Strutture di controllo

Sequenza – Selezione – Iterazione in C

Dott.ssa Veronica Rossano
rossano@di.uniba.it
http://www.di.uniba.it/~rossano

Le strutture di controllo

- Sequenza
  - Concatenazione di azioni
- Selezione
  - Scelta di azioni alternative
    - Dipendenza da una condizione
- Iterazione
  - Ripetizione di una certa azione
    - Dati potenzialmente diversi
    - Dipendenza da una condizione

Le strutture di controllo in C…

- Sequenza

```
Istruzione1;
Istruzione2;
Istruzione3;
….
IstruzioneN;
```

```
a=6;
b=10;
media=(a+b)/2;
printf("Media tra 6 e 10: %d", media);
```


La selezione…

- La struttura di controllo selezione consente di definire quale blocco di istruzioni eseguire al verificarsi o meno di una condizione

...La selezione

- Selezione

```
if (condizione)
  {
        Sequenza di istruzioni da eseguire se la
        condizione è verificata;
  }
  else
  {
        Sequenza di istruzioni da eseguire se la
        condizione non è verificata;
  }
```

```
if (a>b)
  printf("Il primo numero e' maggiore del secondo");
else
  printf("Il secondo numero e' maggiore del primo");
```


Le condizioni...

- Le condizioni sono espressioni particolari che stabiliscono se eseguire o meno un gruppo di istruzioni
- Gli operatori utilizzabili sono:
  - Operatori relazionali
  - Operatori di uguaglianza
- Le condizioni possono essere operazioni di confronto tra variabili e/o costanti

...Le condizioni...

- La sintassi in C:

```
variabile    operatore relazionale   variabile
variabile    operatore relazionale   costante
variabile    operatore di uguaglianza   variabile
variabile    operatore di uguaglianza   costante
```

- Operatori relazionali
  - < minore di
  - > maggiore di
  - <= minore uguale di
  - >= maggiore uguale di

...Le condizioni

- Operatori di uguaglianza
  - == uguale a
  - != diverso da

```
Esempi di condizioni:
a<b
x <= 0
x>=y
risp=='s'
conta!=10
```

Operatori logici...

- Gli operatori logici:
  - && → AND
  - || → OR
  - ! → NOT

  consentono di creare espressioni logiche più complesse

```
voto>=18 && voto<=30
anni<=10 || anni>=65
!(voto==30)
```


...Operatori logici

- Esempi
  - La temperatura odierna è compresa tra [TempMin, TempMax]
    - (TempMin <=TempOggi) && (TempOggi<=TempMax)
  - Gli studenti che alla prova di laboratorio non hanno ottenuto un voto compreso tra 18 e 30 non sono ammessi all'esame orale
    - !((18<= voto) && (voto<=30))

Ordine di precedenza

- La valutazione di una espressione è determinata dall'ordine di precedenza degli operatori

```
! + -        (operatori unari)        alta
* / %
+ -
< <= >= >
== !=
&&
||
=                                     bassa
```


...Ordine di precedenza...

- Esempi
  - -x - y * z → (-x) - (y*z)
  - x + y < min + max → (x + y) < (min + max)
  - x<y || x<z && x>0 → (x < y) || (x<z && x>0)
- Per cambiare l'ordine delle precedenze è possibile utilizzare le parentesi come nell'algebra

...Ordine di precedenza

- Supponendo che le seguenti variabili siano così avvalorate:
  - x=3.0
  - y=4.0
  - z=2.0
  - flag=0
- Determinare i valori delle seguenti espressioni:
  - !flag
  - x + y / z <= 3.5
  - !flag || (y + z >= x – z)
  - !(flag || (y + z >= x – z))

Short circuit evalutation

Assegnazioni logiche

- L'ordine di precedenza consente di realizzare le assegnazioni logiche a variabili intere che assumono valore 0 se la condizione risulta falsa e 1 altrimenti

```
int eta, anziano;
anziano = (eta > 65);
```

  - La variabile anziano assumerà valore 1 se eta è maggiore di 65 e 0 viceversa

```
int lettera;
char ch;
lettera = ('A' <= ch  &&  ch <='Z' ) || ('a' <= ch  &&  ch <='z' );
```


