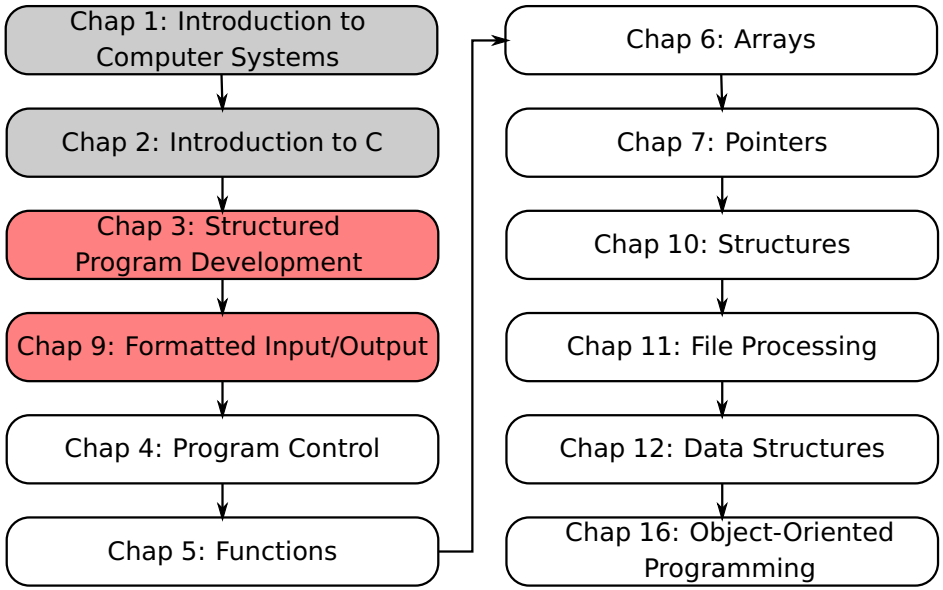
Computer Programming 143 – Lecture 6
Structured Program Development III

Electrical and Electronic Engineering Department
University of Stellenbosch

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan

Module Overview



Lecture Overview

3.10 Nested Control Structures I

Problem statement

Develop a program that would count and display the number of students that have passed and the number of students that have failed from a list of exam results for 10 students. If more than 8 students have passed, display "Bonus to instructor!"

Top-level pseudocode

Analyse exam results and decide if instructor should receive a bonus

First refinement

Initialise variables
Input the 10 exam grades and count passes and failures
Display a summary of the exam results and decide if instructor should receive a bonus

3.10 Nested Control Structures II

Second refinement

Initialise passes to 0
Initialise failures to 0
Initialise student counter to 1

While student counter is less than or equal to 10
    Input the next exam result
    If the student passed
        Add 1 to passes
    else
        Add 1 to failures
    Add 1 to student counter

3.10 Nested Control Structures III

Second refinement (cont'd...)

Display the number of passes
Display the number of failures
If more than 8 students passed
    Display "Bonus to instructor!"

3.10 Nested Control Structures IV

C code

```c
/* Nested Control Structures
 * Copied from Deitel & Deitel Fig. 3.10
 */
#include <stdio.h>

int main( void )
{
   // initialise 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

   setbuf(stdout, 0);  // fix Eclipse for scanf
```


3.10 Nested Control Structures V

C code (cont'd...)

```c
   // process 10 students using counter-controlled loop
   while ( student <= 10 ) {
      // prompt user for input and obtain value from user
      printf( "Enter result for student %d (1=pass;2=fail): ", student);
      scanf( "%d", &result );

      if ( result == 1 ) { // if result is 1, increment passes
         passes = passes + 1;
      } // end if
      else { // otherwise, increment failures
         failures = failures + 1;
      } // end else

      student = student + 1; // increment student counter
   }
```

3.10 Nested Control Structures VI

C code (cont'd...)

