Il ciclo `for`

Programmazione – Canale M-Z

LT in Informatica
19 Dicembre 2016

Università degli Studi di Padova

Tavola dei quadrati

Uno dei primi programmi memorizzati in memoria ed eseguiti su un calcolatore. (David Wheeler e Maurice Wilkes, Cambridge, 1949)

```
// PRE = true
int main() {
    int i = 0;
    while(i < 100)
    {
        cout << i << '\t' << i*i << endl;
        i = i + 1;
    }
}
// POST = stampa la tavola dei quadrati da 0 a 99
```

Cicli e contatori

Il programma per il calcolo dei quadrati è un esempio di ciclo controllato da un contatore:

- Il numero di iterazioni è stabilito prima dell'inizio del ciclo
- La ripetizione è gestita da una variabile di controllo, che "conta" il numero di iterazioni

Formato:

1. Inizializza la `variabile_di_controllo` ad un `valore_iniziale`
2. La guardia è `variabile_di_controllo < valore_finale`
3. Esegui il corpo e alla fine incrementa/decrementa la `variabile_di_controllo`

Il ciclo `for`

Sintassi

```
for(iniziale ; condizione ; incremento)
{
    corpo
}
```

- `iniziale` assegna alla variabile di controllo il valore iniziale
- `condizione` è la guardia che controlla il numero di iterazioni del ciclo
- ad ogni iterazione si esegue `corpo ; incremento`

Il ciclo `for`: diagramma di flusso

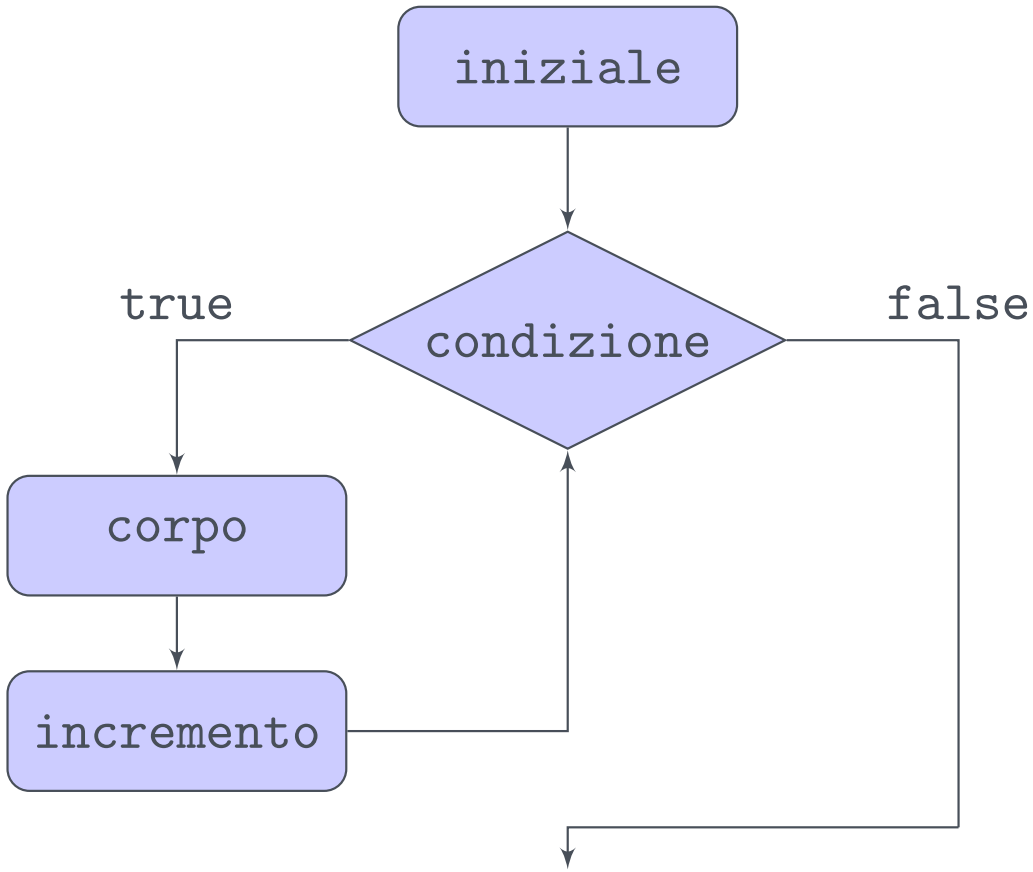

Tavola dei quadrati

Scriviamo il programma che stampa la tavola dei quadrati usando il ciclo `for`

```
// PRE = true
int main()
{
    for(int i = 0; i < 100; i++)
    // R = stampa i quadrati da 0 a i-1 && i <= 100
    {
        cout << i << '\t' << i*i << endl;
    }
}
// POST = stampa la tavola dei quadrati da 0 a 99
```

