



Università degli Studi
di Cassino

Corso di Fondamenti di Informatica

Classi di istruzioni 2

Anno Accademico 2008/2009

Francesco Tortorella

Strutture di controllo

- Caratteristica essenziale degli algoritmi è la possibilità di decidere le operazioni da applicare durante la loro esecuzione in base allo stato dell'esecuzione stessa.
- I meccanismi atti a controllare la sequenza delle operazioni da applicare per l'esecuzione di un algoritmo vengono denominate **strutture di controllo**.

1. Leggi due numeri X e Y , con $X > Y$
2. Dividi X per Y e ottieni il resto R
3. **Se $R=0$, termina: il MCD è Y**
4. Sostituisci X con Y
5. Sostituisci Y con R
6. **Torna al punto 2.**

Strutture di controllo

- Ogni algoritmo può essere implementato in un linguaggio di programmazione che combina solo tre tipi di strutture:
 - **Sequenza**
 - **Selezione**
 - **Ciclo**
- Tipicamente i linguaggi offrono più di un costrutto per ogni tipo di struttura per rendere più agevole la codifica degli algoritmi

Sequenza

- La sequenza è costituita da un insieme di istruzioni successive che vengono eseguite nell'ordine in cui compaiono nel testo.
- Un particolare caso di sequenza in C++ è il **blocco** (o *istruzione composta*, *compound statement*), formato da un insieme di istruzioni tra { }.

Costrutti di selezione

- Permettono di scegliere di eseguire una tra due istruzioni alternative in base alla valutazione di una condizione

Costrutti di selezione: if

- Sintassi

```
if (condizione)  
    istruzione
```

L'*istruzione* è eseguita solo se *condizione* è true

L'*istruzione* può essere costituita da un blocco.

Esempi

- Calcolo del valore assoluto di un numero dato in input
- Verificare che due valori X e Y forniti in input rispettino la condizione $X \geq Y$.

Costrutti di selezione: if...else

- Sintassi

```
if (condizione)  
    istruzione_1
```

```
else  
    istruzione_2
```

istruzione_1
eseguita se
condizione è vera

istruzione_2
eseguite se
condizione è falsa

Esempio: max fra due

```
# include <iostream>
using namespace std;

int main()
{
    int x,y,max;

    cout << "Primo valore: ";
    cin >> x;
    cout << "Secondo valore: ";
    cin >> y;

    if(x>y)
        max=x;
    else
        max=y;

    cout << "Il massimo e': " << max << endl;
}
```

Esempio

- Soluzione di un sistema di due equazioni lineari in due incognite
 - Versione 3: verifica se il determinante è nullo

Esempio: max fra tre

```
# include <iostream>
using namespace std;

int main()
{
    int x,y,z,max;

    cout << "Primo valore: ";
    cin >> x;
    cout << "Secondo valore: ";
    cin >> y;
    cout << "Terzo valore: ";
    cin >> z;

    max=x;

    if(y>max)
        max=y;

    if(z>max)
        max=z;

    cout << "Il massimo e': " << max << endl;
}
```

Costrutti di selezione: if...else if ... else

- Sintassi

```
if (condizione_1)  
    istruzione_1  
else if (condizione_2)  
    istruzione_2  
else if (condizione_3)  
    istruzione_3  
else  
    istruzione_4
```

eseguita solo se *condizione_1* è vera

eseguito solo se *condizione_1* è falsa e *condizione_2* è vera

eseguito solo se *condizione_1* è falsa, *condizione_2* è falsa e *condizione_3* è vera

eseguito solo se *condizione_1* è falsa, *condizione_2* è falsa e *condizione_3* è falsa

Esempio

```
# include <iostream>
using namespace std;

int main()
{
    int voto;

    cout << "Voto ricevuto: ";
    cin >> voto;

    if (voto < 18)
        cout << "Ritorna\n";
    else if (voto < 24)
        cout << "Si puo' dare di piu'! \n";
    else if (voto < 27)
        cout << "Non c'e' male !\n";
    else if (voto == 30)
        cout << "Finalmente ci siamo ! \n";
    else
        cout << "WOW !!\n";
}
```

Costrutti di selezione: switch

- Sintassi

```
switch(espr_sw) {  
  case espr_1:  
    istruzione_1_1  
    ...  
    istruzione_1_m  
    break;  
  case espr_2:  
    istruzione_2_1  
    ...  
    istruzione_2_n  
    break;  
  case espr_3: case espr_4:  
    istruzione_3_1  
    ...  
    istruzione_3_t  
    break;  
  default:  
    istruzione_3_1  
    ...  
    istruzione_3_t  
    break;  
}
```

← eseguite se $espr_sw == espr_1$

← eseguite se $espr_sw \neq espr_1$
e $espr_sw == espr_2$

← eseguite se $espr_sw \neq espr_1$,
 $espr_sw \neq espr_2$ e
 $espr_sw == espr_3$ o
 $espr_sw == espr_4$

← eseguite se $espr_sw$ è diverso
da tutte le $espr_i$ nei **case**

Esempio

```
switch(mese) {  
    case 2:  
        ngiorni=28;  
        break;  
    case 4: case 6: case 9: case 11:  
        ngiorni=30;  
        break;  
    default:  
        ngiorni=31;  
}
```

Esempio

```
switch(car){
  case '.' :
    cout << "punto\n";
    break;
  case ',' :
    cout << "virgola\n";
    break;
  case 'a' : case 'e' : case 'i' :
  case 'o' : case 'u' :
    cout << "vocale\n";
    break;
  default:
    cout << "consonante\n";
}
```


Problema

- Scrivere un programma che legga da input i coefficienti a , b , c di un'equazione di secondo grado e ne calcoli le radici.
- Considerare i casi in cui uno o più dei coefficienti sia nullo.

