# An Introduction to Programming with C++

*Fifth Edition*

*Chapter 6*

*More on the Selection Structure*

# Objectives

- Include a nested selection structure in pseudocode and in a flowchart
- Code a nested selection structure in C++
- Recognize common logic errors in selection structures

# Objectives (continued)

- Include the `switch` form of the selection structure in pseudocode and in a flowchart
- Code the `switch` form of the selection structure in C++
- Display a message along with the contents of one or more variables in a .NET program

# Concept Lesson

- Nested Selection Structures
- Logic Errors in Selection Structures
- Using the `if/else` Form to Create Multiple-Path Selection Structures
- Using the `switch` Form to Create Multiple-Path Selection Structures

# Nested Selection Structures

- A **nested selection structure** is contained within the outer selection structure
  - In the true path or in the false path
  - Use if more than one decision must be made before appropriate action can be taken

| Message | Criteria |
| --- | --- |
| You are too young to vote. | person is younger than 18 years old |
| You can vote. | person is at least 18 years old and is registered to vote |
| You must register before you can vote. | person is at least 18 years old but is not registered to vote |

# Nested Selection Structures (continued)

**Pseudocode – Voter eligibility program**

```
1. enter the age
2. if (the age is greater than or equal to 18)
        enter the registration status
        if (the registration status is Y)
                display "You can vote." message
        else
                display "You must register before you can vote." message
        end if
   else
        display "You are too young to vote." message
   end if
```

nested selection structure

**C++ code – Voter eligibility program**

```
int main()
{
    int age = 0;

    cout << "Enter your age: ";
    cin >> age;

    if (age >= 18)
    {
        char registered = ' ';
        cout << "Are you registered to vote? (Y/N): ";
        cin >> registered;

        if (toupper(registered) == 'Y')
            cout << "You can vote." << endl;
        else
            cout << "You must register before you can vote." << endl;
        //end if
    }
    else
        cout << "You are too young to vote." << endl;
    //end if

    return 0;
}   //end of main function
```

