2.6 Istruzione `switch-case`

Le decisioni a più vie possono essere risolte utilizzando più `if-else` in cascata:

```
if(espressione1)
  istruzione1
else
  if(espressione2)
    istruzione2
  else
    if(espressione3)
      istruzione3
    ...
        else
          istruzioneN
```

Ognuna delle istruzioni può essere formata da più istruzioni, se racchiuse tra parentesi graffe (istruzioni composte).

Un'altra soluzione è data dal costrutto `switch-case`, che consente di implementare decisioni multiple basandosi sul confronto fra il risultato di un'espressione (`int` o `char`) e un insieme di valori costanti (Figura 2.5).

```
switch(espressione) {
  case costante1:
     istruzione
     ...
  case costante2:
     istruzione
     ...
  case costante3:
     istruzione
     ...
  [default:
     istruzione
     ...       ]
}
```

```
switch(espressione) {
  case costante1:
     istruzione
     ...
     break;
  case costante2:
     istruzione
     ...
     break;
  case costante3;
     istruzione
     ...
     break;
  ...
  [default:
     istruzione
     ...       ]
}
```

Figura 2.5 A sinistra sintassi del costrutto `switch-case`; a destra forma spesso utilizzata del costrutto `switch-case`

La parola `switch` è seguita da una *espressione*, racchiusa tra parentesi tonde, il cui risultato deve essere di tipo `int` o `char`. Il resto del costrutto è formato da un'istruzione composta, costituita da un numero qualsiasi di sottoparti, ciascuna delle quali inizia con la parola chiave `case`, seguita da un'espressione costante intera o carattere. Questa è separata, tramite un simbolo di due punti, da una o più istruzioni.

In fase di esecuzione, viene valutata *espressione* e il risultato viene confrontato con *costante1*: se i due valori sono uguali il controllo passa alla prima istruzione che segue i due punti corrispondenti, altrimenti si prosegue confrontando il risultato dell'espressione con *costante2*, e così di seguito. Una volta che il controllo è trasferito a una certa istruzione vengono eseguite linearmente tutte le rimanenti istruzioni presenti nello `switch-case` a sinistra della Figura 2.5.

Spesso, nell'utilizzo di questo costrutto, il programmatore desidera che vengano eseguite solamente le istruzioni associate a un singolo `case`. A questo scopo abbiamo inserito in Figura 2.5 a destra, al termine di ogni `case`, l'istruzione `break`, che causa l'uscita immediata dallo `switch`. Si osservi comunque che anche la situazione a sinistra può rivelarsi utile in particolari circostanze e va interpretata correttamente come una possibilità in più offerta dal linguaggio.

Se l'espressione non corrisponde a nessuna delle costanti, il controllo del programma è trasferito alla prima istruzione che segue la parola riservata `default` (se presente).

I valori *costante1*, *costante2*, .., *costanteN* possono essere delle espressioni costanti come `3*2+5` o `5*DELTA`, dove `DELTA` è una costante. Il Listato 2.4 è un esempio di utilizzo del costrutto `switch-case`.

```
/* Esempio utilizzo case */

#include <stdio.h>

int x;
main()
{
printf("Digita una cifra: ");
scanf("%d", &x);

switch(x) {
  case 0:
    printf("zero\n");
    break;
  case 1:
    printf("uno\n");
    break;
  case 2:
    printf("due\n");
    break;
  case 3:
    printf("tre\n");
    break;
  case 4:
    printf("quattro\n");
    break;
  case 5:
    printf("cinque\n");
    break;
  default:
    printf("non compreso\n");
    break;
}
}
```

Listato 2.4 Esempio di diramazione multipla del flusso di esecuzione

È possibile far corrispondere a un gruppo di istruzioni più costanti, ripetendo più volte la parola chiave `case` seguita dai due punti, come nel Listato 2.5.

```
/* Esempio utilizzo case */

#include <stdio.h>
```

```
char x;
main()
{
printf("Digita una cifra: ");
scanf("%c", &x);

switch(x) {
  case '2':
  case '4':
  case '6':
    printf("pari\n");
    break;
  case '1':
  case '3':
  case '5':
    printf("dispari\n");
    break;
  default:
    printf("altro\n");
}
}
```

Listato 2.5 Più valori costanti corrispondono allo stesso gruppo di istruzioni