CORSO DI PROGRAMMAZIONE
A.A. 2015-16

Dispensa 7

Laboratorio

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

7.2 Istruzione for

L'ordine di esecuzione delle istruzioni nei programmi in C è per default dall'alto verso il basso. L'esecuzione comincia con la prima istruzione di *main()* e va avanti istruzione per istruzione, fino a quando non viene raggiunta la fine di tale funzione. Nei programmi in C reali, raramente quest'ordine viene rispettato. Il linguaggio prevede infatti una gamma di istruzioni di controllo che permettono di gestire il flusso di esecuzione delle istruzioni.

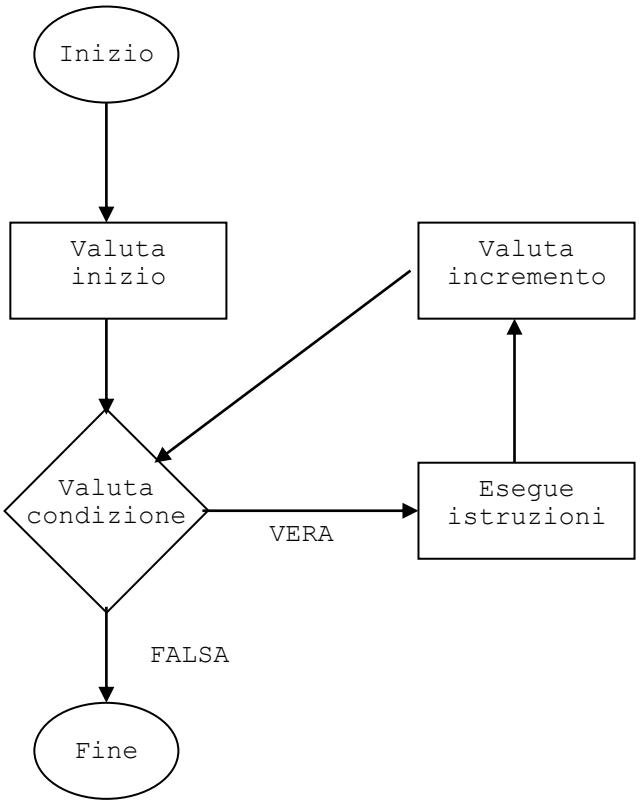
L'istruzione *for* è un costrutto di programmazione che esegue un blocco di istruzioni per un certo numero di volte. Spesso il costrutto viene detto "ciclo" perchè il flusso di esecuzione ripete le esecuzioni più di una volta. Un costrutto *for* ha la struttura seguente:

```
for (inizio; condizione; incremento)
{
      istruzione;
}
```

inizio, *condizione*, e *istruzione* sono espressioni del C, mentre *istruzione* è un blocco di istruzioni. Quando durante l'esecuzione viene incontrata l'istruzione *for* si verifica quanto segue:

1. Viene valutata l'espressione *inizio*. Solitamente questa è un istruzione di assegnamento che imposta una variabile a un particolare valore.
2. Viene valutata l'espressione *condizione*. Questa solitamente è una espressione relazionale.
3. Se la *condizione* è falsa (cioè se vale zero) l'istruzione for si conclude e l'esecuzione passa alla prima istruzione dopo il fot
4. Se la *condizione* è vera (cioè se vale uno) vengono eseguite le istruzioni del blocco *istruzione*
5. Viene valutata l'espressione *incremento* e l'esecuzione torna al punto 2

Vediamo schematicamente una rappresentazione di un'istruzione for:



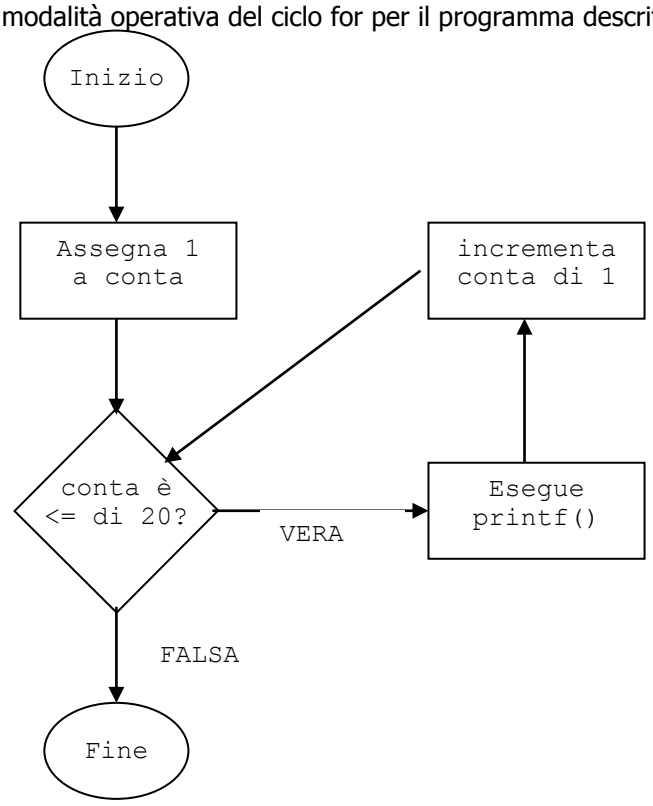
Proviamo a scrivere un programma che stampi tutti i numeri che vanno da 1 a 10. Potremo scrivere un listato con 10 istruzioni printf() oppure potremo scrivere un codice più compatto utilizzando un istruzione for:

```
#include <stdio.h>

int conta;

int main()
{
      /*Stampa i numeri da 1 a 20 */

      for (conta = 1; conta <= 20; conta++)
      {
            printf("%d\n",conta);
      }

      return 0;
}
```

L'output del programma:

```
1
2
3
4
5
6
7
8
9
10
```

La figura sottostante illustra la modalità operativa del ciclo for per il programma descritto sopra:



Un istruzione *for* può essere eseguita tranquillamente all'interno di un'altra istruzione for, questo comportamento viene detto annidamento. Ad esempio proviamo a scrivere un programma che stampi a video un area di 5 righe e 20 colonne di caratteri x.

```
#include <stdio.h>

int righa, colonna;

int main()
{
      for (riga = 1; riga <= 5; riga++)
      {
            for (colonna = 1; colonna <= 20; colonna ++)
            {
                  printf("x");
            }
            printf("\n");
      }

      return 0;
}
```

ecco l'output del programma:

```
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
```

7.3 Break e continue

Torna a volte conveniente interrompere un'iterazione in corrispondenza di punti del codice diversi dall'inizio o dalla fine del corpo. L'istruzione break serve allo scopo del caso di istruzioni *for*, *while* e *do*, in analogia con il suo uso del costrutto *switch*. Essa provoca l'immediata terminazione del ciclo o dell'istruzione *switch*.

L'istruzione continue è legata a *break*, ma d'uso meno comune; causa l'esecuzione dell'iterazione successiva del ciclo *for*. Nel caso *for*, l'esecuzione prosegue con la parte relativa all'incremento. L'istruzione continue non si applica all'istruzione *switch*. La presenza di un istruzione continue all'interno di un istruzione *switch* che sia a sua volta contenuta in un ciclo causa l'esecuzione dell'iterazione successiva del ciclo.

Per capire la sottile, quanto importante, differenza tra le due istruzioni, presentiamo un semplice codice in cui si legge un numero da tastiera che volgiamo sia compreso tra 0 e 100, se tale valore risulta essere negativo, si esce dal ciclo, mentre se è maggiore di cento si richiede di inserire un valore valido; se il valore è tra 1 e 100, si stampa a video il suo quadrato, se è zero si esce:

```
int valore, quadrato;


for (i = 0; i < 20; i++)
{
      scanf("%d", &valore)
```

```
            if (valore < 0)
            {
                printf("Valore non consentito\n");
                break;
                /* esce dal ciclo */
            }

            if (valore > 100)
            {
                printf("Valore non consentito\n");
                continue;
            }

            quadrato = valore * valore;
            printf("%d \n", quadrato);
        }
```

Esercizio 1

Scrivere un programma che calcoli la media di 20 numeri inseriti dall'utente

Esercizio 2

Scrivere un programma che calcoli la somma di n numeri inseriti dall'utente

Esercizio 3

Scrivere un programma che permetta all'utente di inserire 20 numeri, per ogni numero il programma deve indicare se si tratta di un numero pari o dispari.

Esercizio 4

Scrivere un programma che estratto un numero a caso chieda all'utente di indovinarlo concedendo 3 tentativi.

Esercizio 5

Scrivere un programma che stampi tutti i numeri primi compresi nell'intervallo da 1 a 100

La soluzione degli esercizi è nei file allegati