Costrutti di ciclo

- Servono a ripetere l'esecuzione di un'istruzione
- A seconda di come viene definito il numero di ripetizioni dell'esecuzione, si distinguono in
 - Costrutti di ciclo a condizione
 - Costrutti di ciclo a conteggio

Costrutti di ciclo: while

- E' un costrutto di ciclo *a condizione*
- Non si definisce esplicitamente il numero di ripetizioni dell'esecuzione, ma si valuta all'inizio del ciclo un'espressione logica che, fin quando risulta vera, causa un'ulteriore esecuzione dell'istruzione.

Costrutti di ciclo: while

- Sintassi

```
while(condizione)  
    istruzione
```

- Si valuta la *condizione*
- Se risulta vera, si esegue l'istruzione e quindi si torna a verificare la condizione
- Se la condizione risulta falsa, si passa a eseguire le istruzioni che si trovano dopo la chiusura del while
- Qual è il minor numero di cicli che si può effettuare ?

Esempio

- Stampare in output i primi 10 numeri naturali.

Esempio

- Stampare in output i primi 10 numeri naturali.

```
#include <iostream>
using namespace std;
int main () {
    int x;

    x=1;
    while(x<=10){
        cout << x << endl;
        x++;
    }
}
```

Problema

- Leggere da input un insieme di numeri interi e calcolarne la somma. Non si conosce in anticipo la quantità di valori da leggere; la lettura di un valore `== 0` indica che l'insieme da leggere è terminato.

Problema

Leggere da input un insieme di numeri reali e calcolarne la media. Non si conosce in anticipo la quantità di valori da leggere, che comunque è limitata ad un massimo di 50; la lettura di un valore < 0 indica che l'insieme da leggere è terminato.

Problema

ricerca del minimo e del massimo

- **Problema 1**

Leggere da input un insieme di numeri reali ≥ 0 e determinare il valore minimo. Non si conosce in anticipo la quantità di valori da leggere; la lettura di un valore < 0 indica che l'insieme da leggere è terminato.

- **Problema 2**

Nelle stesse ipotesi del problema 1, determinare il valore massimo dell'insieme dei valori letti.

Problema: calcolo del MCD

1. Leggi due numeri X e Y , con $X > Y$
2. Dividi X per Y e ottieni il resto R
3. Se $R=0$, termina: il MCD è Y
4. Sostituisci X con Y
5. Sostituisci Y con R
6. Torna al punto 2.

ACHTUNG !!!

Non è un ciclo WHILE

Come fare ?

Costrutti di ciclo: do while

- E' un costrutto di ciclo *a condizione*
- Non si definisce esplicitamente il numero di ripetizioni dell'esecuzione, ma si valuta al termine del ciclo un'espressione logica che, fin quando risulta vera, causa un'ulteriore esecuzione dell'istruzione.

Costrutti di ciclo: do while

- Sintassi

do

istruzione

while(*condizione*)

- Si esegue l'*istruzione*
- Si valuta la *condizione*
- Se risulta vera si torna a eseguire l'*istruzione*
- Se la condizione risulta falsa, si passa a eseguire le istruzioni che si trovano dopo la chiusura del while
- Qual è il minor numero di cicli che si può effettuare ?

Costrutti di ciclo : for

- E' un costrutto di ciclo *a conteggio*
- Si definisce esplicitamente il numero di ripetizioni dell'esecuzione
- Il conteggio viene gestito grazie ad una *variabile (variabile di conteggio)* che assume un valore iniziale e viene incrementata di un valore fisso ad ogni ripetizione del ciclo finché non raggiunge o supera un valore finale.

Istruzioni cicliche: for

Sintassi

```
for (initialization; condition; increase)  
    istruzione
```

- Si esegue *initialization*
- Si verifica se *condition* è true
- In caso positivo si esegue l'istruzione sotto il ciclo for; al termine dell'esecuzione, si esegue *increase* e si torna a valutare *condition*
- Se *condition* è false, il ciclo termina e si eseguono le istruzioni che seguono il for

Esempio

- Stampare in output i primi 10 numeri naturali.

Esempio

- Stampare in output i primi 10 numeri naturali.

```
for (x=1;x<=10; x++)  
    cout<<"x: " <<x<<endl;
```

```
x=1;  
while(x<=10){  
    cout<<"x: " <<x<< endl;  
    x++;  
}
```


Esempio

- Stampare in output i primi 100 numeri dispari.

Problema

- Leggere da input un insieme di numeri interi e calcolarne la somma. Il numero di valori da leggere è fornito in ingresso prima della sequenza di valori

Problema

- Stampare la “tabellina” di n , dove n è dato in input
- Stampare le “tabelline” dei valori compresi tra 1 e 10.

Problema

- Realizzare un programma che, letti due valori x ed n da input, calcoli x^n .
- n può essere negativo o nullo