nested selection structure

**Figure 6-1:** Pseudocode and C++ code showing the nested selection structure in the true path

# Nested Selection Structures (continued)

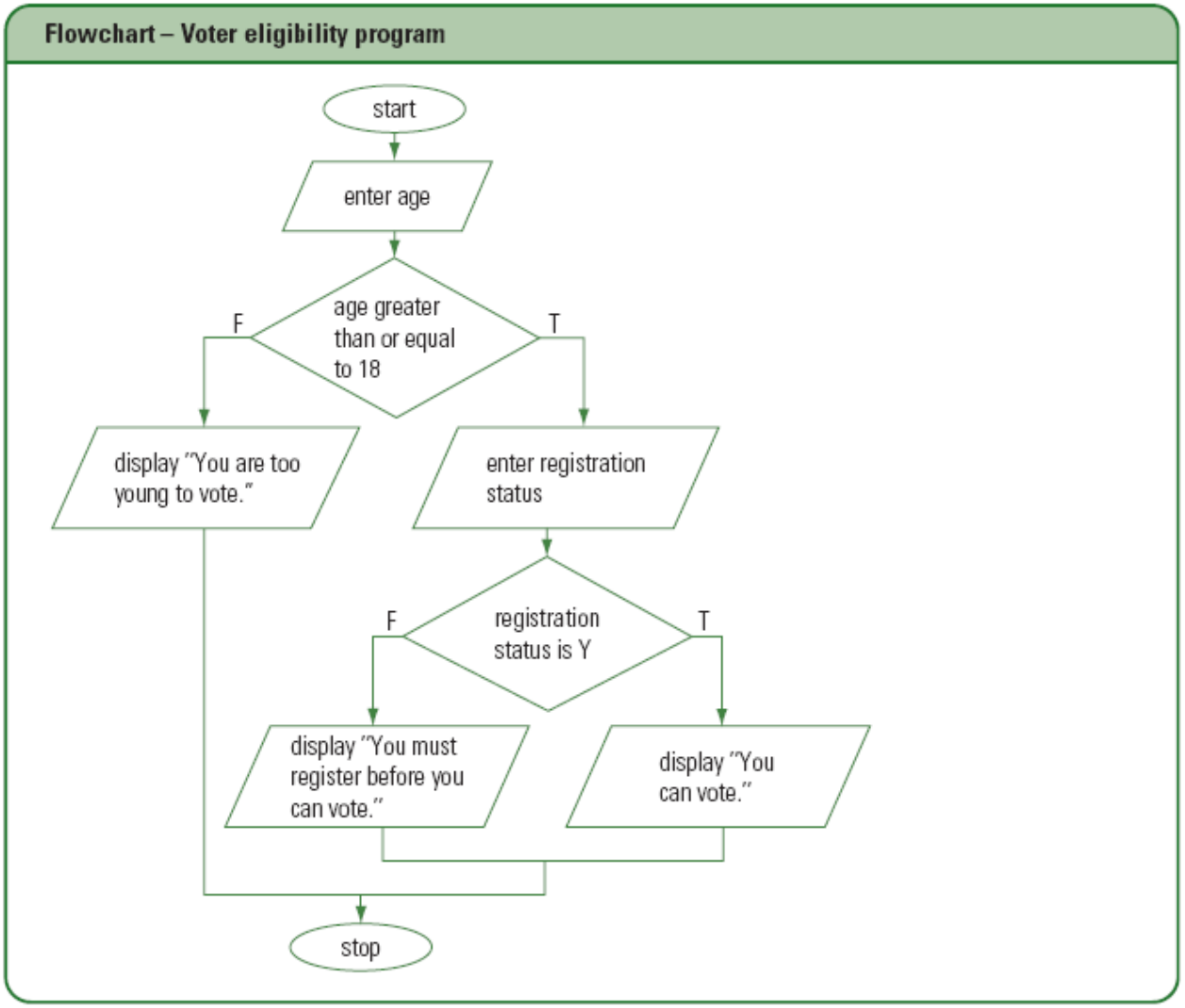



**Figure 6-2:** Flowchart showing the nested selection structure in the true path

# Nested Selection Structures (continued)

**Pseudocode – Voter eligibility program**

```
1. enter the age
2. if (the age is less than 18)
        display "You are too young to vote." message
   else
        enter the registration status
        if (the registration status is Y)
              display "You can vote." message
        else
              display "You must register before you can vote." message
        end if
   end if
```

nested selection structure

**C++ code – Voter eligibility program**

```
int main()
{
    int age = 0;

    cout << "Enter your age: ";
    cin >> age;

    if (age < 18)
        cout << "You are too young to vote." << endl;
    else
    {
        char registered = ' ';
        cout << "Are you registered to vote? (Y/N): ";
        cin >> registered;

        if (toupper(registered) == 'Y')
            cout << "You can vote." << endl;
        else
            cout << "You must register before you can vote." << endl;
        //end if
    } //end if

    return 0;
}   //end of main function
```

