C++ Chapter 3: Decisions

Contents

Time required: 90 minutes

Read: Think C++

- Chapter 4 Conditionals and Recursion

Do: Online Tutorials

- C++ Conditions
- C++ if, if…else and Nested if…else

If Selection Statement

```
#include <iostream>
int main()
{
    int age = 19;
    if (age >= 16)
    {
        std::cout << "You can drive!" << std::endl;
    }
    return 0;
}
```

If Else

```
#include <iostream>
int main()
{
    int age = 19;
    if (age >= 16)
    {
        std::cout << "You can drive!" << std::endl;
    }
    else
    {
        std::cout << "You cannot drive yet!" << std::endl;
    }
    return 0;
}
```

If Else If

```
#include <iostream>
int main()
{
    int age;
    std::cout << "Welcome to the Sunset Bar and Grille." << std::endl;
    std::cout << "Please enter your age: ";
    std::cin >> age;

    if (age >= 21)
    {
        std::cout << "Here, have a beer." << std::endl;
    }
    else if (age >= 16)
    {
        std::cout << "Here have a Coke!" << std::endl;
        std::cout << "At least you can drive!" << std::endl;
    }
    else
    {
        std::cout << "Here have a Coke!" << std::endl;
    }
    std::cout << "Thanks for coming to the Sunset Bar and Grille!" <<
std::endl;
    return 0;
}
```

Tutorial 3.1: Room Area

Create, compile, run and attach the following program.

```cpp
/**
 * @file    room_area.cpp
 * @author
 * @version V1.0.0
 * @date    09/12/2021
 * @brief   Calculate square feet of a room
 */
#include <iostream>

int main()
{
    // TODO: Declarations
    // Declare and initialize constants and variables
    const int BIG_ROOM{1000};
    int room_width{0};
    int room_length{0};
    int square_feet{0};

    // TODO: Input
    // Get width and length of the room from the user
    std::cout << "Enter the width of the room in feet: ";
    std::cin >> room_width;
    std::cout << "Enter the length of the room in feet: ";
    std::cin >> room_length;

    // TODO: Calculate
    // Multiply roomWidth by roomLength to get square feet
    square_feet = room_width * room_length;

    // TODO: Output
    // Display results to user
    if (square_feet > BIG_ROOM)
    {
        std::cout << square_feet
                  << " square feet is a big room." << std::endl;
    }
    else
    {
        std::cout << square_feet
                  << " square feet is a normal size room." << std::endl;
    }
    return 0;
}
```

Example run:

```
Enter the width of the room in feet: 20
Enter the length of the room in feet: 45
900 square feet is a normal size room.
```

```
Enter the width of the room in feet: 40
Enter the length of the room in feet: 40
1600 square feet is a big room.
```

Tutorial 3.2: Ticket to Prize

Create, compile, run and attach the following program.

```cpp
/**
 * @file    TicketToPrize.cpp
 * @author
 * @version V1.0.0
 * @date    09/12/2021
 * @brief   Calculate prize based on number of tickets
 */
#include <iostream>

int main()
{
    // Constants for prize levels
    const int FIVE_TICKETS{5};
    const int TEN_TICKETS{10};
    const int FIFTY_TICKETS{50};
    // Store the number of tickets the user has
    int tickets;

    // Prompt the user and get the number of tickets
    std::cout << "How many tickets do you wish to purchase [5, 10, 50]: ";
    std::cin >> tickets;

    // Determine and display the prize based on the number of tickets
    if (tickets == FIVE_TICKETS)
    {
        std::cout << "Not enough tickets - keep trying!";
    }
    // Second condition
    else if (tickets == TEN_TICKETS)
    {
        std::cout << "You win a slinky! Congratulations!";
    }
    // Third condition
    else if (tickets == FIFTY_TICKETS)
    {
        std::cout << "You win a vacuum cleaner! Congratulations!";
    }
    // The user chose a different number than they were prompted for
    else
    {
        std::cout << "Apparently you can't follow directions, you lose.";
    }
    return 0;
}
```