Confronto tra caratteri

- In C è possibile utilizzare gli operatori relazionali e di uguaglianza anche per i caratteri
  - 'a' < 'b'
    - Vero
  - '9' > '0'
    - Vero
  - 'a' <= car && car <='z'
    - Vero se car è un carattere minuscolo
  - 'a' == 'A'
    - Falso

Selezione ad una alternativa

- È possibile costruire selezioni con un'unica alternativa
- La sequenza di istruzioni è eseguita solo se la condizione è verificata, in caso contrario è ignorata

```
if (x > 0.0)
        prodotto = prodotto * x;
```

Esercizio

- Scrivere un algoritmo e il relativo programma che dati due numeri interi in input restituisca il massimo dei due

Massimo.c

```c
#include <stdio.h>
#include <stdlib.h>

int main ()
{
 int a, b, max;
 printf("Inserire il primo valore ----> ");
 scanf("%d", &a);
 printf("Inserire il secondo valore ----> ");
 scanf("%d", &b);
 if (a>b)
    max=a;
 else
     max=b;
 printf("Il massimo e' %d \n", max);
 system("pause");
 return 0;
}
```


Selezioni innestate

- In un programma le condizioni possono avere più alternative, in questo caso è possibile utilizzare istruzioni di selezione innestate

```c
if (condizione1)
       {
       Sequenza di istruzioni da eseguire se condizione1 è verificata;
       }
 else
       {
         if (condizione2)
         {
            Sequenza di istruzioni da eseguire se condizione2 è verificata;
         }
         else
         {
            Sequenza di istruzioni da eseguire se condizione2 non è verificata;
         }
       }
```


Esempio

- In una serie di numeri calcolare quanti di essi sono positivi, quanti negativi e quanti uguali a zero

```c
if (x > 0)
        numeri_positivi = numeri_positivi + 1;
  else
        if (x < 0)
              numeri_negativi = numeri_negativi + 1;
        else
              numeri_zero = numeri_zero +1;
```

Selezione Multipla (if innestati)

- All'aumentare del numero di alternative aumenta anche il livello di indentazione e la possibilità di commettere errori

```
 if (condizione1)
     {
     Sequenza di istruzioni da eseguire se condizione1 è  verificata;
     }
 else if (condizione2)
     {
     Sequenza di istruzioni da eseguire se condizione2 è  verificata;
     }
…
else if (condizioneN)
     {
     Sequenza di istruzioni da eseguire se condizioneN è verificata;
      }
else
    {
     Sequenza di istruzioni da eseguire se nessuna condizione è
     verificata;
    }
```


…Selezione Multipla (if innestati)

```
if (x > 0)
       numeri_positivi = numeri_positivi + 1;
 else  if (x < 0)
       numeri_negativi = numeri_negativi + 1;
 else
       numeri_zero = numeri_zero +1;
```


Esercizio

CategoriaVento.c

- Scrivere un algoritmo e poi un programma che data in input la velocità del vento (in Km/h) restituisca la categoria di appartenenza secondo la seguente tabella

Velocità del vento (in Km/h)	Classificazione
sotto i 25	Vento debole
25 – 38	Vento forte
39 – 54	Tempesta
55 – 72	Forte tempesta
Oltre i 72	Uragano

Selezione Multipla (switch)

- L'istruzione switch è utilizzata in C per le selezioni che valutano se singole variabili o semplici espressioni assumano valore all'interno di un certo insieme di costanti intere

```
switch (espressione)
  {
        case valore1:   Sequenza di istruzioni1;
                           break;
        case valore2:   Sequenza di istruzioni2:
                           break;
        …
        case valoreN:  Sequenza di istruzioniN;
                           break;
        default:
             Sequenza di istruzioni di default;
  }
```

ATTENZIONE!!!