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

   // if more than 8 students passed, display "Bonus to instructor!"
   if ( passes > 8 ) {
      printf( "Bonus to instructor!\n" );
   } // end if

   return 0; // indicate program ended successfully
} // end main function
```


3.10 Nested Control Structures VII

Output

```
Enter result for student 1 (1=pass;2=fail): 1
Enter result for student 2 (1=pass;2=fail): 2
Enter result for student 3 (1=pass;2=fail): 2
Enter result for student 4 (1=pass;2=fail): 1
Enter result for student 5 (1=pass;2=fail): 1
Enter result for student 6 (1=pass;2=fail): 1
Enter result for student 7 (1=pass;2=fail): 2
Enter result for student 8 (1=pass;2=fail): 1
Enter result for student 9 (1=pass;2=fail): 1
Enter result for student 10 (1=pass;2=fail): 2
Passed 6
Failed 4
```


3.11 Assignment Operators I

- Assignment operators abbreviate assignment expressions

  `c = c + 3;`

  can be abbreviated as `c += 3;` using the addition assignment operator
- Statements of the form

  variable = *variable* `operator` *expression;*

  can be rewritten as

  variable `operator=` *expression;*
- Examples of other assignment operators:

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


3.12 Increment and Decrement Operators I

Increment operator (++)

- Can be used instead of `c += 1`

  `c = c + 1` ⇒ `c += 1` ⇒ `c++`

Decrement operator (--)

- Can be used instead of `c -= 1`

  `c = c - 1` ⇒ `c -= 1` ⇒ `c--`

Preincrement

- Operator is used before the variable (`++c` or `--c`)
- Variable is changed before the expression it is in is evaluated

Postincrement

- Operator is used after the variable (`c++` or `c--`)
- Expression executes before the variable is changed

3.12 Increment and Decrement Operators II

The increment and decrement operators

Operator	Sample	Explanation
++	++a	Increment a by 1 then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	--b	Decrement b by 1 then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

3.12 Increment and Decrement Operators III

Variable in an expression:

```
int j;
j = 5;
printf( "%d", ++j );
```

- prints 6

```
int j;
j = 5;
printf( "%d", j++ );
```

- prints 5

When variable not in an expression:

- Preincrementing and postincrementing have the same effect

9.1-9.11 Formatted Input/Ouput I

printf statement using different types

Declaration&assignment	printf Statement	Output
int myint = 45;	printf("%d", myint);	45
float myfloat = 79.54;	printf("%f", myfloat);	79.540001
float myfloat = 79.54;	printf("%e", myfloat);	7.954000e+001
char mychar = 'a';	printf("%c", mychar);	a
char mychar = 'a';	printf("%d", mychar);	97
char mychar = 98;	printf("%c", mychar);	b
char mychar = 98;	printf("%d", mychar);	98
char mystring[] = "Hello";	printf("%s", mystring);	Hello

Refer to Chapter 9 for more printing options

9.1-9.11 Formatted Input/Ouput II

Printing integers with field width

- Field width specifies the minimum space that the displayed integer should occupy
- **C code:**

```
printf( "%4d\n", 1 );
printf( "%4d\n", 1234 );
printf( "%4d\n", 12345 );
```

- **Output:**

```
   1
1234
12345
```

9.1-9.11 Formatted Input/Ouput III

Printing floating-point values with field width and precision

- Field width specifies the minimum space that the displayed floating point number should occupy
- Precision specifies the number of digits to appear after the decimal point
- **C code:**

```
printf( "%.3f\n", 3.8663 );
printf( "%9f\n", 3.8663 );
printf( "%9.3f\n", 3.8663 );
```

- **Output:**

```
3.866
 3.866300
    3.866
```


9.1-9.11 Formatted Input/Ouput IV

scanf statement using different types

Declaration	scanf Statement
int myint;	scanf("%d", &myint);
float myfloat;	scanf("%f", &myfloat);
double mydouble;	scanf("%lf", &mydouble);

Refer to Chapter 9 for more user input options, including input of characters and strings

Perspective

Today

Structured Program Development III

- Program design 3: nested control structures
- Assignment, increment and decrement operators
- Formatted input/output

Next lecture

Program Control

- 'for' repetition structure

Homework

1. Study Sections 3.10-3.12 in Deitel & Deitel
2. Chapter 9 in Deitel & Deitel
3. Do Self Review Exercises 3.2, 3.3, 3.6-3.8 in Deitel & Deitel
4. Do Exercises 3.10(c)&(d), 3.29 in Deitel & Deitel