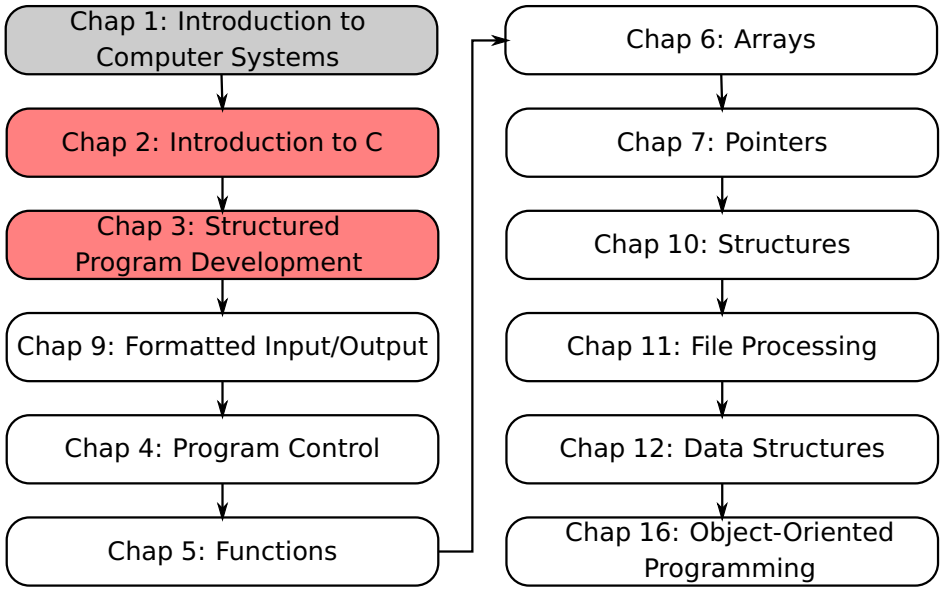
Computer Programming 143 – Lecture 4
Structured Program Development I

Electrical and Electronic Engineering Department
University of Stellenbosch

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan

Module Overview



Lecture Overview

2.6 Decision Making I

Equality operators

- == Is equal to
- != Is not equal to

Relational operators

- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to

2.6 Decision Making II

Examples

```
if ( 2 < 3 ) {
    printf( "2 is less than 3" );
}
if ( 2 != 3 ) {
    printf( "2 is not equal to 3" );
}
```

We can now write programs that:

- Perform actions (calculations, input/output of data)
- Perform decisions

3.1 Introduction to Structured Programming Development

Before writing a program

- Have a thorough understanding of the problem
- Then carefully plan an approach for solving the problem

While writing a program

- Know what “building blocks” are available
- Use good programming principles

3.2 Algorithms

Computing problems

- All can be solved by executing a series of actions written in a specific order

Algorithm: procedure in terms of

- Actions to be executed
- The order in which these actions are to be executed

Program control

- Specifies the order in which statements are to be executed

3.3 Pseudocode

Pseudocode

- Artificial, informal language that helps us develop algorithms
- Similar to everyday English
- Not actually executed on computers
- Helps us “think out” a program before writing it
  - Easy to convert into a corresponding programming language
  - Consists only of executable statements

3.4 Control Structures I

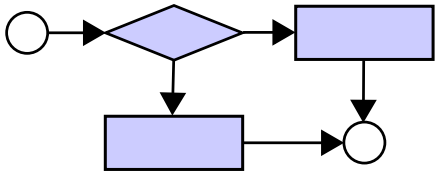
Transfer of control

- When the next statement executed is not the next one in sequence
- Overuse of `goto` statements led to many problems

Bohm and Jacopini's research

- All programs written in terms of 3 control structures
  - **Sequence structures:** Built into C. Programs executed sequentially by default
  - **Selection structures:** C has three types: **`if`**, **`if...else`**, and **`switch`**
  - **Repetition structures:** C has three types: **`while`**, **`do...while`**, and **`for`**

3.4 Control Structures II



Flow diagram

- Graphical representation of an algorithm
- Drawn using certain special-purpose symbols connected by arrows called flow lines
- Symbols:
  - Rectangle: Indicates any type of action
  - Oval: Indicates the beginning or end of a program or a section of code
  - Diamond: Indicates decision is to be made

3.4 Control Structures III

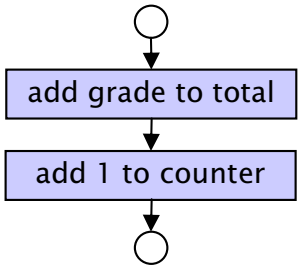
Single-entry/single-exit control structures

- Connect exit point of one control structure to entry point of the next (control-structure stacking)
- Makes programs easy to build

3.4 Control Structures IV

Control Structure 1: Sequential execution

- Statements executed one after the other in the order they were written




```
total = total + grade;
counter = counter + 1;
```

