

Capitolo 3 – Sviluppo di Programmi Strutturati

1

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

2

- 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

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



Strutture di controllo

3

- 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: i f, i f...el se, e swi tch
 - ITERAZIONE: Il C ne ha tre tipi: whi l e, do...whi l e e for

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



i f Selection Statement

4

- 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 (fal se), print statement ignorato e il programma va al prossimo statement
 - L'indentazione rende il programma più leggibile
 - Il C ignora gli spazi bianchi

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



if Selection Statement

- Statement in C:

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



if...else Selection Statement

- if
 - Esegui 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

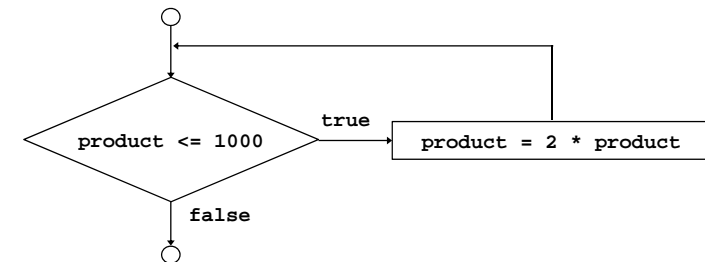
- Ciclo while ripetuto fino a quando la condizione diviene falsa (false)



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

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



```

1 /* Fig. 3.6: fig03_06.c
2   Class average program with counter-controlled repetition */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8     int counter; /* number of grade to be entered next */
9     int grade; /* grade value */
10    int total; /* sum of grades input by user */
11    int average; /* average of grades */
12
13    /* initialization phase */
14    total = 0; /* initialize total */
15    counter = 1; /* initialize loop counter */
16
17    /* processing phase */
18    while ( counter <= 10 ) { /* loop 10 times */
19        printf( "Enter grade: " ); /* prompt for input */
20        scanf( "%d", &grade ); /* read grade from user */
21        total = total + grade; /* add grade to total */
22        counter = counter + 1; /* increment counter */
23    } /* end while */
24

```

fig03_06.c (Part 1 of 2)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```

25 /* termination phase */
26 average = total / 10; /* integer division */
27
28 /* display result */
29 printf( "Class average is %d\n", average );
30
31 return 0; /* indicate program ended successfully */
32
33 } /* end function main */

```

fig03_06.c (Part 2 of 2)

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

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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"

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



Outline

fig03_08.c (Part 1 of 2)

```

1  /* Fig. 3.8: fig03_08.c
2     Class average program with sentinel-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int counter; /* number of grades entered */
9     int grade; /* grade value */
10    int total; /* sum of grades */
11
12    float average; /* number with decimal point for average */
13
14    /* initialization phase */
15    total = 0; /* initialize total */
16    counter = 0; /* initialize loop counter */
17
18    /* processing phase */
19    /* get first grade from user */
20    printf( "Enter grade, -1 to end: " ); /* prompt for input */
21    scanf( "%d", &grade ); /* read grade from user */
22
23    /* loop while sentinel value not yet read from user */
24    while ( grade != -1 ) {
25        total = total + grade; /* add grade to total */
26        counter = counter + 1; /* increment counter */
27

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



Outline

fig03_08.c (Part 2 of 2)

```

28    printf( "Enter grade, -1 to end: " ); /* prompt for input */
29    scanf( "%d", &grade ); /* read next grade */
30 } /* end while */
31
32 /* termination phase */
33 /* if user entered at least one grade */
34 if ( counter != 0 ) {
35
36     /* calculate average of all grades entered */
37     average = ( float ) total / counter;
38
39     /* display average with two digits of precision */
40     printf( "Class average is %.2f\n", average );
41 } /* end if */
42 else { /* if no grades were entered, output message */
43     printf( "No grades were entered\n" );
44 } /* end else */
45
46 return 0; /* indicate program ended successfully */
47
48 } /* end function main */

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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 risultati 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

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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*

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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"*

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



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"

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



```

1 /* Fig. 3.10: fig03_10.c
2  Analysis of exam nation results */
3 #include <stdio.h>
4
5 /* function main begins program execution */
6 int main()
7 {
8     /* initialize variables in definitions */
9     int passes = 0; /* number of passes */
10    int failures = 0; /* number of failures */
11    int student = 1; /* student counter */
12    int result; /* one exam result */
13
14    /* process 10 students using counter-controlled loop */
15    while ( student <= 10 ) {
16
17        /* prompt user for input and obtain value from user */
18        printf( "Enter result ( 1=pass, 2=fail ): " );
19        scanf( "%d", &result );
20
21        /* if result 1, increment passes */
22        if ( result == 1 ) {
23            passes = passes + 1;
24        } /* end if */

```



Outline

25

fig03_10.c (Part 1 of 2)

```

25     else { /* otherwise, increment failures */
26         failures = failures + 1;
27     } /* end else */
28
29     student = student + 1; /* increment student counter */
30 } /* end while */
31
32 /* termination phase; display number of passes and failures */
33 printf( "Passed %d\n", passes );
34 printf( "Failed %d\n", failures );
35
36 /* if more than eight students passed, print "raise tuition" */
37 if ( passes > 8 ) {
38     printf( "Raise tuition\n" );
39 } /* end if */
40
41 return 0; /* indicate program ended successfully */
42
43 } /* end function main */

```



Outline

26

fig03_10.c (Part 2 of 2)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```

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

```



Outline

27

Program Output

```

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

```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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)
```

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.



28

Operatori di assegnamento

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

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



```



1  /* Fig. 3.13: fig03_13.c
2     Preincrementing and postincrementing */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int c;           /* define variable */
9
10     /* demonstrate postincrement */
11     c = 5;          /* assign 5 to c */
12     printf( "%d\n", c ); /* print 5 */
13     printf( "%d\n", c++ ); /* print 5 then postincrement */
14     printf( "%d\n\n", c ); /* print 6 */
15
16     /* demonstrate preincrement */
17     c = 5;          /* assign 5 to c */
18     printf( "%d\n", c ); /* print 5 */
19     printf( "%d\n", ++c ); /* preincrement then print 6 */
20     printf( "%d\n", c ); /* print 6 */
21
22     return 0; /* indicate program ended successfully */
23
24 } /* end function main */

```

 [Outline](#)
fig03_13.c

5
5
6

5
6
6

 Outline

Program Output