Week 3: Control Structures - Selection

BJ Furman
14FEB2011
The Plan for Today

- Review control structures
  - Sequence
  - Selection
  - Repetition
- Selection structures
  - If
  - If/else
  - Switch
- Relational operators
- Selection structure example
Learning Objectives

- Explain what is meant by a control structure
- Explain the three basic types of control structures
- Determine the result of relational comparisons
- Apply the if and if/else control structures
Control Structures - Review

All programs can be written in terms of three control structures (like building blocks)

- **Sequence**
  - ‘Built-in’ to C
    - Unless otherwise directed, one statement after the next is executed

- **Selection** (three types)
  - Depending on a *condition*, *select* between one statement or another
    - If var1 is greater than 10, do *this*…, else do *that*…

- **Repetition** (three types)
  - Depending on a *condition*, execute one or more statements *repeatedly*
Selection Structure Overview

- Three kinds of selections structures
  - **if** (also called, ‘single-selection’)
    - if *condition* is true
      - Perform action
    - if *condition* is false, action is skipped
  - **if/else** (also called, ‘double-selection’)
    - if *condition* is true
      - Perform action
    - else (if *condition* is false)
      - Perform a different action (this will be skipped if condition is true)
  - **switch** (also called ‘multiple-selection’)
    - Allows selection among many actions depending on the value of a variable or expression
Single Selection IF - Flowchart

- **Decision Symbol**: Speed > 65
- **Flow Line**: From decision symbol to TRUE
- **Connector**: From TRUE to action symbol
- **Action Symbol**: Print “You’re speeding!”
- **FALSE**: From action symbol to decision symbol
Double-Selection IF - Flowchart

Print “Within limit”  
FALSE  
Speed > 65  
TRUE  
Print “Over speed limit!”
SWITCH - Flowchart

Fig. 4.8 | switch multiple-selection statement with breaks.

Adapted from Deitel & Deitel, C How to Program, 6th ed., p. 111
IF statement (single-selection)

- Syntax

```java
if(expression) /* if expression is TRUE (not equal to zero) */
    statement1; /* then execute this statement */
    statement2; /* otherwise execute this statement */
```

- Notes
  - Indent statements
  - Can have multiple statements
    - Enclose a ‘block’ of statements using `{ }` (curly braces)

```java
if( x <= 2 )
{
    statement1;
    statement2;
}
statement3;
```
IF statement example

- **Pseudocode** *(notice indentation!)*
  
  If speed is greater than 65 mph
  print “You’re speeding!”

- **C code**
  
  ```c
  if(speed > 65)
    printf("You’re speeding!\n");
  ```

- **C code with statement block**
  
  ```c
  if(speed > 65)
    /* statements below executed only if speed > 65 is true */
    {
      printf("You’re speeding!\n");
      printf("Slow down!\n");
      printf("Keep speed below 65 MPH\n");
    }
  ```
IF-ELSE statement - Double Selection

Syntax

```plaintext
if(expression) /* if expression is TRUE */
    statement1; /* execute this statement */
else
    statement2; /* else execute the following statement */
```

Notes:

- If `expression` is non-zero, `statement1` is executed, then the program continues with the statement after `statement2`, i.e., `statement2` is skipped.
- If `expression` is equal to zero, `statement1` is skipped and `statement2` is executed, then the program continues with the statement after `statement2`. 
IF-ELSE statement example

- **Pseudocode** (notice indentation!)
  
  If speed is greater than 65 mph
  
  print “Over speed limit!”

  else
  
  print “Within speed limit”

- **C code**

  ```c
  if(speed > 65)
      printf(“Over speed limit!\n”);
  else
      printf(“Within limit\n”);
  ```
Relational Operators

Important for constructing the decision expression

<table>
<thead>
<tr>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 &lt; 7 result is _____</td>
</tr>
<tr>
<td>5 &gt; 7 result is ______</td>
</tr>
<tr>
<td>7 &lt;= 7 result is _____</td>
</tr>
<tr>
<td>8 &gt;= 7 result is _____</td>
</tr>
<tr>
<td>5 == 5 result is ____</td>
</tr>
<tr>
<td>5 == 7 result is _____</td>
</tr>
<tr>
<td>8.0 == 5 result is ___</td>
</tr>
<tr>
<td>6 != 5 result is _____</td>
</tr>
</tbody>
</table>