Example run:

```
How many tickets do you wish to purchase [5, 10, 50]:
Not enough tickets - keep trying!
```

```
How many tickets do you wish to purchase [5, 10, 50]: 10
You win a slinky! Congratulations!
```

```
How many tickets do you wish to purchase [5, 10, 50]: 50
You win a vacuum cleaner! Congratulations!
```

```
How many tickets do you wish to purchase [5, 10, 50]: 2
Apparently you can't follow directions, you lose.
```

Compound Boolean Expressions

Simple Boolean expressions, each involving one relational operator, can be combined into more complex Boolean expressions using the logical operators **&&** (and), **||** (or), and **!** (not). A combination of two or more Boolean expressions using logical operators is called a compound Boolean expression.

&& (and)

```
\\ && (and operator) Both sides must be true
sun_is_shining = true
no_rain = true
if sun_is_shining && no_rain
    go outside
```

|| (or)

```
\\ || (or operator) Only one side must be true
sun_is_shining = true
no_rain = false
if sun_is_shining || no_rain
    go outside
```

Tutorial 3.3: Compound Conditional

The following program demonstrates compound conditionals.

```cpp
/**
 * @file    compound_conditional.cpp
 * @author
 * @version V1.0.0
 * @date    10/17/2022
 * @brief   More than one condition in a decision statement
 */

#include <iostream>
int main()
{
    int age;
    std::cout << "Enter your age: ";
    std::cin >> age;

    // && and - both conditions must be true
    if (age >= 12 && age <= 50)
    {
        std::cout << "Young" << std::endl;
    }
    else
    {
        std::cout << "Not Young" << std::endl;
    }

    // || or - either condition must be true
    if (age < 12 || age > 50)
    {
        std::cout << "Eligible for the offer" << std::endl;
    }
    else
    {
        std::cout << "Not eligible for the offer" << std::endl;
    }
    return 0;
}
```

```
Enter your age: 21
Young
Not eligible for the offer
```

```
Enter your age: 51
Not Young
Eligible for the offer
```

Tutorial 3.4: Input Validation

Using input validation is defensive programming.

Defensive programming is a software development approach that aims to anticipate and mitigate potential issues, errors, and vulnerabilities in code. It involves strategies and practices to enhance the robustness and reliability of software systems. Some key aspects include input validation, error handling, and thorough testing to ensure that software can withstand unexpected situations and minimize the risk of failures.

The following example shows how to prevent a divide by 0 error.

```cpp
#include <iostream>

int main()
{
    int a, b, c;
    std::cout << "Enter a number: " << std::endl;
    std::cin >> a;

    std::cout << "Enter a second number: " << std::endl;
    std::cin >> b;
    if (b == 0)
    {
        std::cout
            << "Invalid denominator, you cannot divide by 0"
            << std::endl;
    }
    else
    {
        c = a / b;
        std::cout << c << std::endl;
    }
    return 0;
}
```

Example runs:

```
Enter a number: 4
Enter a second number: 0
Invalid denominator, you cannot divide by 0
```

```
Enter a number: 5
Enter a second number: 4
1
```

Assignment 1: Space Ship Fuel Level

You are traveling across the galactic void. You are ready for your last jump. It is time to make fuel calculations to determine if you can make it home.

The user is prompted to enter the remaining fuel level in gallons. The program uses an if-else statement to make decisions based on the fuel level entered. Here are the decision conditions:

- If the fuel level is greater than or equal to 1000 gallons, the program outputs that the fuel level is sufficient for the journey.
- If the fuel level is between 500 and 999 gallons (inclusive), the program outputs a warning message to consider refueling.
- If the fuel level is less than 500 gallons, the program outputs a critical low fuel message and advises to abort the mission.

Please use named constants for anything we know before the program starts.

NOTE: Remember the evaluation methods we talked about it in class. You can eliminate and do not have to specifically test for between 500 and 999 gallons.

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show evidence of understanding.
3. Attach the program files.
4. Attach screenshots showing the successful operation of the program.
5. Submit in Blackboard.