Correttezza del `for`: gli invarianti

- Per dimostrare che un ciclo `for` è corretto dobbiamo definire un invariante per il ciclo
- Le regole di dimostrazione sono analoghe a quelle del `while`

L'invariante deve rispettare **tre condizioni**:

1. **Condizione iniziale:** dev'essere vero dopo l'esecuzione dell'inizializzazione
2. **Invarianza:** dev'essere vero dopo che si è eseguito corpo e incremento del ciclo
3. **Condizione di uscita:** assieme alla negazione della guardia deve implicare la POST-condizione

Un invariante per la tavola dei quadrati

`R = stampa i quadrati da 0 a i-1 && i <= 100`

1. Rispetta la condizione iniziale?
   - ✓ Si: dopo l'inizializzazione `int i = 0` abbiamo che `i = 0 <= 99` e non ho stampato nulla (la tavola dei quadrati da 0 a -1 è vuota)
2. Rispetta l'invarianza?
   - ✓ Si: in ogni iterazione `i < 100`, si stampa il quadrato di `i` e poi si incrementa `i` di `1`.
3. Rispetta la condizione di uscita?
   - ✓ Si: all'uscita dal ciclo `i >= 100` (negazione della guardia) e `i <= 100` (dall'invariante), quindi `i == 100`. Ho stampato i quadrati da `0` a `i-1`, cioè da `0` a `99`.

Osservazione: ogni ciclo `for` può essere riscritto usando il `while`:

```
for(iniziale; controllo; incremento)
{
    corpo;
}
```

...è equivalente a ...

```
{
    iniziale;
    while(controllo)
    {
        corpo;
        incremento;
    }
}
```

Vale anche il **viceversa**: ogni `while` si può riscrivere con un `for`

Esercizio

Scrivere un programma che legga dallo standard input dei caratteri e man mano che li legge li scrive sullo standard output.
Il programma deve continuare a leggere e stampare caratteri finché non legge la sequenza di tre caratteri `"END"`.

Il programma userà tre variabili booleane per riconoscere la sequenza finale:

- `letto_e` che diventa `true` dopo aver letto `"E"`
- `letto_en` che diventa `true` dopo aver letto `"EN"`
- `letto_end` che diventa `true` dopo aver letto `"END"`

Quando `letto_end` diventa vera si esce dal ciclo di lettura/scrittura.

Il codice dell'esercizio


```
#include <iostream>
using namespace std;

// PRE = cin contiene una sequenza di caratteri c_1,...,c_k che contiene "END"
int main() {
    bool letto_e=false, letto_en=false, letto_end=false;

    for(char c; !letto_end; cout << c) {
        cin >> c;
        if(c == 'D' && letto_en) {
            letto_end = true;
        }
        if(c == 'N' && letto_e) {
            letto_en = true;
        } else {
            letto_en = false;
        }
        if(c == 'E') {
            letto_e = true;
        } else {
            letto_e = false;
        }
    }
    cout << endl;
}
// POST = stampo il prefisso minimo c_1,...,c_j che termina con "END"
//        della sequenza c_1,...,c_k
```

L'invariante per l'esercizio


```
// R = ( ho stampato il prefisso c_1,...,c_h &&
//       letto_e se e solo se l'ultimo carattere e' 'E' &&
//       letto_en se e solo se gli ultimi due caratteri sono "EN" &&
//       letto_end se e solo se gli ultimi tre caratteri sono "END" )
```

1. Rispetta la condizione iniziale?
   - ✓ Si: dopo l'inizializzazione non ho stampato nulla e le tre variabili sono `false`
2. Rispetta l'invarianza?
   - ✓ Si: in ogni iterazione leggo e stampo un carattere. Il valore delle tre variabili booleane si aggiorna rispettando le condizioni
3. Rispetta la condizione di uscita?
   - ✓ Si: all'uscita dal ciclo `letto_end == false` (negazione della guardia) e l'invariante mi dice che ho stampato un prefisso che termina con `"END"`.