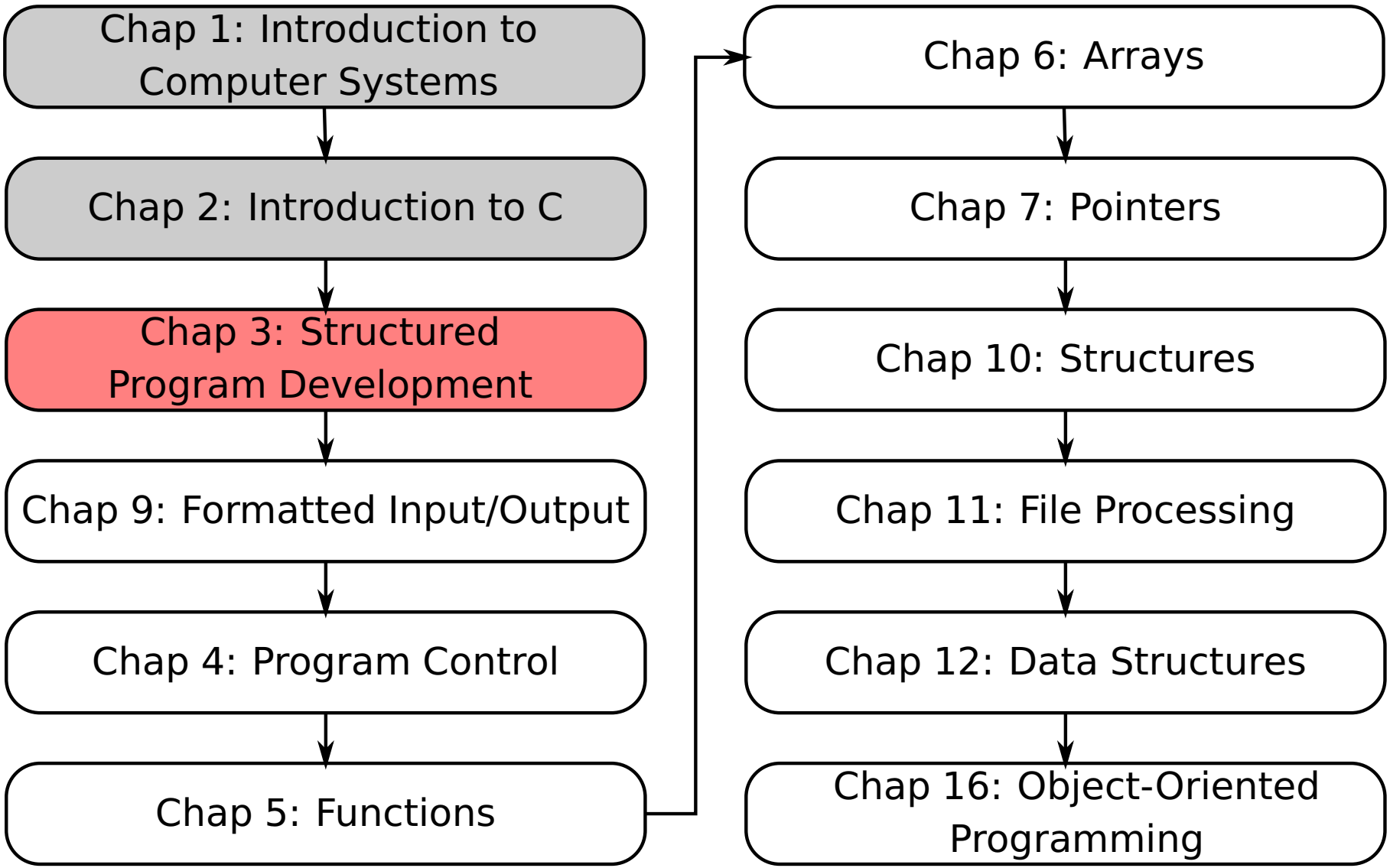
Computer Programming 143 – Lecture 5
Structured Program Development II

