3.1 Istruzione `for`

Quando si desidera ripetere una operazione un determinato numero di volte, si può riscrivere sequenzialmente l'istruzione corrispondente. Per esempio, se si vuole sommare tre volte alla variabile `somma`, inizializzata a 0, il valore 7, si può scrivere:

```
somma = 0;
```

```
somma = somma+7;
somma = somma+7;
somma = somma+7;
```

Risulta però decisamente più comodo inserire in un ciclo l'istruzione che si ripete:

```
somma = 0;
     for(i=1; i<=3; i=i+1)
       somma = somma+7;
```

L'esempio precedente è volutamente semplice per concentrare l'attenzione sulle caratteristiche del costrutto `for`. Alla variabile `somma` viene dato il valore `0`. L'istruzione `for` assegna il valore `1` alla variabile `i`. L'operazione

```
i=1
```

compresa tra la parentesi tonda aperta e il primo punto e virgola è detta *inizializzazione* e non verrà mai più eseguita. Successivamente l'esecuzione prosegue così:

1. 1. se `i<=3` allora vai al passo 2 altrimenti termina
2. 2. `somma=somma+7`
3. 3. `i=i+1`, vai al passo 1

Inizialmente la condizione risulta vera, in quanto 1 è minore o uguale a 3, e quindi viene eseguito il corpo del ciclo, che in questo caso è composto dalla sola istruzione `somma=somma+7`. La variabile `somma` assume il valore `7`. Viene incrementato di 1 il valore di `i`, che quindi assume il valore `2`.
Alla fine del terzo ciclo la variabile `somma` ha il valore `21` e `i` vale `4`. Nuovamente viene verificato se `i` è minore o uguale a 3. La condizione risulta falsa e l'iterazione ha termine; il controllo passa all'istruzione successiva del programma.

Il formato del costrutto `for` è il seguente:

```
for(esp1; esp2; esp3)
  istruzione
```

Si faccia attenzione ai punti e virgola all'interno delle parentesi. Il ciclo inizia con l'esecuzione di *esp1*, la quale non verrà mai più eseguita. Quindi viene esaminata *esp2*. Se *esp2* risulta vera, viene eseguita *istruzione*, altrimenti il ciclo non viene percorso neppure una volta.

Successivamente viene eseguita *esp3* e di nuovo valutata *esp2* che se risulta essere vera dà luogo a una nuova esecuzione di `istruzione`. Il processo si ripete finché *esp3* non risulta essere falsa. Nell'esempio precedente,

```
for(i=1; i<=3; i=i+1)
  somma=somma+7;
```

esp1 era `i=1`, *esp2* `i<=3`, *esp3* `i=i+1` e *istruzione* era `somma=somma+7`.

Nella sintassi del `for`, *istruzione*, così come nel costrutto `if`, può essere un blocco, nel qual caso deve iniziare con una parentesi graffa aperta e terminare con parentesi graffa chiusa. Supponiamo di voler ottenere la somma di tre numeri interi immessi dall'utente: si può scrivere:

```
somma = 0;
scanf("%d", &numero);
somma = somma+numero;
scanf("%d", &numero);
somma = somma+numero;
scanf("%d", &numero);
somma = somma+numero;
```

La variabile `somma` che conterrà, di volta in volta, la somma degli interi letti, viene inizializzata a zero. Successivamente, per tre volte è richiesta l'immissione di un valore che viene immagazzinato nella variabile `numero`. A ogni lettura corrisponde un incremento di `somma` del valore di `numero`. Lo stesso risultato lo si ottiene in una forma più sintetica con il seguente codice:

```
somma = 0;
for(i=1; i<=3; i =i+1) {
  scanf("%d", &numero);
```

```
  somma = somma+numero;
}
```

