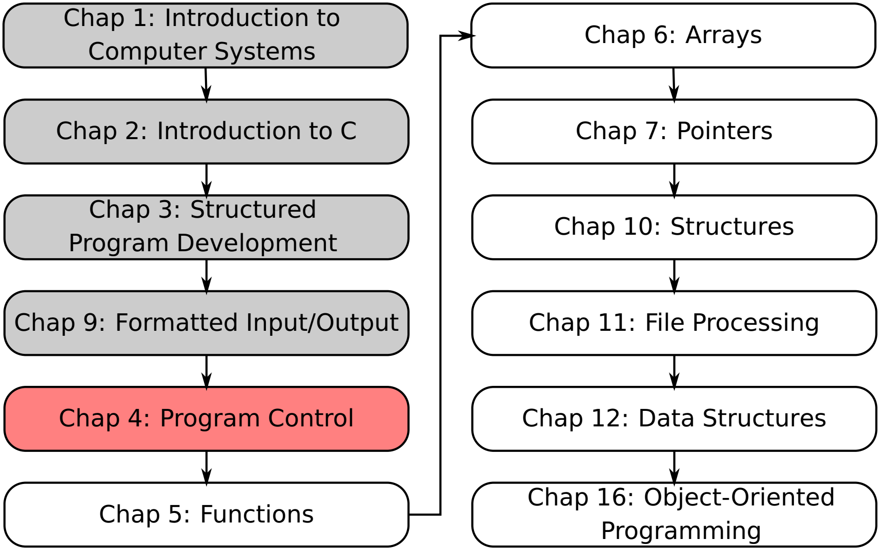
Computer Programming 143 – Lecture 8
Program Control II

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


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

Lecture Overview

1. The `do...while` Repetition Statement (4.8)
2. Logical Operators (4.10)
3. Confusing Equality and Assignment Operators (4.11)

4.8 The **do...while** Repetition Statement I

The **do...while** repetition statement

- Similar to the while structure
- Condition for repetition tested after the body of the loop is performed
  - All actions are performed at least once
- Format:

```
do {
    statement(s);
} while ( condition );
```

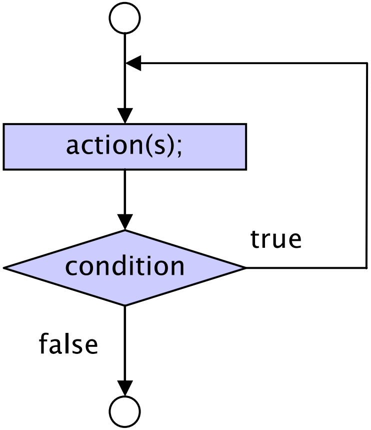
4.8 The **do...while** Repetition Statement II

Example:

```
counter = 1;
do {
  printf( "%d  ", counter );
} while (++counter <= 10);
```

- Prints the integers from 1 to 10

4.8 The **do...while** Repetition Statement III



4.10 Logical Operators I

&& (logical AND)

- Returns **`true`** if both conditions are **`true`**

|| (logical OR)

- Returns **`true`** if either of its conditions are **`true`**

! (logical NOT, logical negation)

- Reverses the truth/falsity of its condition
- Unary operator, has one operand

Useful for conditions in loops

`if ((2<x) && (x<7))` *`equivalent to`* $(2 < x < 7)$

`or to calculate a student's final mark:`

```
Pass = (0.1*S+0.4*A1+0.5*A2>=50)||(0.1*S+0.4*A1+0.5*A3>=50);
```

4.10 Logical Operators II

&& (logical AND)

0	&&	0	=	0
nonzero	&&	0	=	0
0	&&	nonzero	=	0
nonzero	&&	nonzero	=	1

|| (logical OR)

0	||	0	=	0
nonzero	||	0	=	1
0	||	nonzero	=	1
nonzero	||	nonzero	=	1

4.10 Logical Operators III

! (logical NOT, logical negation)

!0	=	1
!nonzero	=	0

4.10 Example: `do...while` and logical operators I

Problem statement

As part of a survey, read the user's level of happiness on a scale of 1 to 10 and inform the user whether he/she has a normal (3 to 8) or abnormal (1, 2, 9 or 10) level of happiness. Ensure that the user enters a valid level.

Pseudocode

```
Do
    Read user's level of happiness
Until the user has input a valid level

If the user has a normal level of happiness
    Inform the user that his/her happiness level is normal
Else
    Inform the user that his/her happiness level is abnormal
```

4.10 Example: do. . .while and logical operators II

C code

```c
/* HappinessMeter.c
 * Program that measures your level of happiness */
#include <stdio.h>
#include <stdlib.h>
int main( void )
{
  int happiness; // the store of happiness
  // Repeatedly reads happiness level from user until 1 <= level <= 10
  do {
    printf( "Enter your happiness level on a scale of 1 to 10: " );
    scanf( "%d", &happiness ); // reads the user's level of happiness
  } while ( (happiness < 1 )|| (happiness > 10) );
  // repeat if invalid level entered
```

4.10 Example: do...while and logical operators III

C code

```c
  if ( (happiness >= 3) && (happiness <= 8) ) { // if happiness in [3..8]
    printf( "You are normal - congratulations!");
  } // end if
  else { // if happiness is not in [3..8]
    printf( "You are either very happy or very sad - seek help!\n");
  } // end else

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

4.10 Example: do. . .while and logical operators IV

Output

```
Enter your level of happiness on a scale of 1 to 10: 0
Enter your level of happiness on a scale of 1 to 10: 11
Enter your level of happiness on a scale of 1 to 10: 1
You are either very happy or very sad - seek help!
```

Output

```
Enter your level of happiness on a scale of 1 to 10: 5
You are normal - congratulations!
```

4.10 Example: nested loops and logical operators I

What does the following code do?

```c
int i, j;
for ( i = 1; i <= 7; i++ ) {
  for ( j = 1; j <= 7; j++ ) {
    if ( !( i == 4 || j == 4 ) ) {
      printf( "* " );
    }
    else {
      printf( "  ");
    }
  }
  printf( "\n" );
}
```

4.10 Example: nested loops and logical operators II

Output

```
* * *   * * *
* * *   * * *
* * *   * * *

* * *   * * *
* * *   * * *
* * *   * * *
```

4.11 Confusing Equality and Assignment Operators I

Equality operator

```c
if ( payCode == 4 ) {
  printf( "You get a bonus!" );
}
```

- Displays "You get a bonus!" if variable payCode has value 4

Assignment operator in stead of equality operator

```c
if ( payCode = 4 ) {
  printf( "You get a bonus!" );
}
```

- Stores 4 in variable payCode and displays "You get a bonus!"

4.11 Confusing Equality and Assignment Operators II

Equality operator in stead of assignment operator

```
x = 1;
```

- Assigns a value of 1 to variable x

```
x == 1;
```

- Tests if variable x is equal to 1, but does not change its value

Perspective

Today

Program Control II

- `do...while` repetition structure
- Logical operators
- Confusing equality and assignment operators

Next lecture

Program Control III

- `switch` selection structure

Homework

1. Study Sections 4.8, 4.10, 4.11 in Deitel & Deitel
2. Do Self Review Exercises 4.2(c)&(d) in Deitel & Deitel
3. Do Exercises 4.5(f), 4.29, 4.36 in Deitel & Deitel