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Module Overview



Lecture Overview

1. Review of Structured Programming so far (3.1-3.6)
2. The ‘while’ Repetition Statement (3.7)
3. Program Design 1: Counter-Controlled Repetition (3.8)
4. Program Design 2: Sentinel-Controlled Repetition (3.9)

Review of Structured Programming I

Short overview of structured programming so far

- An **algorithm** is a procedure for solving a problem in terms of
  - actions to be executed
  - the order in which to execute the actions
- **Pseudocode** and **flow diagrams** describe algorithms
- Structured program development:
  - Before one starts writing a program
  - Design an algorithm using top-down, stepwise refinement
  - Consisting of only a few control structures
  - Only then implement the algorithm in a program
  - C has 7 control structures: 1 sequence, 3 selection and 3 repetition structures
- Previous lecture: sequence structure, ‘if’ statement and ‘if...else’ statement

3.7 The ‘while’ Repetition Statement I

Repetition structure

- Programmer specifies an action to be repeated while some condition remains `true`
- **Pseudocode:**

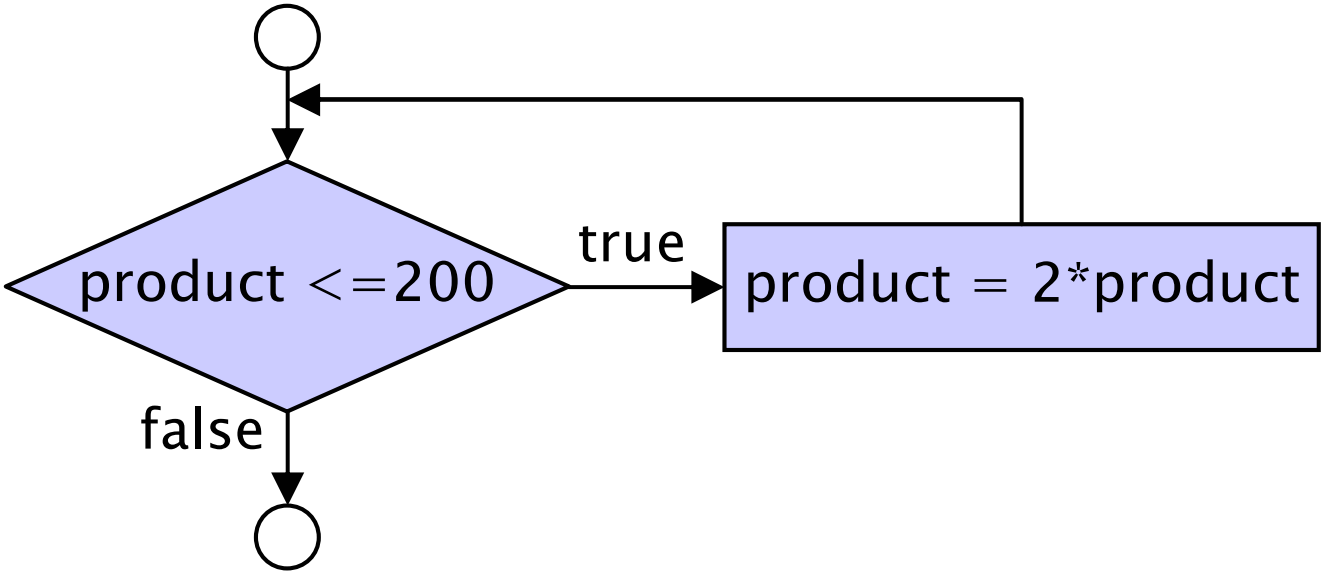
  While there are more items on my shopping list
  Purchase next item and cross it off my list

- **`while`** loop repeated until condition becomes `false`

Example:

```
int product = 2;
while ( product <= 200 ) {
  product = 2 * product;
}
```

3.7 The ‘while’ Repetition Statement II



3.8 Counter-Controlled Repetition I

Problem statement

A class of ten students took a test. The grades (integers in the range 0 to 100) for this test are available to you. Determine the class average for the test.