nested selection structure

**Figure 6-3:** Pseudocode and C++ code showing the nested selection structure in the false path

# Nested Selection Structures (continued)

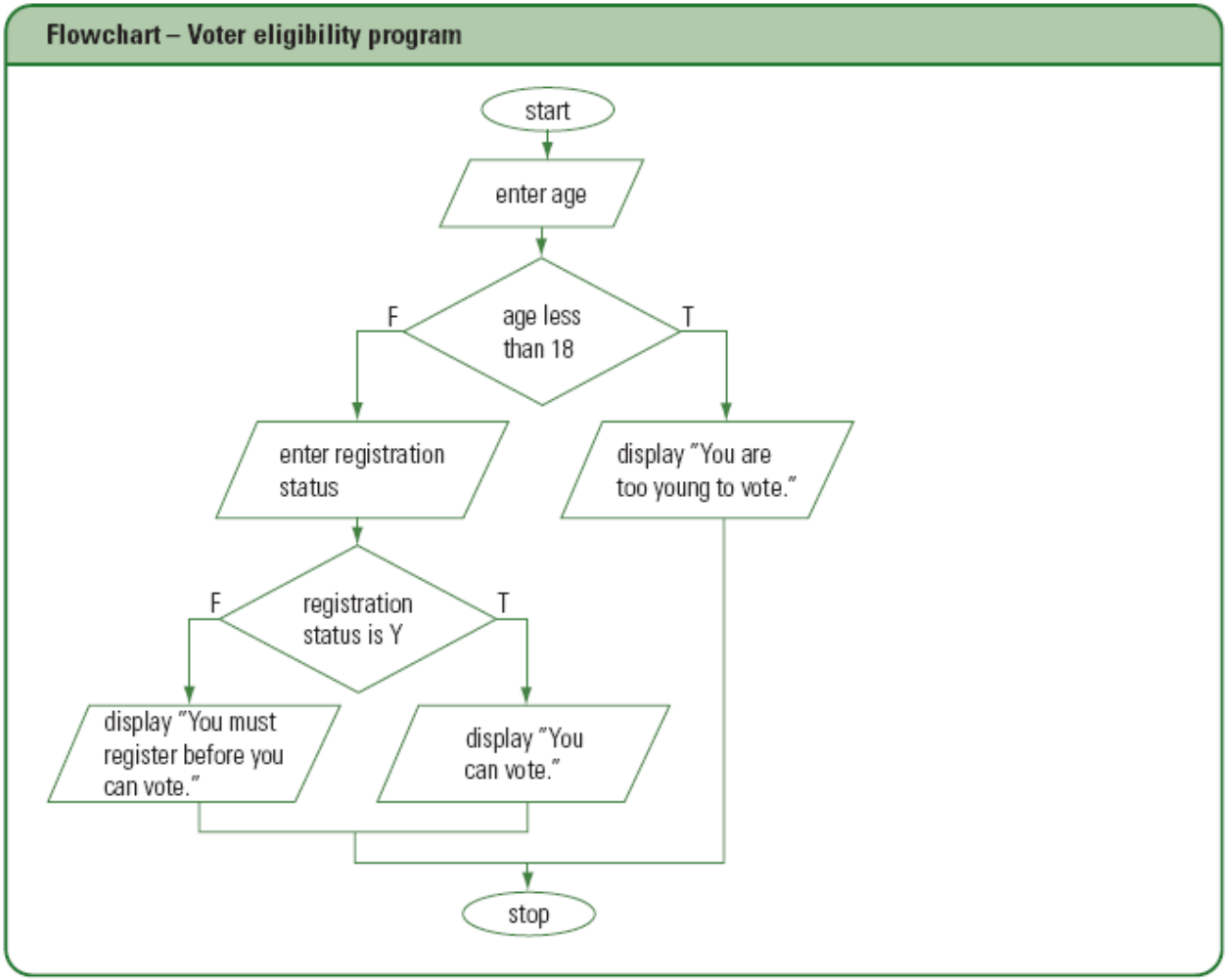



**Figure 6-4:** Flowchart showing the nested selection structure in the false path

# Another Example of a Nested Selection Structure

- Lab 5.2: program for Marshall Cabello
  - Allows Marshall to enter number of calories and grams of fat contained in a specific food
  - Calculates and displays:
    - Food's fat calories
    - Its fat percentage
- Modify program so that it displays "This food is high in fat." if fat percentage is > 30%
  - Otherwise, display "This food is not high in fat."

# Another Example of a Nested Selection Structure (continued)

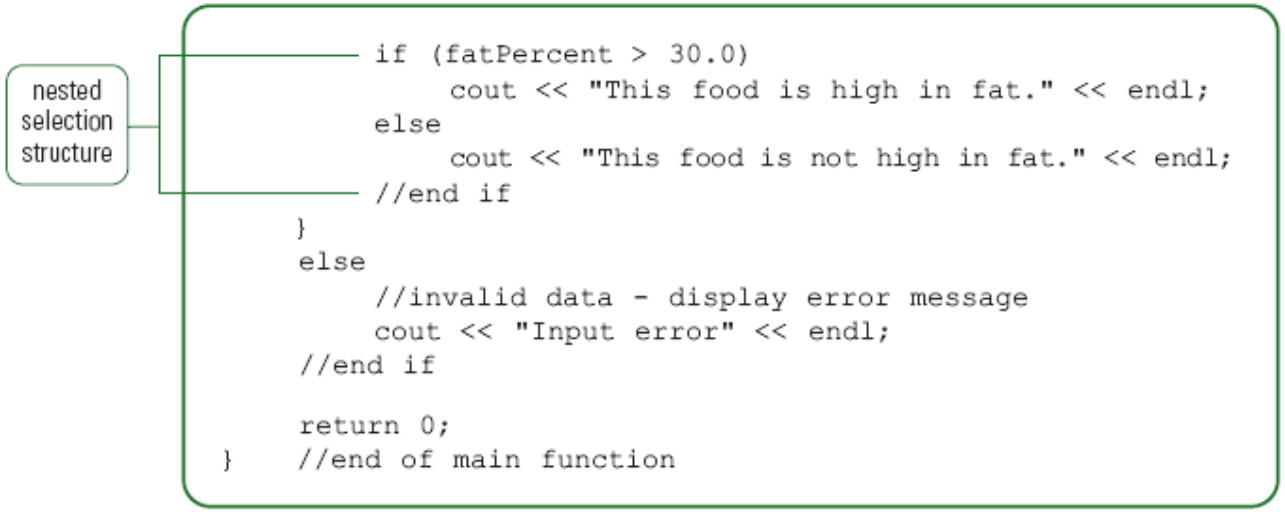
```
int main()
{
     int totalCal      = 0;
     int fatGrams      = 0;
     int fatCal        = 0;
     double fatPercent = 0.0;

     //enter input data
     cout << "Enter the total calories: ";
     cin >> totalCal;
     cout << "Enter the grams of fat: ";
     cin >> fatGrams;

     //validate input data
     if (totalCal >= 0 && fatGrams >= 0)
     {
          //valid data - calculate fat calories
          //and fat percentage
          fatCal = fatGrams * 9;
          fatPercent =
               static_cast<double>(fatCal)
               / static_cast<double>(totalCal) * 100.0;
          cout << fixed << setprecision(0);
          cout << "Fat calories: " << fatCal << endl;
          cout << "Fat percentage: " << fatPercent << endl;
```

**Figure 6-5:** Modified `main()` function containing a nested selection structure

# Another Example of a Nested Selection Structure (continued)




```
            if (fatPercent > 30.0)
                cout << "This food is high in fat." << endl;
            else
                cout << "This food is not high in fat." << endl;
            //end if
        }
        else
            //invalid data - display error message
            cout << "Input error" << endl;
        //end if

        return 0;
}       //end of main function
```

**Figure 6-5:** Modified `main()` function containing a nested selection structure *(Continued)*

# Logic Errors in Selection Structures

- Logic errors in selection structures are usually the result of:
  - Using a logical operator rather than a nested selection structure
  - Reversing the primary and secondary decisions
  - Using an unnecessary nested selection structure
- XYZ Company's bonus program will be used to demonstrate these errors

# Logic Errors in Selection Structures (continued)

**Pseudocode – Bonus program**

```
1. enter the code and sales amount
2. calculate the bonus amount by multiplying the sales amount by .08
3. if (the code is X)
        if (the sales are greater than or equal to 10000)
            add 150 to the bonus amount
        else
            add 125 to the bonus amount
        end if
   end if
4. display the bonus amount
```

