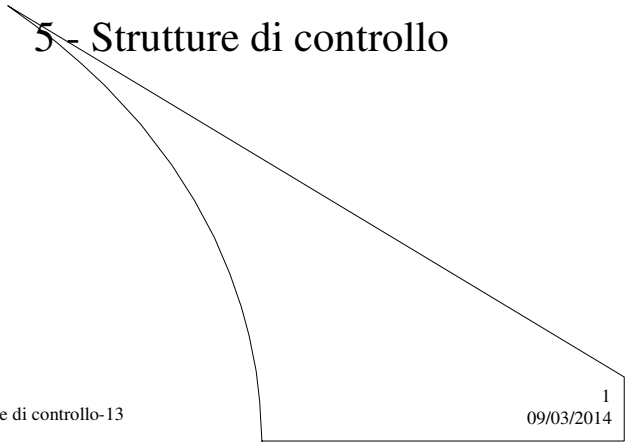
# Corso sul linguaggio C

## Modulo 1

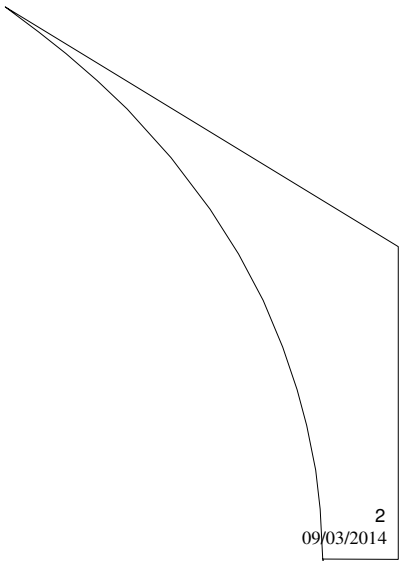
## 5 - Strutture di controllo





# Prerequisiti

- Elementi di logica
- Istruzioni semplici e loro uso
- Tipi di dato



# Introduzione

In molti casi le sole istruzioni semplici (lettura, stampa e assegnamento) non sono sufficienti a risolvere certi problemi.

In questo caso c'è la necessità di introdurre strutture che consentono di alterare il flusso di esecuzione del programma.

Questi costrutti sintattici prendono il nome di **strutture di controllo** e possiamo classificarle in:

- **Sequenza**
- **Selezione**
- **Iterazione**



# Informazioni generali

Attraverso l'uso delle strutture di controllo sarà possibile realizzare programmi anche molto complessi.

Le strutture di controllo sono strumenti sintattici che *consentono di alterare l'esecuzione sequenziale dei programmi.*

E' sempre indispensabile che la progettazione dei programmi segua la tecnica **top-down**, che prevede la scomposizione di un problema in sottoproblemi, via via più semplici, da risolvere in un secondo momento.

# La sequenza

Tutte le istruzioni presenti nella parte eseguibile del programma si dice che costituiscono una **SEQUENZA**, nel senso che costituiscono un blocco che può essere considerato come un'unica istruzione

Quando si vuole considerare un blocco di istruzioni come se fosse una sola istruzione si utilizza la sintassi seguente:

{ *istruzione*;
*istruzione*;
……
}

Questa struttura di controllo prende il nome di **SEQUENZA** ed è caratterizzata dalla coppia di parentesi graffe aperta "{" e chiusa "}".



# La sequenza

```
/* PROGRAMMA: calcolo della somma di 2 valori interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Dev-C++ */
/* FILE: somma.c. */
#include <stdio.h>
#define RISULTATO "La somma e': %d\n "
int main()
{  int a,b,c;
   printf(("%s","Primo valore: ");      scanf("%d",&a);
   printf("%s","Secondo valore: ");    scanf("%d",&b);
   c=a+b;
   printf("RISULTATO,  c);
  return 0;
}
```

La sequenza è sempre presente anche nel **main**() ed è individuata come già detto dalla coppia di parentesi graffe

# La sequenza

```
/* PROGRAMMA: calcolo della somma di 2 valori interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Dev-C++ */
/* FILE: somma.c. */
#include <stdio.h>
#define RISULTATO "La somma e': %d\n "
int main()
{  int a,b,c;
   printf((“%s”,”Primo valore: “);    scanf(“%d”,&a);
   printf(“%s”,"Secondo valore: “);    scanf(“%d”,&b);
   c=a+b;
   printf(“RISULTATO,  c);
  return 0;
}
```

Notare i commenti posti all'inizio del file. Anche se non sono obbligatori, vanno sempre inseriti perché hanno lo scopo di documentare il lavoro prodotto e facilitare eventuali modifiche future.



