Istruzione **switch**

- Può essere usata per realizzare una selezione a più vie.

Sintassi:

```
switch (espressione) {
  case valore-1:  istruzioni-1
                  break;
  ...
  case valore-n:  istruzioni-n
                  break;
  default :  istruzioni-default
}
```

Semantica:

1. viene valutata `espressione`
2. viene cercato il primo `i` per cui il valore di `espressione` è uguale a `valore-i`
3. se si trova tale `i`, allora vengono eseguite `istruzioni-i` altrimenti vengono eseguite `istruzioni-default`

Esempio:

```
int giorno;
...
switch (giorno) {
  case 1:  printf("Lunedi'\n");
           break;
  case 2:  printf("Martedi'\n");
           break;
  case 3:  printf("Mercoledi'\n");
           break;
  case 4:  printf("Giovedi'\n");
           break;
  case 5:  printf("Venerdi'\n");
           break;
  default :  printf("Week end\n");
}
```

- Se abbiamo più valori a cui corrispondono le stesse istruzioni, possiamo raggrupparli come segue:

```
case valore-1:  case valore-2:
                        istruzioni
                        break;
```

Esempio:

```
int giorno;
...
switch (giorno) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5:  printf("Giorno lavorativo\n");
           break;
  case 6:
  case 7:  printf("Week end\n");
           break;
  default :  printf("Giorno non valido\n");
}
```


Osservazioni sull'istruzione **switch**

- L'`espressione` usata per la selezione può essere una qualsiasi espressione C che restituisce un valore intero.
- I valori specificati nei vari **case** devono invece essere costanti (o meglio valori noti a tempo di compilazione). In particolare, non possono essere espressioni in cui compaiono variabili.

Esempio: Il seguente frammento di codice è sbagliato:

```
int a;
switch (a) {
  case a<0:  printf("negativo\n");
             /* ERRORE: a<0 non e' una costante*/
  case 0:  printf("nullo\n");
  case a>0:  printf("positivo\n");
             /* ERRORE: a>0 non e' una costante*/
}
```

- In realtà il C non richiede che nei **case** di un'istruzione **switch** l'ultima istruzione sia **break**.

  Quindi, in generale la **sintassi** di un'istruzione **switch** è:

```
switch (espressione) {
  case valore-1:  istruzioni-1
  ...
  case valore-n:  istruzioni-n
  default :  istruzioni-default
}
```

Semantica:

1. viene prima valutata `espressione`
2. viene cercato il primo `i` per cui il valore di `espressione` è pari a `valore-i`
3. se si trova tale `i`, allora si eseguono in sequenza `istruzioni-i`, `istruzioni-(i+1)`, ..., fino a quando non si incontra **break** o è terminata l'istruzione **switch**,
   altrimenti vengono eseguite `istruzioni-default`

Esempio: più **case** di uno **switch** eseguiti in sequenza (corretto)

```
int lati;
printf("Immetti il massimo numero di lati del poligono (al piu' 6):  ");
scanf("%d", &lati);
printf("Poligoni con al piu' %d lati:  ", lati);

switch (lati) {
  case 6:  printf("esagono, ");
  case 5:  printf("pentagono, ");
  case 4:  printf("rettangolo, ");
  case 3:  printf("triangolo\n");
           break;
  case 2:  case 1:  printf("nessuno\n");
                    break;
  default :  printf("\nErrore:  valore immesso > 6.\n");
}
```

- N.B. Quando si omettono i **break**, diventa rilevante l'ordine in cui vengono scritti i vari **case** . Questo può essere facile causa di errori.

  È buona norma mettere **break** come ultima istruzione di ogni **case**

Esempio: più case di uno switch eseguiti in sequenza (scorretto)

```
int b;
printf("Immetti un numero tra 1 e 6:  ");
scanf("%", &b);

switch (b) {
  case 1:  case 2:  case 3:  case 5:  printf("Numero primo\n");
  case 4:  case 6:                    printf("Numero non primo\n");
  default :                           printf("Valore non valido!\n");
}
```

```
=> 3↩
Numero primo
Numero non primo
Valore non valido!
=>
```

```
=> 4 ↩
Numero non primo
Valore non valido!
=>
```