- Lo SWITCH traduce nel linguaggio di programmazione una serie di SELEZIONI innestate, non esiste un costrutto della programmazione strutturata per realizzare lo SWITCH

Esempio

Velocità del vento (in Km/h)	Classificazione	Classe
sotto i 25	Vento debole	D
25 – 38	Vento forte	V
39 – 54	Tempesta	T
55 – 72	Forte tempesta	F
Oltre i 72	Uragano	U

```
switch (Classe)
     {
     case 'D':  printf("Vento debole, velocità sotto i 25 KM/h");
        break;
     case 'V': printf("Vento forte, velocità tra i 25 e i 38 KM/h");
        break;
     case 'T': printf("Tempesta, velocità tra i 39 e i 54 KM/h");
        break;
     case 'F': printf("Forte tempesta, velocità tra i 55 e i 72KM/h");
        break;
     case 'U': printf("Uragano, velocità oltre i 72 KM/h");
        break;
     }
```


Esercizio

AreaToR.c

- Scrivere un algoritmo e un programma che consenta di calcolare l'area di un triangolo o l'area di un rettangolo semplicemente chiedendo all'utente di inserire la lettera iniziale della figura e le dimensioni

Esercizio

ContaGiorni.c

- Scrivere un algoritmo e un programma che fornita una data in input fornisca come risultato il numero di giorni dall'inizio dell'anno

Il controllo del programma in C

- La maggior parte dei programmi richiede delle iterazioni o cicli
- Il ciclo è un insieme di istruzioni che è eseguito fino al soddisfacimento di una determinata condizione
- I tipi di iterazioni possono essere :
  - Condizionali
    - Pre-condizionali
    - Post-condizionali
  - Limitati

I cicli condizionali

- I cicli condizionali sono utili quando non è possibile determinare a priori il numero di volte in cui il ciclo dovrà essere ripetuto
- Si utilizza un valore che consenta di rendere falsa la condizione in un determinato istante dell'esecuzione
- I valori prendono i nomi di:
  - Sentinella che può assumere un valore qualsiasi
  - Flag (bandiera) che assume solo valori booleani

L'iterazione pre-condizionale in C...

- While

```
while (condizione)
{
  Sequenza di istruzioni da eseguire fino a
   quando la condizione resta verificata;
}
```

```
conta=0;
while (conta<10)
 {
   conta=conta+1;
   printf("%d. Hello World!\n", conta);
 }
```


...L'iterazione pre-condizionale in C

WhileSentinella.c

```
#include <stdlib.h>
int main ()
{
     int numero, quadrato;
     printf("*********************************************************************************");
     printf("\n****   Il programma calcola il quadrato di una serie di interi positivi   ***");
     printf("\n****            dati in input dall'utente. Digitare -1 per terminare     ***");
     printf("\n*********************************************************************************");
     printf("\n\nInserire il valore da elevare al quadrato --> ");
     scanf("%d", &numero);
     while (numero!=-1)
     {
         quadrato=numero*numero;
         printf("Il quadrato di %d e' %d", numero, quadrato);
         printf("\n\nInserire il valore da elevare al quadrato --> ");
         scanf("%d", &numero);
     }
     system ("pause");
     return 0;
}
```

La sentinella deve essere scelta tra valori che l'utente non inserirà mai

L'iterazione post-condizionale in C…

- Do-While

```
do
{
  Sequenza di istruzioni da eseguire fino a
   quando la condizione resta verificata;
 } while (condizione);
```

```
  /* esegui le operazioni fino al primo numero pari
  */
do
 {   …
     printf("Inserire un numero ->");
     scanf ("%d", &num);
  } while ( (num %  2) != 0);
```


ATTENZIONE!!!

- Il DO-WHILE non è la traduzione precisa del REPEAT-UNTIL
- Nell'algoritmo è necessario continuare ad usare il costrutto REPEAT UNTIL che poi deve essere tradotto nel linguaggio di programmazione con il DO-WHILE

…L'iterazione post-condizionale in C

DoWhileSentinella.c

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
     int numero, quadrato;
     printf("*********************************************************************************");
     printf("\n****   Il programma calcola il quadrato di una serie di interi positivi   ****");
     printf("\n****             dati in input dall'utente. Digitare -1 per terminare     ****");
     printf("\n*********************************************************************************");
     do
     {
         printf("\n\nInserire il valore da elevare al quadrato --> ");
         scanf("%d", &numero);
         if (numero!=-1)
         {
             quadrato=numero*numero;
             printf("Il quadrato di %d e' %d", numero, quadrato);
         }
     }while(numero!=-1);
     system ("pause");
     return 0;
}
```


Esercizio

- Scrivere un programma che consenta all'utente, dato un importo in input, di scegliere una tra le seguenti operazioni:
  - Calcolare uno sconto del 10 per cento
  - Calcolare uno sconto del 20 per cento
  - Calcolare un aumento del 30 per cento
  - Calcolare un aumento del 40 per cento

ScontoAumento.c