# La sequenza

```
/* PROGRAMMA: calcolo della somma di 2 valori interi */
/* AUTORE: Mario Rossi */
/* DATA: 15/02/2002 */
/* COMPILATORE: Dev-C++ */
/* FILE: somma.c. */
#include <stdio.h>
#define RISULTATO "La somma e': %d\n "
int main()
{  int a,b,c;
   printf((“%s”,”Primo valore: “);    scanf(“%d”,&a);
   printf(“%s”,"Secondo valore: “);    scanf(“%d”,&b);
   c=a+b;
   printf(“RISULTATO,  c);
  return 0;
}
```

Notare l'uso delle costanti stringa

La costante può essere utilizzata ad esempio nelle istruzioni di stampa.

# La selezione doppia

Quando si vuole che il programma segua un percorso diverso, tra due possibili alternative, si utilizza la **selezione doppia** che ha la sintassi seguente:

**if** (*espressione_logica)*
*istruzione1;*
**else** *istruzione2***;**

Se *espressione_logica* risulta vera (diversa da 0) viene eseguita *istruzione1*, altrimenti viene eseguita *istruzione2*

Questa struttura di controllo prende il nome di **SELEZIONE DOPPIA**



# La selezione doppia

```
/* PROGRAMMA: Verificare se un numero è pari o dispari */
/* AUTORE: ..... */
/* DATA: 20/02/2003 */
/* COMPILATORE: Dev-C++ */
/* FILE: PariDispari.c */
#include <stdio.h>
#define NUMERO "\nIl numero e' "
#define PARI          "pari\n"
#define DISPARI     "dispari\n"
int main()
{ int numero;
  printf ("%s","Immetti il numero:");
  scanf("%d",&numero);
  if (numero % 2)   printf("%s%s", NUMERO, DISPARI);
  else  printf("%s%s", NUMERO, PARI);
  return 0;
}
```

L'espressione logica (*numero % 2*) risulta **VERA** se b% 2 è diverso da zero (ossia b di valore dispari)

# Operatore ternario

Un caso particolare di selezione doppia si può rappresentare con **l'operatore ternario** "?:".

- L'operatore ternario ha la sintassi:

(*espress_logica*) **?** *istruzione1* **:** *istruzione2***;**

- Esso calcola *espress_logica* e se risulta VERA esegue *istruzione1* altrimenti esegue *istruzione2*.
- Costituisce un modo abbreviato per effettuare una selezione doppia.



# Operatore ternario

Ad esempio l'istruzione

1) (primo>secondo) ? a=primo-secondo : a=secondo-primo;

pone in a la differenza positiva tra primo e secondo.

2) (modulo_x=x>0) ? x : (-x);

se x è >0 in modulo_x pone x, altrimenti pone –x.

3) x= u*x+((y==z) ? a:b)

equivale a x=u*x+a se y=z, a x=u*x+b altrimenti

4) printf("%d è maggiore di %d",(a>b)? a:b,(a<=b)? a:b)

equivale a stampare prima il maggiore e poi il minore tra i due valori a e b.

# La selezione semplice

In alcuni casi, la struttura selettiva si può utilizzare nella forma seguente:

**if** (*espressione_logica*)
    *istruzione***;**

Questa struttura di controllo prende il nome di **SELEZIONE SEMPLICE**



# La selezione semplice

```
#include <stdio.h>
int main()
{    int a;
     printf("%s","Valore: ");
     scanf("%d", &a);
     if (a>0)
        printf("%s", "Il numero è positivo\n");
     return 0;
}
```

Se a>o, l'espressione logica risulta **VERA** e quindi viene eseguita la stampa

# La selezione multipla

Quando la scelta deve essere fatta tra molti valori si usa quest'altra struttura di controllo:

**switch** (*espressione_intera*)
{ **case** *valore_1*:
*istruzione*;**break**;
……..
**default**: *istruzione*;
}

Questa struttura di controllo prende il nome di **SELEZIONE MULTIPLA**

La selezione multipla serve a selezionare un dato valore tra diversi possibili. Se viene trovato il valore corrispondente ad ***espressione_intera*** viene eseguita la relativa ***istruzione***.

In caso contrario la parola **default** consente di trattare le situazioni di mancata corrispondenza.



