Capitolo 3 – Sviluppo di Programmi Strutturati

Outline

Introduzione
Strutture di controllo
If Selection Statement
If…Else Selection Statement
While Repetition Statement
Ripetizione Counter-Controlled
Uso di una sentinella
Strutture di controllo innestate
Operatori di assegnamento
Operatori di Incremento e Decremento

Introduzione

- Prima di scrivere un programma:
  - Comprendere a fondo il problema
  - Pianificare con cura un approccio per risolverlo
- Mentre si scrive un programma:
  - Individuare quali “building blocks” sono disponibili
  - Usare buoni principi di programmazione

Strutture di controllo

- Teorema di Bohm e Jacopini
  - Tutti i programmi possono essere scritti usando tre strutture di controllo fondamentali
    - SEQUENZA: Nativa nel C. I programmi vengono eseguiti sequenzialmente per default
    - SELEZIONE: Il C ne ha tre tipi: `if`, `if…else`, e `switch`
    - ITERAZIONE: Il C ne ha tre tipi: `while`, `do…while` e `for`

`if` Selection Statement

- Struttura di SELEZIONE:
  - Usata per scegliere tra diverse alternative
  - Pseudocodice:

```
If student’s grade is greater than or equal to 60
   Print “Passed”
```

- Se la condizione è vera (`true`)
  - Statement Print eseguito e il programma va al prossimo statement
  - Se falsa (`false`), print statement ignorato e il programma va al prossimo statement
  - L’indentazione rende il programma più leggibile
    - Il C ignora gli spazi bianchi

if Selection Statement

- Statement in C:

```
if ( grade >= 60 )
   printf( "Passed\n" );
```

if…else Selection Statement

- `if`
  - Esegue un’azione solo se la condizione è vera (`true`)
- `if…else`
  - Specifica l’azione da eseguire quando la condizione è vera (`true`) e quando è falsa (`false`)
- Pseudocodice:

```
If student’s grade is greater than or equal to 60
  Print “Passed”
else
  Print “Failed”
```

  - Nota la spaziatura e l’indentazione

if…else Selection Statement

- Codice C:

```
if ( grade >= 60 )
   printf( "Passed\n");
else
   printf( "Failed\n");
```

- Operatore condizionale ternario (`?:`)
  - Ha tre argomenti (condizione, valore se `true`, valore se `false`)
  - Possiamo scrivere:

```
printf( "%s\n", grade >= 60 ? "Passed" : "Failed");
```

  - Oppure:

```
grade >= 60 ? printf( “Passed\n” ) : printf(
  “Failed\n” );
```

if…else Selection Statement

- `if…else` statement innestati
  - Test per casi multipli tramite `if…else` selection statements all’interno di `if…else` selection statement

if…else Selection Statement

– Pseudocodice per if…else innestati

```
If student’s grade is greater than or equal to 90
  Print “A”
else
  If student’s grade is greater than or equal to 80
     Print “B”
  else
     If student’s grade is greater than or equal to 70
        Print “C”
     else
        If student’s grade is greater than or equal to 60
           Print “D”
        else
           Print “F”
```

if…else Selection Statement

- Statement composti:
  - Insieme di statements all’interno di coppie di parentesi graffe
  - Esempio:

```
if ( grade >= 60 )
   printf( "Passed.\n" );
else {
   printf( "Failed.\n" );
   printf( "You must take this course again.\n" );
 }
```

  - Senza le parentesi, lo statement

```
printf( "You must take this course again.\n" );
```

sarebbe eseguito automaticamente

while Repetition Statement

- Struttura iterativa
  - Il programmatore specifica un’azione che deve essere eseguita fino a quando una condizione rimane vera (`true`)
  - Pseudocodice:

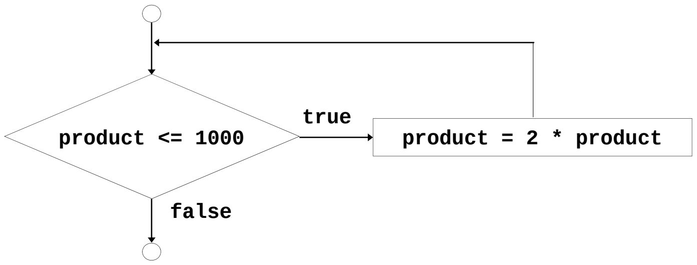
```
While there are more items on my shopping list
   Purchase next item and cross it off my list
```

  - Ciclo `while` ripetuto fino a quando la condizione diviene falsa (`false`)

while Repetition Statement

- Esempio:

```
int product = 2;
while ( product <= 1000 )
      product = 2 * product;
```



