Multiway Decisions

Lecture 11
Sections 4.2 - 4.6

Robb T. Koether

Hampden-Sydney College

Wed, Sep 18, 2019

Outline

Multi-Way Decisions

- `if` statements may be nested to allow the programmer to make multi-way decisions.
- The false action of each `if` statement is itself an `if` statement.
- Thus, if the first boolean expression is false, the program tests the second boolean expression, and so on.

Multi-Way Decisions

The `if` Statement

```
if (boolean-expression-1)
{
    true-action-1
}
else
    if (boolean-expression-2)
    {
        true-action-2
    }
    else
        if (boolean-expression-3)
        {
            true-action-3
        }
            :
            else
            {
                false-action
            }
```

Multi-Way Decisions

The `if` Statement

```
if (boolean-expression-1)
{
    true-action-1
}
else if (boolean-expression-2)
{
    true-action-2
}
else if (boolean-expression-3)
{
    true-action-3
}
        ⋮
else
{
    false-action
}
```

Multi-Way Decisions

- The boolean expressions are tested sequentially until one of them is found to be true, or until they all are found to be false.
- When one is found to be true, its *true-action* is performed and the remainder of the structure is skipped.

Multi-Way Decisions

- If all of the boolean expressions are found to be false, then the false-action (the “else” part) is performed.
- The final “else” part is optional.
- If the final else part is omitted and all the boolean expressions are found to be false, then no action is performed.

Outline

The `switch` Statement

- The `switch` statement allows the programmer to make a decision based on the value of an integer-valued expression.
- The value of the expression is computed.
- Then execution branches to one of several cases, depending on that value.

The **switch** Statement

The **switch** Statement

```
switch (integer-valued-expression)
{
    case value-1:
        action-1
    case value-2:
        action-2
            ⋮
    case value-n:
        action-n
    default:
        default-action
}
```

The `switch` Statement

- The *integer-valued-expression* must be of one of the types **`bool`**, **`char`**, **`short`**, **`int`**, **`long`**, or **`long long`**.
- *action-i* is performed if the value of the expression matches *value-i*.
- The default part is optional.
- If the value fails to match any case value, then the *default-action* is performed, if there is one.

The `switch` Statement

- Normally, each action ends with a **`break`** statement.
- The **`break`** statement causes execution to exit the **`switch`** structure.
- In the absence of a **`break`** statement, execution will drop through to the next action part.

The switch Statement

The switch Statement

```
switch (integer-valued-expression)
{
    case value-1:
        action-1
        break;
    case value-2:
        action-2
        break;
            ⋮
    case value-n:
        action-n
        break;
    default:
        default-action
}
```

The switch Statement

The switch Statement

```
switch (letter)
{
case 'A': case 'E': case 'I': case 'O': case 'U':
case 'a': case 'e': case 'i': case 'o': case 'u':
    is_vowel = true;
    break;
default:
    is_vowel = false;
    break;
}
```

The `switch` Statement

- The individual case actions do not require braces.
- However, if any variable is declared within the case, then the braces are required.

Example of a **`switch`** Statement

- Example
  - `ComputeGradeEquivalent.cpp`
  - `AssignGradePoint.cpp`

Outline

Assignment

Assignment

- Read Sections 4.2 - 4.6.