# La selezione multipla

```
switch (mese)
  { case 1:  printf("Gennaio\n"); break;
    case 2:  printf("Febbraio\n"); break;
    case 3:  printf("Marzo\n"); break;
    case 4:  printf("Aprile\n"); break;
    case 5:  printf("Maggio\n"); break;
    case 6:  printf("Giugno\n"); break;
    case 7:  printf("Luglio\n"); break;
    case 8:  printf("Agosto\n"); break;
    case 9:  printf("Settembre\n"); break;
    case 10: printf("Ottobre\n"); break;
    case 11: printf("Novembre\n"); break;
    case 12: printf("Dicembre\n"); break;
    default: printf("Non so!\n");
  }
….
```

Quando viene trovata una corrispondenza di valori, viene eseguita l'istruzione

Dopo eseguita l'istruzione corrispondente al valore ***mese,*** l'istruzione **break** fa saltare il controllo alla fine dell'istruzione **switch**.

La parola **default** serve a prevedere il caso in cui nessuno dei valori elencati sia corrispondente a ***mese***.

# L'iterazione

Quando si deve eseguire ripetutamente un blocco di istruzioni si usano le **strutture di controllo iterative**.

- Le strutture di controllo iterative servono ad eseguire i **cicli**.
- Un ciclo è la ripetizione di un dato blocco di istruzioni.
- Nelle strutture di controllo iterative è spesso necessario utilizzare una variabile detta **contatore** per tenere traccia del numero di volte che il ciclo viene effettuato.



# L'iterazione predefinita

Un primo tipo di costrutto iterativo è il seguente:

**for** (*contatore=inizio;* *contatore<=fine;* *contatore++*)
*istruzione***;**

Condizione di ripetizione

Questa struttura di controllo esegue ripetutamente *istruzione* per ciascuno dei valori che *contatore* assume, partendo da *inizio* fino a *fine*. L'istruzione *contatore*++ serve ad incrementare il valore di *contatore*.

Questa struttura prende il nome di **ITERAZIONE PREDEFINITA** e si usa quando il programmatore conosce a priori il numero di volte che il ciclo deve essere eseguito

L'**ITERAZIONE PREDEFINITA** esegue il ciclo fintantochè la condizione al centro risulta **VERA**. Quando diviene **FALSA**, ossia quando *contatore>fine* il ciclo termina

# L'iterazione predefinita

Un altro tipo di iterazione predefinita è

**for** (*contatore=inizio;* *contatore>=fine;* *contatore- -*)
*istruzione*;

Condizione di ripetizione

Questa è analoga alla precedente, ma si usa quando il valore *inizio* è maggiore del valore *fine*. Per indicare che il contatore questa volta deve indietreggiare, si usa l'istruzione *contatore - -*.

L'**ITERAZIONE PREDEFINITA** esegue il ciclo fintantochè la condizione al centro risulta **VERA**. Quando diviene **FALSA**, ossia quando *contatore<fine*, il ciclo termina



# L'iterazione predefinita

```
/* PROGRAMMA: Calcolo e stampa della somma di N numeri */
/* …..  */
# include <stdio.h>
# define RISPOSTA "La somma e': "
#define  NUMERO   "Immetti valore: "
int main()
{  int somma=0,  valori,  cont,  valore;
   printf("%s","Numero di valori da sommare: ");
   scanf("%d", &valori);
   for (cont=0;cont<valori;cont++)
   { printf("%s", NUMERO);
     scanf("%d", &valore);
     somma+=valore;
   }
   printf("%s %d\n", RISPOSTA, somma);
   return 0;
}
```

La variabile *cont* è assunta come **contatore** del numero di cicli. La condizione di ripetizione è *cont<valori.*

L'incremento del contatore è *cont++*

# L'iterazione precondizionata

Quando **NON** si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente:

**while** (*espress-logica*)
*istruzione***;**

Questa struttura di controllo esegue il ciclo fintantochè *espressione-logica* risulta **VERA**. Esce quando diventa **FALSA**.

Ovviamente, se *espressione-logica* è falsa già all'inizio, la while NON SARA' ESEGUITA NEMMENO UNA VOLTA

Questa struttura prende il nome di **ITERAZIONE PRECONDIZIONATA** poiché la condizione da verificare si trova prima del blocco di istruzioni da ripetere



# L'iterazione postcondizionata

In altri casi, e sempre quando NON si conosce a priori il numero di volte che un ciclo deve essere ripetuto, si usa la seguente:

**do**
{ *istruzione*
} **while** (*espress_logica*)**;**

Questa struttura prende il nome di **ITERAZIONE POSTCONDIZIONATA** poiché la condizione da verificare si trova dopo il blocco di istruzioni da ripetere

Questa struttura di controllo esegue il ciclo fino a quando *espressione_logica* risulta **VERA**. Esce quando diventa **FALSA**.