**Figure 6-6:** A correct algorithm for the bonus program

# Logic Errors in Selection Structures (continued)

| code | sales amount | bonus amount |
|---|---|---|
| X | 15000 | 1200 |

**Figure 6-7:** Current status of the desk-check table

| code | sales amount | bonus amount |
|---|---|---|
| X | 15000 | ~~1200~~<br>1350 |

**Figure 6-8:** Desk-check table after completing the first desk-check

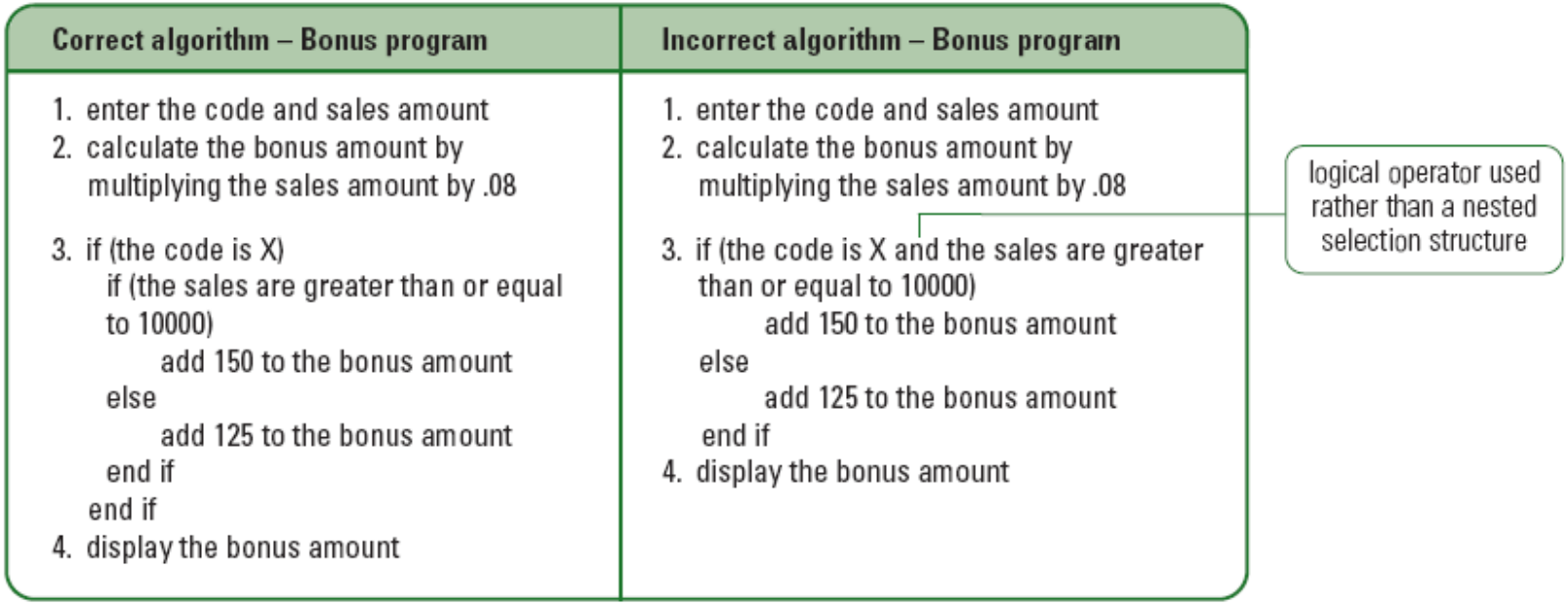
| code | sales amount | bonus amount |
|---|---|---|
| ~~X~~<br>X | ~~15000~~<br>9000 | ~~1200~~<br>~~1350~~<br>~~720~~<br>845 |

**Figure 6-9:** Desk-check table after completing the second desk-check

| code | sales amount | bonus amount |
|---|---|---|
| ~~X~~<br>~~X~~<br>A | ~~15000~~<br>~~9000~~<br>13000 | ~~1200~~<br>~~1350~~<br>~~720~~<br>~~845~~<br>1040 |

**Figure 6-10:** Desk-check table after completing the third desk-check

# Using a Logical Operator Rather than a Nested Selection Structure



**Correct algorithm – Bonus program**

1. enter the code and sales amount
2. calculate the bonus amount by multiplying the sales amount by .08
3. if (the code is X)
   if (the sales are greater than or equal to 10000)
   add 150 to the bonus amount
   else
   add 125 to the bonus amount
   end if
   end if
4. display the bonus amount

**Incorrect algorithm – Bonus program**

1. enter the code and sales amount
2. calculate the bonus amount by multiplying the sales amount by .08
3. if (the code is X and the sales are greater than or equal to 10000)
   add 150 to the bonus amount
   else
   add 125 to the bonus amount
   end if
4. display the bonus amount



