Week 3: Control Structures - Selection

BJ Furman
12SEP2009
The Plan for Today

- 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

- 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
  - **switch** (also called ‘multiple-selection’)
    - Allows selection among many actions depending on the value of a variable or expression
If statement

- Syntax

```java
if(expression) /* if expression not equal to zero */
    statement1; /* execute this statement */
/* if expression is equal to zero, statement1 is skipped */
```

- Can execute multiple statements in a ‘block’ by enclosing statements in curly brackets

```java
if(expression)
{
    statement1;
    statement2;
}
```
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”);
      }
  ```
Graphical View - Flowchart

Speed > 65

TRUE

Print “You’re speeding!”

FALSE
If/Else statement

Syntax

```c
if(expression) /* if expression not equal to zero */
    statement1; /* execute this statement */
else /* else execute the following statement */
    statement2;
```

Notes:

- if `expression` is equal to `one`, `statement1` is executed, then the program continues with the statement after `statement2`
- if `expression` is equal to `zero`, `statement1` is skipped and `statement2` is executed
Graphical View - Flowchart

Print “Within limit”  
Speed > 65  
Print “Over speed limit!”
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

5 < 7  result is _____
5 > 7 result is ______
7 <= 7 result is ____
8 >= 7 result is ____
5 == 5 result is ____
5 == 7 result is ____
5.0 == 5 result is ___
6 != 5 result is ____

Adapted from H. Cheng chap04.ppt, slide 5
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

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

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’
Nesting If/else – C Code – Two Ways

```c
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
  - 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, "%d", &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;
    }
}
```
Review
References


Nesting selection structures

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

/* 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;
}