### Operations and Associativity

<table>
<thead>
<tr>
<th>Operations</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>::</td>
<td></td>
</tr>
<tr>
<td>() []</td>
<td>left to right</td>
</tr>
<tr>
<td>Function_name()</td>
<td>right to left</td>
</tr>
<tr>
<td>. -&gt;</td>
<td>left to right</td>
</tr>
<tr>
<td><code>!</code> ++)</td>
<td>left to right</td>
</tr>
<tr>
<td>&amp; (type) sizeof</td>
<td>right to left</td>
</tr>
<tr>
<td>* / % . * . /</td>
<td>left to right</td>
</tr>
<tr>
<td>+ -</td>
<td>left to right</td>
</tr>
<tr>
<td>&lt;&lt; &gt;&gt;</td>
<td>left to right</td>
</tr>
<tr>
<td>&lt;= &gt;=</td>
<td>left to right</td>
</tr>
<tr>
<td>== !=</td>
<td>left to right</td>
</tr>
<tr>
<td>&amp;</td>
<td>left to right</td>
</tr>
<tr>
<td>^</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>left to right</td>
</tr>
<tr>
<td>^^</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>?:</td>
<td>right to left</td>
</tr>
<tr>
<td>= += -= *= /= %=</td>
<td>=</td>
</tr>
<tr>
<td>&lt;&lt;= &gt;&gt;=</td>
<td>right to left</td>
</tr>
<tr>
<td>,</td>
<td>left to right</td>
</tr>
</tbody>
</table>

Adapted from H. Cheng chap04.ppt, slide 5
Compound Condition - &&

- Logical operators for more complex decisions
  - Logical AND operator && (double ampersand)
    - `if(switch1 == 0 && switch2 == 1)`
      - turn on the motor
    - The condition evaluates to TRUE if and only if BOTH expressions on either side of && evaluate to TRUE
      - Note operator precedence
    - Otherwise condition evaluates to FALSE
      - Beware of ‘short circuit evaluation’
        - Make the condition most likely to be FALSE the left-most condition
Compound Condition - | | 

- Logical operators for more complex decisions, cont.
  - Logical OR operator  | | (double vertical bar)
    - `if(switch1 == 0 || switch2 == 1)`
      turn on the motor
    - The condition evaluates to TRUE if one or the other or both expressions on either side of && evaluate to TRUE
      - Note operator precedence
    - Otherwise condition evaluates to FALSE
      - Beware of ‘short circuit evaluation’
        - Make the condition most likely to be TRUE the left-most condition
Nesting selection structures

Selection structures can be stacked and nested to handle more sophisticated decision/action functionality

- Ex. Figuring grades
  - Pseudocode →

Notes:

- “an else is always associated with the nearest previous if” (Darnell & Margolis, 1996)
- Use braces ({ }) to clarify the association of the else for other situations where the decision structure is more complicated

Grade Determination for Overall Percentage (OP)

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’

Adapted from Deitel & Deitel, C How to Program, 3rd ed., p. 64
Nesting If/else – C Code – Two Ways

if(grade > 90)
    printf(“A\n”);
else
    if(grade >= 80)
        printf(“B\n”);
    else
        if(grade >= 70)
            printf(“C\n”);
        else
            if(grade >= 60)
                printf(“D\n”);
            else
                printf(“F\n”);

if(grade > 90)
    printf(“A\n”);
else if(grade >= 80)
    printf(“B\n”);
else if(grade >= 70)
    printf(“C\n”);
else if(grade >= 60)
    printf(“D\n”);
else
    printf(“F\n”);

