UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #12 –
Decision Making (II): multi-way selection

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Administrative Issues

- Lab#5
  - Review Exam#1 problems
  - due **5pm, Thursday, Feb. 23**

- Today's topics
  - Lecture#12 (Multi-way selection)
  - Lecture#13 (Loops)

Review of Lectures #10 (1)

- Logical data: true (1) or false (0)
  - C supports this through int type: zero (false), non-zero (true)

- 3 logical operators:
  - ! NOT, && (logical AND), || (logical OR)

- 6 relational operators

<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
==	equal
!=	not equal

Review of Lectures #10 (2)

- *Two-way selection: if…else* statement

```
if (expression)
    {
     Action 1
    }
   else
    {
   Action 2
   }
```

- Nested *if…else* statement: An *if…else* is included within another *if…else*
- Dangling *else* problem: when there is no matching *else* for every *if,* Solution: *Always pair an "else" to the most recent unpaired "if" in the current block!*
- Ternary conditional operator

  expression1 ? expression2 : expression3

  – This means that if expression1 is true, then the overall expression evaluates to expression 2, else it evaluates to expression3.

Topics

- **Multi-way selection**
  - *switch* statement
  - *if-else-if* control structure

Multi-Way Selection

- C also provides multi-way selection concept
  - Choose among several alternatives
- Two methods
  - *switch* statement
  - *if-else-if* control structure (a convenient style to the nested *if…else*)

switch statements

```
switch (expression)
{
  case constant-1:
                    statements1
                    break;
  case constant-2:
                    statements2
                    break;
   case constant-3:
                    statements3
                    break;
    ....
  default:
                    statements
                    break;
}
```

- The *break* statement causes the program to jump out of the *switch* – to go to the closing brace and continue the code following the *switch*
- The *default* statement is executed whenever none of the other case values matches the value in the *switch* expression
- However, the *default* label is not required, but it is a good idea to have it.
- In case of no *default* and the value of the control expression does not match with any label, the compiler will simply continue with the statement after the closing brace in the *switch*

An Example

```
#include <stdio.h>
void main(void)
{
      char c;
      printf("Enter a stock rating:\n");
      scanf_s("%c", &c);

      switch (c)
      {
            case 'A':
                        printf("This is an excellent stock\n");
                              break;
            case 'B':
                        printf("This is an OK stock\n");
                              break;
            case 'C':
                       printf("This is not a good stock\n");
                             break;
            default:
                       printf(" The stock rating you entered does not match our records\n");
     }   /*end of switch*/
}
```

Exercise (1)

- Please enter, compile and run the program on slide 8
- Try the following inputs:
  - A
  - a
  - B
  - C
  - E
- Remove all the *break;* and see what happens.
- Remove the *default* case and see what happens.

switch statements (Rules)

```
switch (expression)
{
  case constant-1:
                    statements
                    break;
  case constant-2:
                    statements
                    break;
   case constant-3:
                    statements
                    break;
       ……
    default:
                    statements
                    break;
}
```

- The control expression that *switch* tests must be an integral type, i.e., it can not be a float or a double for example.
- The expression followed by each case label must be a constant expression.
- Two *case* labels can not have the same value.
- However, two cases can have the same statements.
- The *switch* can include at most one *default* label. And it can be coded anywhere, but is traditionally coded last.

Note!

The *switch* statements can be used only when the selection condition can be reduced to an integral expression!

Agenda

- Multi-way selection
  - *switch* statement
  - ***if-else-if* control structure**

```
if (expression-1)
    {
      statement-block-1
    }
else if (expression-2)
    {
      statement-block-2
    }
    ……

else if (expression-n)
    {
      statement-block-n
    }
else
    {
      statement-block-n+1
    }
```

if-else-if
control
structure

Exercise (2)

- Write a *if-else-if* statement that can convert a numeric score to a letter grade
  - 90 or more → A
  - 80 - 90 → B
  - 70 - 80 → C
  - 60 - 70 → D
  - Below 60 → F

Exercise (3)

- Recode the score-to-grade conversion problem on Slide 14 using the *switch* statement

Review Questions (True/False)

- ____Multiway selection can be accomplished using either the *switch* statement or an *if-else-if* format
- ____The *case* constants within a *switch* statement must be arranged in sequence, such as 10, 11, 12, and so on
- ____A *switch* statement can be replaced by an *if-else-if* control structure
- ____A *switch* statement must contain a *default* case section
- ____The *switch* statement is used to make a decision between many alternatives when different conditions can be expressed as integral values

Summary of Lecture #12

- Multi-way selection using
  - *switch* statement: can be used only when the selection condition can be reduced to an integral expression!
  - *if-else-if* control structure: no the above limitation

Things To Do

- Review Exam#1