Ripetizione Counter-Controlled

- Ripetizione Counter-controlled
  - Ciclo ripetuto fino a quando un contatore raggiunge un certo valore
  - Il numero di ripetizioni è noto a priori
  - Esempio: 10 studenti rispondono ad un quiz. I risultati (interi tra 0 e 100) per questo quiz sono disponibili. Determinare la media della classe per quel quiz
  - Pseudocodice:

```
Set total to zero
   Set grade counter to one
While grade counter is less than or equal to ten
   Input the next grade
   Add the grade into the total
   Add one to the grade counter
Set the class average to the total divided by ten
   Print the class average
```

Outline

fig03_06.c (Part 1 of 2)

```
/* Fig. 3.6: fig03_06.c
   Class average program with counter-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter; /* number of grade to be entered next */
   int grade;   /* grade value */
    int total;   /* sum of grades input by user */
    int average; /* average of grades */

    /* initialization phase */
    total = 0;    /* initialize total */
    counter = 1; /* initialize loop counter */

    /* processing phase */
    while ( counter <= 10 ) {     /* loop 10 times */
       printf( "Enter grade: " ); /* prompt for input */
       scanf( "%d", &grade );     /* read grade from user */
       total = total + grade;     /* add grade to total */
       counter = counter + 1;    /* increment counter */
    } /* end while */

```

Outline

fig03_06.c (Part 2 of 2)

```
   /* termination phase */
   average = total / 10;         /* integer division */

   /* display result */
   printf( "Class average is %d\n", average );

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Program Output

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

Uso di una sentinella

- Il problema diviene:

  Sviluppare un programma per calcolare la media della classe su un numero qualsiasi di studenti.
  - Numero di studenti non conosciuto a priori
  - Come fa il programma a sapere quando deve terminare?
- Usare un valore *sentinella*
  - Indica "la fine dell'inserimento dei dati"
  - Il ciclo termina quando l'utente inserisce il valore sentinella
  - Il valore sentinella viene scelto in modo che non possa essere confuso con un input regolare (ad esempio `-1` in questo esempio)

Uso di una sentinella

```
Initialize total to zero
Initialize counter to zero

Input the first grade
While the user has not as yet entered the sentinel
   Add this grade into the running total
   Add one to the grade counter
   Input the next grade (possibly the sentinel)

If the counter is not equal to zero
   Set the average to the total divided by the counter
   Print the average
else
   Print “No grades were entered”
```

Outline

fig03_08.c (Part 1 of 2)

```c
/* Fig. 3.8: fig03_08.c
   Class average program with sentinel-controlled repetition */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int counter;   /* number of grades entered */
   int grade;     /* grade value */
    int total;     /* sum of grades */

    float average; /* number with decimal point for average */

    /* initialization phase */
    total = 0;     /* initialize total */
    counter = 0;   /* initialize loop counter */

    /* processing phase */
    /* get first grade from user */
    printf( "Enter grade, -1 to end: " );    /* prompt for input */
    scanf( "%d", &grade );                   /* read grade from user */

    /* loop while sentinel value not yet read from user */
    while ( grade != -1 ) {
       total = total + grade;                /* add grade to total */
       counter = counter + 1;                /* increment counter */

```

Outline

fig03_08.c (Part 2 of 2)

```c
      printf( "Enter grade, -1 to end: " ); /* prompt for input */
      scanf("%d", &grade);                  /* read next grade */
   } /* end while */

   /* termination phase */
   /* if user entered at least one grade */
   if ( counter != 0 ) {

      /* calculate average of all grades entered */
      average = ( float ) total / counter;

      /* display average with two digits of precision */
      printf( "Class average is %.2f\n", average );
   } /* end if */
   else { /* if no grades were entered, output message */
      printf( "No grades were entered\n" );
   } /* end else */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Outline

Program Output

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

Strutture di controllo innestate

- Problema
  - Un college ha una lista di risultati di test (`1` = superato, `2` = fallito) per 10 studenti
  - Scrivere un programma che analizzi i risultati
    - Se più di 8 studenti superano il test, allora stampa "Insegnamento OK"
- Nota
  - Il programma deve processare 10 resultati del test
    - Sarà usato un ciclo counter-controlled
  - Possono essere usati due contatori
    - Uno per il numero di "superato", uno per il numero di "fallito"
  - Ogni risultato del test result è un numero —`1` oppure `2`
    - Se il numero non è `1`, assumiamo sia `2`

Strutture di controllo innestate

- Top level outline

```
Analyze exam results and decide if tuition should be raised
```

- First Refinement

```
Initialize variables
Input the ten quiz grades and count passes and failures
Print a summary of the exam results and decide if tuition should
  be raised
```

- Refine *Initialize variables* to

```
Initialize passes to zero
Initialize failures to zero
Initialize student counter to one
```

Strutture di controllo innestate

- Refine *Input the ten quiz grades and count passes and failures* to

```
While student counter is less than or equal to ten
  Input the next exam result
  If the student passed
    Add one to passes
  else
    Add one to failures
  Add one to student counter
```