```c
#include <stdio.h>
#include <stdlib.h>


int main ()
{
     /*Dichiarazione delle variabili locali*/
     float importo ;/* I/O, */
     int numero ;  /* I/O, */

     printf("************************************************************\n");
     printf("**              Calcola sconto e aumento                  **\n");
     printf("** Dato in input un importo e il numero dell'operazione **\n");
     printf("**          da eseguire il programma restituisce          **\n");
     printf("**                l'importo aggiornato                    **\n");
     printf("************************************************************\n");

    do
    {
         printf ("\n\n\n--->  1  <--- per uno sconto del 10 per cento");
         printf ("\n--->  2  <--- per uno sconto del 20 per cento ");
         printf ("\n--->  3  <--- per un aumento del 30 per cento");
         printf ("\n--->  4  <--- per un aumento del 40 per cento ");
         printf ("\n--->  5  <--- per uscire");
         printf ("\n\nDigitare il numero dell'operazione da eseguire---> ");
         scanf ("%d", &numero);
```



```c
         switch (numero)   {
                case 1:
                     printf ("\nInserire l'importo da scontare  ---> ");
                     scanf ("%f", &importo);
                     importo = importo * 0.9 ;
                     printf ("\n Importo aggiornato=  %.2f   \n\n", importo);
                     break ;
                case 2:
                     printf ("\nInserire l'importo da scontare  ---> ");
                     scanf ("%f", &importo);
                     importo = importo * 0.8;
                     printf ("\n Importo aggiornato=  %.2f   \n\n", importo);
                     break ;
                case 3 :
                     printf ("\nInserire l'importo da aumentare  ---> ");
                     scanf ("%f", &importo);
                     importo = importo * 0.3;
                     printf ("\n Importo aggiornato=  %.2f   \n\n", importo);
                     break ;
                case 4 :
                     printf ("\nInserire l'importo da aumentare  ---> ");
                     scanf ("%f", &importo);
                     importo = importo * 0.4;
                     printf ("\n Importo aggiornato=  %.2f   \n\n", importo);
                     break ;
                default :
                         printf ("\n");
          }  /* fine switch */

          system ("PAUSE");
    } while ( numero != 5 ) ;

return 0;
}
```


Iterazione limitata o controllata da un contatore

- Un contatore (o accumulatore) è una variabile che consente di controllare il numero di volte in cui un ciclo deve essere ripetuto
- È necessario:
  - Inizializzare la variabile fuori dal ciclo (assegnarle un valore iniziale)
  - Definire una condizione che consenta di ripetere le istruzioni del ciclo
  - Incrementare la variabile accertandosi che dopo un numero finito di volte raggiunga il valore indicato nella condizione del ciclo

Esempio

MediaWhile7.c

```c
#include <stdio.h>
#include <stdlib.h>

int main ()
{
int conta_numeri, numero;
float media;
/* Inizializzazione del contatore e dell'accumulatore*/
conta_numeri = 1;
media=0;
printf("\n\n*************************************************************************\r
printf("***** Il programma calcola la media tra 7 numeri interi dati in input ****\n");
printf("*************************************************************************\n");
/* Il ciclo wile terminerà dopo l'inserimento del 7° numero */
while (conta_numeri <= 7)
    {
    printf("Inserisci il %d numero -->", conta_numeri);
    scanf("%d", &numero);
    conta_numeri=conta_numeri+1;
    media= media+numero;
   }
media=media/7;
printf("\n\nLa media e' --> %f \n\n", media);
system ("pause");
return 0;
}
```

Iterazione Limitata in C

HelloFor.c

- For

