0, somma=0;
6. while (true){
7. cout << "Inserisci un numero: \n";
8. cin >> i;
9.    if (i < 0){
10.      break;
11.   }
12.   somma += i;
13.}
14.return 0;}
```

Esempio continue 8.7

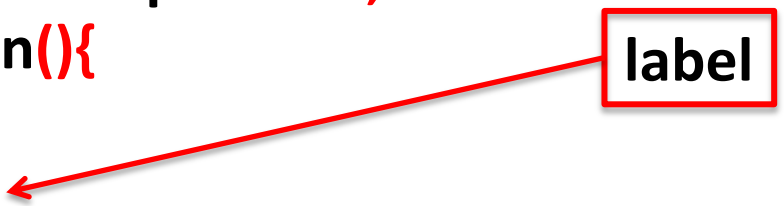
- Somma di 5 numeri inseriti dall'utente. I numeri negativi non vengono sommati.

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5. int i=0, somma=0;
6. for(int j=0; j<5; j++){
7. cout << "Inserisci un numero: \n";
8. cin >> i;
9.    if (i < 0){
10.      continue;
11.   }
12.   somma += i;
13.}
14.return 0;}
```

Esempio goto 8.8

- Stampa a video i primi 5 numeri interi

```
1. #include<iostream>
2. #include<cstdlib>
3. using namespace std;
4. int main(){
5. int i=1;
6. inizio:
7. cout << "i: " << i << "\n";
8. i++;
9. if(i==6){
10.       return 0;
11. }
12. goto inizio;
13.return 0;
14.}
```

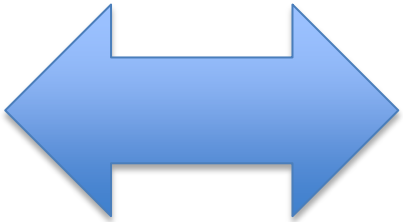


Sconsigliato, si effettua un salto verso un qualsiasi punto del programma senza effettuare alcun controllo

Condizioni booleane

- Le due notazioni sono equivalenti.
  - La condizione booleana può essere salvata in una variabile
  - Valido per tutti i costrutti di controllo (if, while, for ,ecc..)

```
1. int x=0;
2. cout<<“numero: “;
3. cin>>x;
4. while( x==1 ){
5.     …
6.     …
7.     …
8. }
```



```
1. int x=0;
2. cout<<“numero: “;
3. cin>>x;
4. bool cond = x==1;
5. while( cond ){
6.     …
7.     …
8.     …
9. }
```

Esercizio 1

Scrivere un programma che, dati due valori numerici di tipo reale inseriti dall'utente *a* e *b*, effettui somma, sottrazione, moltiplicazione e divisione (esercizio 7.3)

Oltre ad inserire i due valori e la scelta, l'utente deve anche specificare quando chiudere il programma. Si indichi con *stop* la variabile booleana atta a indicare quando terminare l'esecuzione del programma.