- Refine *Print a summary of the exam results and decide if tuition should be raised* to

```
Print the number of passes
Print the number of failures
If more than eight students passed
  Print "Raise tuition"
```

Strutture di controllo innestate

```
Initialize passes to zero
Initialize failures to zero
Initialize student to one

While student counter is less than or equal to ten
    Input the next exam result

    If the student passed
        Add one to passes
    else
        Add one to failures

    Add one to student counter


Print the number of passes
Print the number of failures
If more than eight students passed
    Print "Raise tuition"
```

Outline

fig03_10.c (Part 1 of 2)

```
/* Fig. 3.10: fig03_10.c
   Analysis of examination results */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   /* initialize variables in definitions */
   int passes = 0;   /* number of passes */
   int failures = 0; /* number of failures */
   int student = 1;  /* student counter */
   int result;       /* one exam result */

   /* process 10 students using counter-controlled loop */
   while ( student <= 10 ) {

      /* prompt user for input and obtain value from user */
      printf( "Enter result ( 1=pass,2=fail ): " );
      scanf( "%d", &result );

      /* if result 1, increment passes */
      if ( result == 1 ) {
         passes = passes + 1;
      } /* end if */
```

Outline

fig03_10.c (Part 2 of 2)

```
      else { /* otherwise, increment failures */
         failures = failures + 1;
      } /* end else */

      student = student + 1; /* increment student counter */
   } /* end while */

   /* termination phase; display number of passes and failures */
   printf( "Passed %d\n", passes );
   printf( "Failed %d\n", failures );

   /* if more than eight students passed, print "raise tuition" */
   if ( passes > 8 ) {
      printf( "Raise tuition\n" );
   } /* end if */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Outline

Program Output

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Passed 9
Failed 1
Raise tuition
```

Operatori di assegnamento

- Gli operatori di assegnamento abbreviano le espressioni di assegnamento

  `c = c + 3;`

  può essere abbreviato in `c += 3`
- Statement nella forma

  variable = *variable operator expression*;

  possono essere riscritti nella forma

  variable ***`operator`***= *expression*;
- Esempi di altri operatori di assegnamento:

```
d -= 4    (d = d - 4)
e *= 5    (e = e * 5)
f /= 3    (f = f / 3)
g %= 9    (g = g % 9)
```

Operatori di assegnamento

Siano: `int c = 3, d = 5, e = 4, f = 6, g = 12;`

Operatore di assegnamento	Esempio di espressione	Spiegazione	Assegna
`+=`	`c += 7`	`c = c + 7`	`10` a `c`
`-=`	`d -= 4`	`d = d - 4`	`1` a `d`
`*=`	`e *= 5`	`e = e * 5`	`20` a `e`
`/=`	`f /= 3`	`f = f / 3`	`2` a `f`
`%=`	`g %= 9`	`g = g % 9`	`3` a `g`

Fig. 3.11 Operatori aritmetici di assegnamento.

Operatori di Incremento e Decremento

- Operatore di Incremento (`++`)
  - Può essere usato invece di `c+=1`
- Operatore di Decremento (`--`)
  - Può essere usato invece di `c-=1`
- Preincremento
  - L'operatore è usato prima della variabile (`++c` or `--c`)
  - La variable è cambiata prima che l'espressione che la contiene sia valutata
- Postincremento
  - L'operatore è usato dopo la variabile (`c++` or `c--`)
  - L'espressione viene eseguita prima che la variabile sia cambiata

Operatori di Incremento e Decremento

- Se `c` è uguale a `5`, allora

  ```
  printf( "%d", ++c );
  ```
  - Stampa `6`

  ```
  printf( "%d", c++ );
  ```
  - Stampa `5`
  - In entrambi i casi, `c` ha il valore `6`
- Quando la variabile non è in un'espressione
  - Preincremento e postincremento hanno lo stesso effetto

  ```
  ++c;
  printf( “%d”, c );
  ```
  - Ha lo stesso effetto di

  ```
  c++;
  printf( “%d”, c );
  ```

```
/* Fig. 3.13: fig03_13.c
   Preincrementing and postincrementing */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int c;                 /* define variable */

   /* demonstrate postincrement */
   c = 5;                 /* assign 5 to c */
   printf( "%d\n", c );   /* print 5 */
   printf( "%d\n", c++ ); /* print 5 then postincrement */
   printf( "%d\n\n", c ); /* print 6 */

   /* demonstrate preincrement */
   c = 5;                 /* assign 5 to c */
   printf( "%d\n", c );   /* print 5 */
   printf( "%d\n", ++c ); /* preincrement then print 6 */
   printf( "%d\n", c );   /* print 6 */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

Outline

fig03_13.c

```
5
5
6

5
6
6
```

▲ ▼ Outline

Program Output