**Figure 6-11:** Correct algorithm and an incorrect algorithm containing the first logic error

# Using a Logical Operator Rather than a Nested Selection Structure (continued)

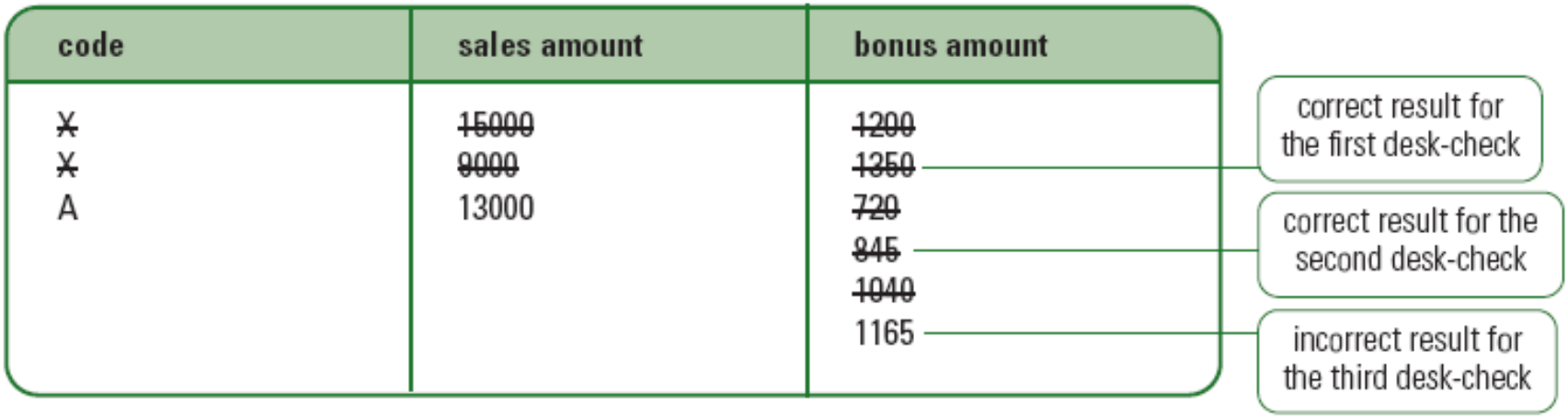



**Figure 6-12:** Desk-check table for the incorrect algorithm in Figure 6-11

# Reversing the Primary and Secondary Decisions

| Correct algorithm – Bonus program | Incorrect algorithm – Bonus program |
|---|---|
| 1. enter the code and sales amount<br>2. calculate the bonus amount by multiplying the sales amount by .08<br>3. if (the code is X)<br>if (the sales are greater than or equal to 10000)<br>add 150 to the bonus amount<br>else<br>add 125 to the bonus amount<br>end if<br>end if<br>4. display the bonus amount | 1. enter the code and sales amount<br>2. calculate the bonus amount by multiplying the sales amount by .08<br>3. if (the sales are greater than or equal to 10000)<br>if (the code is X)<br>add 150 to the bonus amount<br>else<br>add 125 to the bonus amount<br>end if<br>end if<br>4. display the bonus amount |

primary and secondary decisions reversed

**Figure 6-13:** Correct algorithm and an incorrect algorithm containing the second logic error

# Reversing the Primary and Secondary Decisions (continued)

| code | sales amount | bonus amount |
|---|---|---|
| ~~X~~ | ~~15000~~ | ~~1200~~ |
| ~~X~~ | ~~9000~~ | ~~1350~~ |
| A | 13000 | ~~720~~ |
| | | ~~1040~~ |
| | | 1165 |