Ovviamente, anche se *espressione logica* è falsa già all'inizio, la do-while verrà eseguita ALMENO UNA VOLTA.

# L'iterazione postcondizionata

```
# include <stdio.h>
#include <time.h>
int main()
{  int i,j,risposta;   int fine, esatte=0, prove=0;
   float percentuale;   time_t t=time(0);
   srand(t);
   do  {   prove++; i=rand()%100;  j=rand()%100;
            printf("\nQuanto fa %d + %d ? ",i,j);
           scanf("%d",&risposta);
           if(risposta != i+j)    printf("Errato!!\n");
           else    { printf("Esatto\n");     esatte++;       }
          printf("Ancora (ESC per terminare)? ");
    } while ((fine=getchar())!=27 );
    printf("\nEsatte = %d\n", esatte);
    printf("Totali = %d\n", prove);
   printf("Percentuale = %2.2f", (float) esatte/prove*100);
   return 0;
}
```

Riportiamo un semplice programma per verificare l'abilità dell'utente nel fare i calcoli a mente. Il programma funziona a ciclo continuo fino a quando l'utente preme il tasto **ESC** a cui corrisponde il codice **ASCII** 27.



# Strutture di controllo composte

```
#include <stdio.h>
int main()
{  float a,b,r;
   char op;
  printf("immetti il primo operando: ");
  scanf("%f",&a);
  r=a;
  do  { printf("operatore (+,-,*,/,=): ");
        do  { op=getchar();
        } while(!strchr("+-*/=", op));
       if (op!='=')
       { printf("Secondo operando: ");
         scanf("%f", &b);
         switch(op)
         {  case '+':
                 r= r + b;
                 break;
            case '-':
         r= r - b;
         break;
case '*':
        r= r * b;
        break;
      case '/':
         if (b!=0)   r= r / b;
        break;
     }
     if ((b==0) && (op=='/'))
        printf("Operazione
   impossibile!\n");
   }
  } while (op!='=');
  printf("Risultato: %f\n", r);
  return 0;
}
```

Riportiamo un altro programma di esempio che rappresenta una calcolatrice a ciclo continuo. Il programma termina quando l'utente immette il simbolo '='; il programma stampa il risultato

# Argomenti

- La sequenza
- La selezione doppia
- L'operatore ternario
- La selezione semplice
- La selezione multipla
- L'iterazione
- L'iterazione predefinita
- L'iterazione precondizionata
- L'iterazione postcondizionata
- Strutture di controllo composte



# Attività

1. Scrivere un programma che, letta da input una sequenza di valori interi, calcoli e stampi:
   a. il massimo;
   b. la posizione del massimo;
   c. il minimo;
   d. la posizione del minimo;
   e. la media aritmetica totale;
   f. la media aritmetica escludendo il massimo e il minimo.
2. Scrivere un programma che, leggendo da input un intero N, stampi in modo formattato le tavole aritmetiche da 1 a N (nelle tavole devono essere presenti in ordine: il numero N, il suo quadrato, il cubo, la radice quadrata e la radice cubica di N).
3. Letto in input un intero N determinare e stampare la somma delle cifre di cui è composto.

# Attività

4. Ad un gioco partecipano 4 giocatori e il banco. Ogni giocatore riceve un numero da 1 a 4 ed effettua una puntata a piacere, disponendola in un piatto. Terminate le puntate, il banco pone in un altro piatto una manciata di semi ed inizia a raccoglierli a 5 a 5. Il gioco termina quando nel piatto resta un numero di semi minore o uguale a 5. Se sono esattamente 5 il banco vince tutte le puntate, altrimenti vince tutto il giocatore il cui numero è uguale al numero di semi rimasti. Scrivere il programma relativo.
5. Un numero intero positivo dicesi perfetto se è uguale alla somma dei suoi divisori eccetto se stesso. Letto un generico intero positivo da terminale, stabilire se esso è perfetto o meno.



# Altre fonti di informazione

- J. Purdum, C – ed. Jackson
- Romagnoli-Ventura, C/C++ - Ed. Petrini
- A.Bellini-A.Guidi, Conoscere il C – ed. McGraw Hill