Nel Listato 3.1 osserviamo un programma che calcola la somma di cinque numeri interi immessi dall'utente.

```
/* Esempio di utilizzo dell'istruzione for
   Calcola la somma di cinque numeri interi
   immessi dall'utente                        */

#include <stdio.h>

int i, somma, numero;

main()
{
printf("SOMMA 5 NUMERI\n");
somma = 0;

for(i=1; i<=5; i=i+1) {
  printf("Inser. intero: ");
  scanf("%d", &numero);
  somma = somma + numero;
}

printf("Somma: %d\n",somma);
}
```

Listato 3.1 Iterazione con l'istruzione `for`

Durante l'esecuzione del programma l'utente sarà chiamato a introdurre cinque valori interi:

```
SOMMA 5 NUMERI
Inser. intero: 32
Inser. intero: 111
Inser. intero: 2
Inser. intero: 77
Inser. intero: 13
Somma: 235
```

In questo caso l'utente ha inserito 32, 111, 2, 77 e 13. Naturalmente il numero dei valori richiesti può variare: se si vogliono accettare 100 valori si deve modificare soltanto la condizione di fine ciclo *esp2* nel `for`:

```
for(i=1; i<=100; i=i+1)
```

Potrebbe risultare utile far apparire il numero d'ordine d'inserimento; a tale scopo si deve modificare la prima `printf`:

```
printf("\nInser. intero n.%d: ", i);
```

che genererà

```
Inser. intero n.1: 32
Inser. intero n.2: 111
Inser. intero n.3: 2
...
```

Nel costrutto `for`

```
for(esp1; esp2; esp3)
  istruzione
```

esp1, come *esp2* ed *esp3*, può essere una qualsiasi espressione ammessa in C .

Per ora limitiamoci a vederne alcune applicazioni classiche:

```
for(i=5; i>=1; i=i-1)
```

Il ciclo viene ripetuto cinque volte ma la variabile che controlla il ciclo viene inizializzata al valore massimo (5) e decrementata di uno a ogni passaggio; l'ultima iterazione avviene quando il valore assunto è `1`. Se si desidera far assumere alla variabile che controlla un ciclo, ripetuto quattro volte, i valori 15, 25, 35 e 45 si potrà scrivere

```
for(i=15 ; i<=45; i=i+10)
```

Analogamente, se i valori devono essere 7, 4, 1, –2, –5, –8 si avrà:

```
for(i=7; i>=-8; i=i-3)
```

Quando si predispone la ripetizione ciclica di istruzioni si deve fare molta attenzione a che l'iterazione non sia infinita, come nell'esempio seguente:

```
for(i=5; i>=5; i=i+1)
```

Il valore di `i` viene inizializzato a `5`; è dunque verificata la condizione `i>=5`. Successivamente `i` viene incrementato di una unità e assume di volta in volta i valori `6`, `7`, `8` ecc. che risulteranno essere sempre maggiori di `5`: il ciclo è infinito. Il compilatore non segnalerà nessun errore ma l'esecuzione del programma probabilmente non farà ciò che si desidera.

> ✓ **NOTA**
>
> Situazioni di questo genere si presentano di frequente perché non è sempre banale riconoscere un'iterazione infinita; perciò si utilizzino pure le libertà linguistiche del C, ma si abbia cura di mantenere sempre uno stile di programmazione strutturato e lineare, in modo da accorgersi rapidamente degli eventuali errori commessi.

Ognuna delle *esp1*, *esp2* ed *esp3* può essere l'espressione nulla, nel qual caso comunque si deve riportare il punto e virgola corrispondente. Vedremo nei prossimi paragrafi alcuni esempi significativi. Anche l'*istruzione* del `for` può essere nulla, corrispondere cioè a un punto e virgola, come nell'esempio:

```
for(i=1; i<1000; i=i+100)
  ;
```

che incrementa il valore `i` di 100 finché `i` risulta minore di 1000. Si osservi che al termine dell'esecuzione dell'istruzione `i` avrà valore 1001: è chiaro perché?