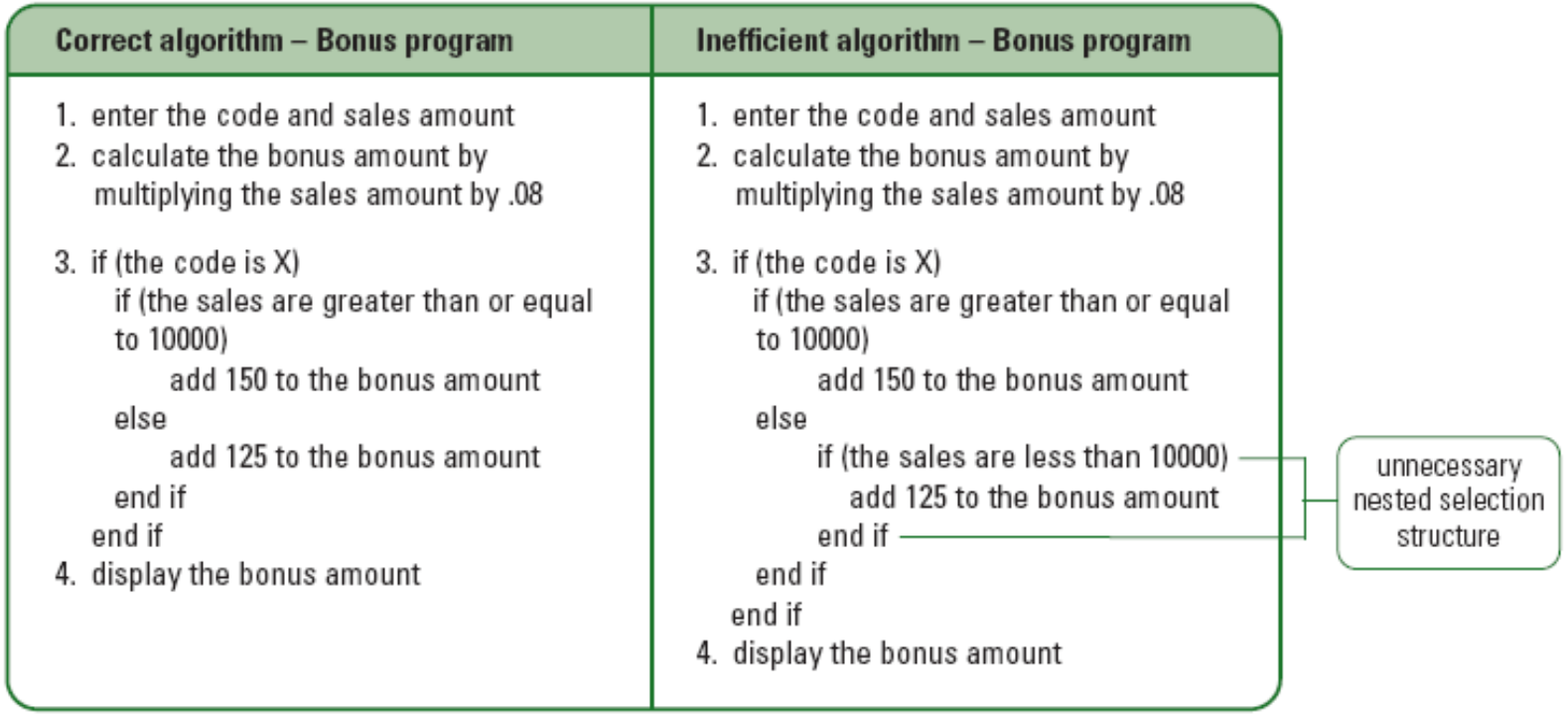
correct result for the first desk-check

incorrect result for the second desk-check

incorrect result for the third desk-check

**Figure 6-14:** Desk-check table for the incorrect algorithm in Figure 6-13

# Using an Unnecessary Nested Selection Structure



**Correct algorithm – Bonus program**

1. enter the code and sales amount
2. calculate the bonus amount by multiplying the sales amount by .08

3. if (the code is X)
   if (the sales are greater than or equal to 10000)
   add 150 to the bonus amount
   else
   add 125 to the bonus amount
   end if
   end if
4. display the bonus amount

**Inefficient algorithm – Bonus program**

1. enter the code and sales amount
2. calculate the bonus amount by multiplying the sales amount by .08

3. if (the code is X)
   if (the sales are greater than or equal to 10000)
   add 150 to the bonus amount
   else
   if (the sales are less than 10000)
   add 125 to the bonus amount
   end if
   end if
   end if
4. display the bonus amount