```
for (InizializzazioneContatore; Condizione; Incremento)
  {
    Sequenza di istruzioni eseguite per un numero di passi
      prestabilito;
  }
```

```
for (conta=0; conta<10; ++conta)
  {
   printf("%d. Hello World!\n", conta+1);
  }
```


L'istruzione for

- L'istruzione for è particolarmente indicata per costruire iterazioni controllate da un contatore
- Consente in un'unica istruzione di:
  - inizializzare il contatore
  - testare la condizione di fine ciclo
  - aggiornare il contatore (variabile di controllo del ciclo)

Esempio

MediaFor7.c

Conviene definire una costante N che rende flessibile il programma

conta_numeri+=1
consente di incrementare il contatore

%.2f
consente di visualizzare il dato definito double con sole due cifre decimali

```
#include <stdio.h>
#include <stdlib.h>
#define N 7

int main ()
{
int conta_numeri, numero;
float media;
/*inizializzazione dell'accumulatore */
media=0;

printf("\n\n **** Il programma calcola la media di N numeri *****\n\n\n");
/* L'istruzione FOR consente di eseguire un gruppo di istruzioni N volte */
for (conta_numeri = 1; conta_numeri <= N; conta_numeri+=1)
    {
    printf("Inserisci il %d numero -->", conta_numeri);
    scanf("%d", &numero);
    media= (float)media+numero;
    }
media=media/N;

printf("\n\nLa media e' --> %.2f \n\n", media);
system ("pause");
return 0;
}
```


Come funziona il for

- Prima è eseguita l'inizializzazione, poi è verificata la condizione di ripetizione
- Se la condizione è vera viene eseguita l'istruzione da ripetere, poi viene eseguita l'istruzione di aggiornamento e verificata nuovamente la condizione di ripetizione
- Quando la condizione di ripetizione assume valore falso allora l'istruzione eseguita è l'istruzione scritta dopo la chiusura del for

Esercizio

PagaOraria.c

- Costruire un algoritmo e il relativo programma che calcola e mostra il salario di un numero predeterminato (richiesto in input) di impiegati
- Si assume che
  - Il salario di un impiegato è dato dalla formula

    *salario = orelavorate * pagaoraria*

Abbreviazione dell'istruzione di assegnazione...

- Le istruzioni di assegnazione seguono la sintassi

```
variabile = variabile operatore espressione
```

- `conta_numeri = conta_numeri +1`
- `totale = totale + paga`

- C fornisce una notazione molto più concisa che consente di abbreviare le operazioni di assegnamento

...Abbreviazione dell'istruzione di assegnazione

- Le istruzioni di assegnazione abbreviate seguono la sintassi

```
variabile operatore = espressione
```

- `conta_numeri + = 1`
- `totale += paga`

- In C questa notazione può essere utilizzata con tutti gli operatori aritmetici +, -, *, / e %

Incremento e decremento dei contatori...

- Il C fornisce gli operatori unari di incremento e decremento la sintassi è la seguente

```
                Notazione prefissa
++ variabile                          -- variabile
                Notazione postfissa
variabile ++                          variabile --
```

- `conta_numeri ++`
- `++indice`
- `temperatura--`

- Il valore della variabile dipende dalla posizione dell'operatore

...Incremento e decremento dei contatori...

- L'istruzione
  - alfa = beta++;
- È equivalente all'esecuzione delle istruzioni:
  - alfa = beta;
  - beta = beta + 1;
- L'istruzione
  - alfa = ++beta;
- È equivalente all'esecuzione delle istruzioni:
  - beta=beta + 1;
  - alfa=beta;

...Incremento e decremento dei contatori

```
Se n= 4 le istruzioni seguenti
  printf("%d", --n);
  printf("%d", n);

stampano 3 3

Se n= 4 le istruzioni seguenti
  printf("%d", n--);  /*prima usa n e poi usa
                        l'operatore*/
  printf("%d", n);

stampano 4 3
```


Esercizi

- Scrivere l'algoritmo e il programma che calcoli il MCD tra due numeri dati in input usando l'algoritmo euclideo
- Scrivere l'algoritmo e il programma che per N coppie di numeri inserite in input dall'utente calcoli il massimo di ognuna di esse