Top-level pseudocode

Determine the class average of the test

First refinement

Initialise variables
Input and sum 10 test grades
Calculate and print class average

3.8 Counter-Controlled Repetition II

Second refinement

Set total to zero
Set counter to one

While counter is less than or equal to 10
    Input next grade
    Add the grade into the total
    Add one to the grade counter

Set the class average to the total divided by 10
Print the class average

3.8 Counter-Controlled Repetition III

C code

```c
/* Description: Class average program with counter-controlled repetition
 * Copied from Deitel & Deitel Fig. 3.6 */

#include <stdio.h>

// Function main begins program execution
int main( void )
{
  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
```

3.8 Counter-Controlled Repetition IV

C code (cont'd...)

```c
// Initialisation phase
total = 0;    // initialise total
counter = 1;  // initialise 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
```

3.8 Counter-Controlled Repetition V

C code (cont'd...)

```
  // Termination phase
  average = total / 10;          // integer division

  printf( "Class average is %d\n", average ); // display result
  return 0;                      // program ended successfully
} // end function main
```

3.8 Counter-Controlled Repetition VI

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
```

3.8 Sentinel-Controlled Repetition I

Problem statement

Develop a class averaging program that will process an arbitrary number of grades each time the program is run.

Top-level pseudocode

Determine the class average of the test

First refinement

Initialise variables
Input, sum and count the test grades
Calculate and print class average

3.8 Sentinel-Controlled Repetition II

Second refinement

Set total to zero
Set counter to zero

Input the first grade
While the user has not entered the sentinel
    Add the grade into the total
    Add one to the grade counter
    Input 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 a message that no grades were entered

3.8 Sentinel-Controlled Repetition III

C code

```c
/* Class averaging program with sentinel-controlled repetition
 * Copied from Deitel & Deitel Fig. 3.8 */
#include <stdio.h>

// Function main begins program execution
int main( void )
{
  int counter;   // number of grades entered
  int grade;     // grade value
  int total;     // sum of grades
  float average; // number with decimal point for average
```

3.8 Sentinel-Controlled Repetition IV

C code (cont'd...)

```
total = 0;      // initial total
counter = 0;    // initialise loop counter

// 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;                 // ad grade to total
  counter = counter + 1;                 // increment counter
  // get next grade from user
  printf( "Enter grade, -1 to end: " ); // prompt for input
  scanf( "%d", &grade );                 // read next grade
} // end while
```

3.8 Sentinel-Controlled Repetition V

C code (cont'd...)

```
  // Termination phase
  // if user entered at least one grade
  if ( counter != 0 ) {
    average = (float) total / counter;            // calculate average
     printf( "Class average is %.2f\n", average ); // display 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
}
```

3.8 Sentinel-Controlled Repetition VI

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
```

Floating-point numbers I

Floating-point numbers

- Describes real numbers
- Declaration:

```
float average;
```

Casting

- Conversion from one data type to another

```
// total and counter are integers; average is a float
average = ( float ) total / counter;
```

- Explicit conversion
- Implicit conversion

Floating-point numbers II

Display of floating-point number

```
x = 3.446; // Assign 3.446 to float variable x

printf( "%f\n", x );   // Displays 3.446000
printf( "%.2f\n", x ); // Displays 3.45
printf( "%.1f\n", x ); // Displays 3.4
```

Perspective

Today

Structured Program Development II

- ‘while’ repetition structure
- Counter-controlled repetition
- Sentinel-controlled repetition
- Floating-point numbers

Next lecture

Structured Program Development III

- Nested control structures
- Assignment, increment and decrement operators
- Formatted input/output

Homework

1. Study Sections 3.7-3.9 in Deitel & Deitel
2. Do Self Review Exercises 3.1, 3.4, 3.5, 3.9 in Deitel & Deitel
3. Do Exercises 3.12, 3.14(c) in Deitel & Deitel