**Figure 6-15:** Correct algorithm and an inefficient algorithm containing the third logic error

# Using an Unnecessary Nested Selection Structure (continued)

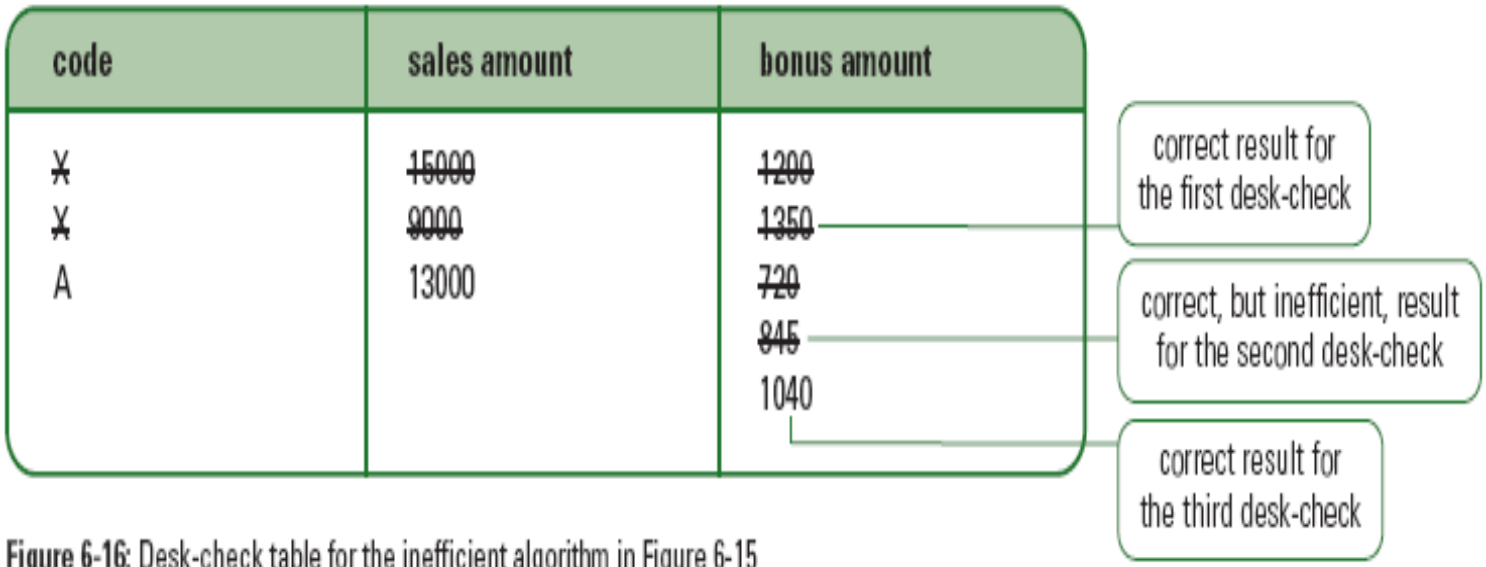



**Figure 6-16:** Desk-check table for the inefficient algorithm in Figure 6-15

# Using the `if/else` Form to Create Multiple-Path Selection Structures

- You can create a selection structure that can choose from several alternatives
  - **Multiple-path selection structure**
    - Also called **extended selection structure**
  - E.g., display message based on letter grade

| Letter grade | Message |
|---|---|
| A | Excellent |
| B | Above Average |
| C | Average |
| D | Below Average |
| F | Below Average |

# Using the `if/else` Form to Create Multiple-Path Selection Structures (continued)

**Version 1 – Grade problem using the `if/else` form of the selection structure**

```
int main()
{
     char grade = ' ';

     //enter input data
     cout << "Letter grade: ";
     cin >> grade;
     grade = toupper(grade);
     if (grade == 'A')
         cout << "Excellent" << endl;
     else
         if (grade == 'B')
              cout << "Above Average" << endl;
         else
              if (grade == 'C')
                   cout << "Average" << endl;
              else
                   if (grade == 'D' || grade == 'F')
                        cout << "Below Average" << endl;
                   else
                        cout << "Input error" << endl;
                   //end if
              //end if
         //end if
     //end if

     return 0;
}    //end of main function
```

**Figure 6-17:** Two versions of the C++ code for the grade problem

# Using the `if/else` Form to Create Multiple-Path Selection Structures (continued)

**Version 2 – Grade problem using the `if/else` form of the selection structure**

```
int main()
{
     char grade = ' ';

     //enter input data
     cout << "Letter grade: ";
     cin >> grade;
     grade = toupper(grade);

     if (grade == 'A')
          cout << "Excellent" << endl;
     else if (grade == 'B')
          cout << "Above Average" << endl;
     else if (grade == 'C')
          cout << "Average" << endl;
     else if (grade == 'D' || grade == 'F')
          cout << "Below Average" << endl;
     else
          cout << "Input error" << endl;
     //end ifs

     return 0;
}    //end of main function
```

you can use one comment to mark the end of the entire structure

**Figure 6-17:** Two versions of the C++ code for the grade problem *(Continued)*

# Using the `switch` Form to Create Multiple-Path Selection Structures

- If selection structure has many paths from which to choose, it is often simpler to use the `switch` form
  - Also known as the `case` form

**Pseudocode – Grade problem using the `switch` form of the selection structure**

```
1. enter grade
2. grade value:
      A        display "Excellent"
      B        display "Above Average"
      C        display "Average"
      D, F     display "Below Average"
      Other    display "Error"
```