3.5 The ‘if’ Selection Statement I

Control structure 2: the ‘if’ statement

- Used to choose among alternative courses of action

Pseudocode:

If student’s grade is greater than or equal to 50
Print “Passed”

- If the condition is `true`:
  - Print statement executed and program goes on to the next statement
- If the condition is `false`:
  - Print statement is ignored and the program goes onto the next statement

3.5 The ‘if’ Selection Statement II

C code:

```
if ( grade >= 50 ) {
    printf( "Passed\n" );
}
```

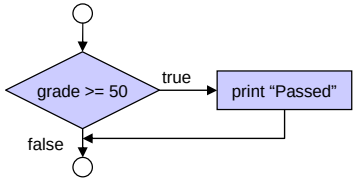
- C code corresponds closely to the pseudocode
- A set of braces may be removed when it contains only one statement

C code:

```
if ( grade >= 50 )
   printf( "Passed\n" );
```


3.5 The ‘if’ Selection Statement III

- **`if`** statement is a single-entry/single-exit structure



- A decision can be made on any expression.
  - zero - false
  - nonzero - true
- Example:
  - a - b is true if a is not equal to b (valid expression but bad programming)
  - a != b is equivalent, but better programming

3.5 The ‘if’ Selection Statement IV

Assignment example:

```
int j;
j = ( 50 > 60 );
if ( j ) {
    printf( "j is true\n" );
}
```

Example program: Using 'if' statements, relational and equality operators I

```c
/* description: Making decisions
 * version: 1
 * date: 18/07/2011
 * author: CvD
 */
#include <stdio.h>

/* function main() begins program execution */
int main( void )
{
   int num1; // first number to be read from user
   int num2; // second number to be read from user

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


Example program: Using 'if' statements, relational and equality operators II

(cont'd. . .)

```c
   printf( "Enter two integers and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d", &num1 ); // read first integer
   scanf( "%d", &num2 ); // read second integer

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   }
   if ( num1 < num2 ) {
      printf( "%d is smaller than %d\n", num1, num2 );
   }
   if ( num1 > num2 ) {
      printf( "%d is greater than %d\n", num1, num2 );
   }
   return 0; // program ended successfully
} /* end main */
```


3.6 The 'if...else' Statement I

Control structure 3: the 'if...else' statement

- Specifies an action to be performed both when the condition is `true` and when it is `false`
- **Pseudocode:**

  If student's grade is greater than or equal to 50
  Print "Passed"
  else
  Print "Failed"

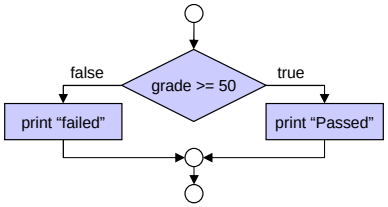
- Note spacing/indentation conventions

C code:

```c
if ( grade >= 50 )
   printf( "Passed\n" );
else
   printf( "Failed\n" );
```


3.6 The 'if...else' Statement II

Flow diagram of the 'if...else' selection statement



Nested 'if...else' statements

- Test multiple cases by placing **`if...else`** selection statements inside **`if...else`** selection statement
- Once condition is met, rest of statements skipped

3.6 The ‘if...else’ Statement III

Pseudocode:

If grade is greater than or equal to 80
Print “A”
else
If grade is greater than or equal to 70
Print “B”
else
If grade is greater than or equal to 60
Print “C”
else
If grade is greater than or equal to 50
Print “D”

3.6 The ‘if...else’ Statement IV

C code

```
if ( grade >= 80 ) {
   printf( "A\n" );
} // end if
else {
   if ( grade >= 70 ) {
      printf( "B\n" );
   } // end if
   else {
      if ( grade >= 60 ) {
         printf( "C\n" );
      } // end if
      else {
         if ( grade >= 50 ) {
            printf( "D\n" );
         } // end if
      } // end else
   } // end else
} // end else
```


3.6 The ‘if...else’ Statement V

Testing multiple conditions

- The following ‘if’ statement is *wrong*:

```
if ( 0 < x < 5 )
   printf( "x lies between 0 and 5" );
```

- Use nested ‘if’ statements (for now):

```
if ( x > 0 )
   if ( x < 5 )
      printf( "x lies between 0 and 5" );
```


3.6 The ‘if...else’ Statement VI

Compound statement

- Set of statements within a pair of braces

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

- Without the braces, the statement

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

would be executed automatically

Perspective

Today

Structured program development I

- Decision making in C
- Algorithms, pseudocode, flow diagrams and control structures
- 3 structures: sequence, ‘if’ and ‘if...else’ statements

Next lecture

Structured program development II

- ‘while’ repetition structure

Homework

1. Study Sections 2.6 and 3.1-3.6 in Deitel&Deitel
2. Do Self Review Exercises 2.3, 2.6 in Deitel&Deitel
3. Do Exercises 2.7, 3.10(a), 3.14(a)&(b) in Deitel&Deitel