Adapted from Deitel & Deitel, C How to Program, 3rd ed., p. 64
SWITCH

- Good when faced with testing multiple alternatives that depend on a single variable
  - The test is done once
    - Must be an integral expression
      - int or char
      - NOT float, double
    - case items must be constant integral expressions
  - No variables
  - The structure is very organized and readable

```c
#include <stdio.h>

int main()
{
    int user_sel;  /* variable for user menu choice */
    char line[100]; /* container for line of user input */

    /* Show the user the menu of choices, and ask for choice */
    printf("1. Play game\n"
    "2. Load game\n"
    "3. Play multiplayer\n"
    "4. Exit\n" "Enter your selection: ");

    /* Get user input - more robust method than scanf */
    fgets(line, sizeof(line), stdin);
    sscanf(line, &user_sel);

    switch (user_sel)
    {
    case 1:  /* Note the colon, not a semicolon */
        printf("\nOK, let's play the game...");
        /* call game function here */
        break;

    case 2:  /* Note the colon, not a semicolon */
        printf("\nOK, loading the game...");
        /* call load game function here */
        break;

    case 3:  /* Note the colon, not a semicolon */
        printf("\nOK, multiplayer game...");
        /* call multiplayer game function here */
        break;

    case 4:  /* Note the colon, not a semicolon */
        printf("\nOK, thank you for playing!");
        break;

    /* Always should have a default case for unexpected cases */
    default:  /* Note the colon, not a semicolon */
        printf("Error, bad input, quitting\n");
        break;
    }
}```
SWITCH - Flowchart

Fig. 4.8 | switch multiple-selection statement with breaks.

Adapted from Deitel & Deitel, C How to Program, 6th ed., p. 111
Practice - 1

- Pair up with someone next to you that you do not know
- Develop an algorithm for:
  - Finding and printing out the largest of two numbers
- (3 min) One person work on the pseudocode, the other on a flowchart
- (1 min) Compare pseudocode and flowchart
- (3 min) Write out the algorithm in C
Practice - 2

- Develop an algorithm for the ignition control in a car:

Requirements:

- The starter will only start when:
  - Key must be in the ignition slot
  - Transmission selector must be in ‘Park’
  - Key must be turned to ‘Start’ position

- The starter is energized with the statement `starter_on();`
Review
References


Nesting selection structures

- Selection structures can be **stacked** and **nested** to handle more sophisticated decision/action functionality

```c
/* File: ifc.c */
#include <stdio.h>
int main ()
{
    int i;
    i = 10;
    if(i==2 || i == 4)
    {
        printf("i = 2 or 4\n");
    }
    else if(i == 10)
    {
        printf("i = 10\n");
    }
    else
    {
        printf("i = %d\n", i);
    }
    return 0;
}
```

Adapted from H. Cheng chap05.ppt, slide 12
### Operators

<table>
<thead>
<tr>
<th>Operations</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>::</td>
<td></td>
</tr>
<tr>
<td>()  []</td>
<td>left to right</td>
</tr>
<tr>
<td><code>Function_name()</code></td>
<td>right to left</td>
</tr>
<tr>
<td>.  -&gt;</td>
<td>left to right</td>
</tr>
<tr>
<td><code>!</code> <code>+</code> <code>--</code> <code>+</code> <code>*</code></td>
<td>right to left</td>
</tr>
<tr>
<td>&amp;(type) sizeof</td>
<td>left to right</td>
</tr>
<tr>
<td><code>*</code> <code>/</code> <code>%</code> <code>*.</code> <code>./</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>+</code> <code>-</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>&lt;&lt;</code> <code>&gt;&gt;</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>&lt;</code> <code>&lt;=</code> <code>&gt;</code> <code>&gt;=</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>==</code> <code>!</code> <code>!=</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>^</code></td>
<td>left to right</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td><code>&amp;&amp;</code></td>
<td>left to right</td>
</tr>
<tr>
<td><code>^^</code></td>
<td>left to right</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
<tr>
<td><code>?:</code></td>
<td>right to left</td>
</tr>
<tr>
<td>`= += -= *= /= %=</td>
<td>right to left</td>
</tr>
<tr>
<td>`</td>
<td>= &lt;&lt;= &gt;&gt;=`</td>
</tr>
<tr>
<td><code>,</code></td>
<td>left to right</td>
</tr>
</tbody>
</table>

Adapted from H. Cheng chap04.ppt, slide 5