**Figure 6-18:** Pseudocode for the grade problem

# Using the `switch` Form to Create Multiple-Path Selection Structures (continued)

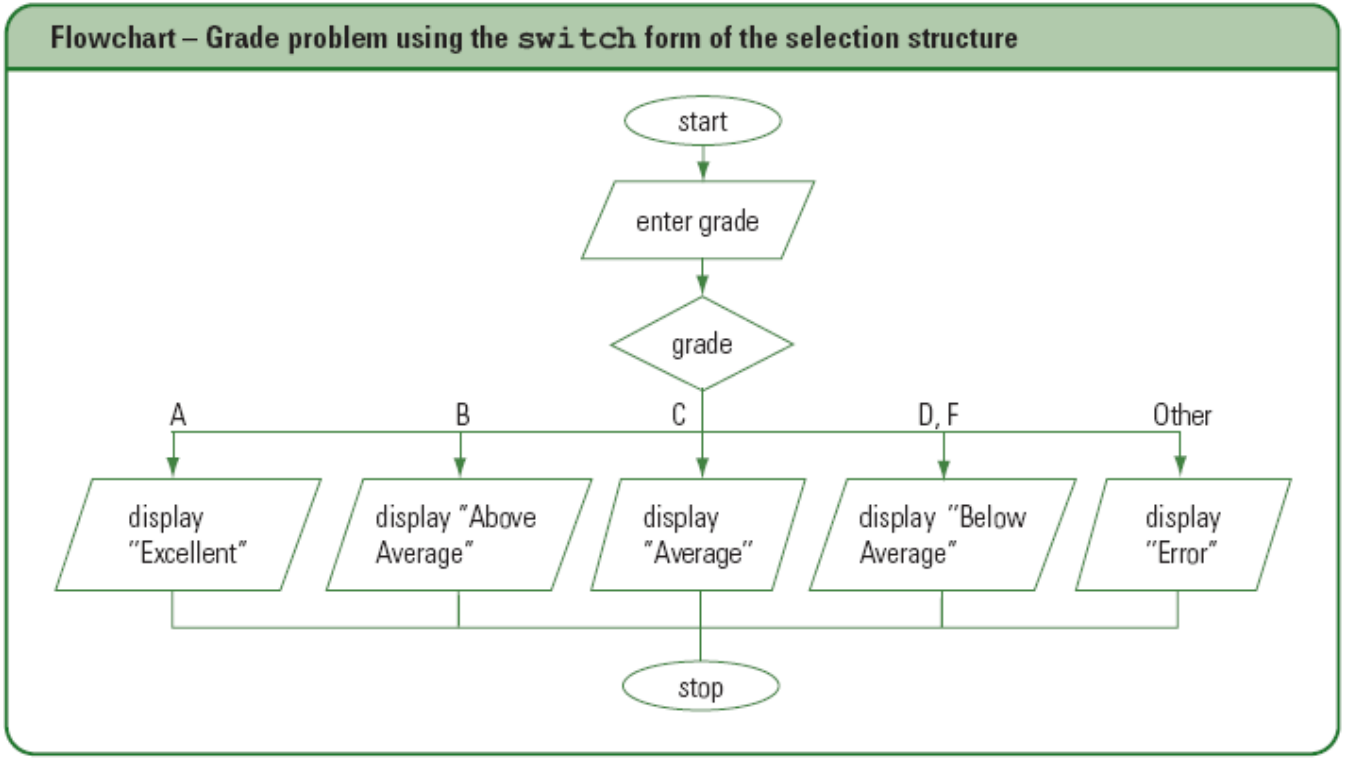



**Figure 6-19:** Flowchart for the grade problem

# Using the `switch` Form to Create Multiple-Path Selection Structures (continued)

**Use the `switch` Statement**

Syntax

```
switch (selectorExpression)
{
case value1:
        one or more statements
        [break;]
[case value2:
        one or more statements
        [break;]]
[case valueN:
        one or more statements
        [break;]]
[default:
        one or more statements processed when the selectorExpression does not match any
        of the values]
}  //end switch
```

**Figure 6-20:** How to use the `switch` statement

# Using the `switch` Form to Create Multiple-Path Selection Structures (continued)

Example

```
int main()
{
     char grade = ' ';

     //enter input data
     cout << "Letter grade: ";
     cin >> grade;
     grade = toupper(grade);

     switch (grade)
     {
     case 'A':
             cout << "Excellent" << endl;
             break;
     case 'B':
             cout << "Above Average" << endl;
             break;
     case 'C':
             cout << "Average" << endl;
             break;
     case 'D':
     case 'F':
             cout << "Below Average" << endl;
             break;
     default:
             cout << "Input error" << endl;
     }     //end switch

     return 0;
}    //end of main function
```

**Figure 6-20:** How to use the `switch` statement *(Continued)*

# Desk-Checking the Grade Program

- Input ‘b’
  - `cout << "Above Average" << endl;` displays the message on the screen
  - `break;` tells computer to skip remaining instructions in `switch`
- Input ‘D’
  - `cout << "Below Average" << endl;`
  - `break;`
- Input ‘X’
  - `default` clause displays `"Input error"`

# Switch Statement Program: Displaying a Price

- A program needs to display a price based on a product ID entered by the user
- The valid product IDs and their prices are shown in the following chart

| Product ID | Price |
|---|---|
| 1 | 50.55 |
| 2 | 12.35 |
| 5 | 11.46 |
| 7 | 11.46 |
| 9 | 12.35 |
| 11 | 11.46 |

# Switch Statement Program: Displaying a Price (continued)

```
int main()
{
     int productId = 0;
     double price  = 0.0;

     //enter input data
     cout << "Product ID (1, 2, 5, 7, 9, or 11): ";
     cin >> productId;

     //assign price
     switch (productId)
     {
     case 1:
          price = 50.55;
          break;
     case 2:
     case 9:
          price = 12.35;
          break;
     case 5:
     case 7:
     case 11:
          price = 11.46;
          break;
     default:
          cout << "Input error" << endl;
     }     //end switch

     //display price
     cout << "Price: " << price << endl;

     return 0;
}    //end of main function
```

**Figure 6-21:** C++ code for the price program *(Continued)*

# Summary

- You can nest a selection structure within either the true path or false path of another selection structure
- Common errors when writing selection structures
  - Using a logical operator when a nested selection structure is needed
  - Reversing the primary and secondary decisions
  - Using an unnecessary nested selection structure
- Code multiple-path selection structures using `if/else` or `switch`
  - The flowchart symbol for the `switch` is a diamond

# Application Lesson: Using the Selection Structure in a C++ Program

- Lab 6.1: Stop and Analyze
- Lab 6.2
  - Jennifer Yardley owns Golf Pro
  - Salespeople receive commission based on their total domestic and foreign sales
- Lab 6.3
  - Modify program from Lab 6.1 by replacing `if` statements with a `switch`
- Lab 6.4: Desk-Check Lab
- Lab 6.5: Debugging Lab