Programming Fundamentals with C++

Lecture 8 – Conditional Statements

Dr. Muhammad Sajjad

RA: Asad Ullah

Overview

- **The Conditional Operator**
- **The Switch Statement**
- **The ‘goto’ Statement**

The Conditional Operator (?:)

- C++ provides the conditional operator (?:), which is closely related to the if…else statement.
- The conditional operator is C++'s only ternary operator—it takes three operands.
- The operands, together with the conditional operator, form a conditional expression.
- The conditional operator is consist of "?" (Question-Mark) and a ":" (Colon).
- **Syntax**: Its syntax is:

(Condition) **?** {Expression1} **:** {Expression2}

Where

Condition: Represents the test condition.

Expression1: Value for the entire conditional expression, if the condition is true.

Expression2: Value for the entire conditional expression, if the condition is false.

Note Expression1 & Expression2 may be arithmetic expression or constant values.

Input / Output statements can be used.

The Conditional Operator (?:)

- For example, the output statement

```
cout << (grade >= 60 ? "Passed" : "Failed" );
```

- contains a conditional expression, grade >= 60 ? "Passed" :"Failed“
- Or the values in a conditional expression also can be actions to execute. For example the following conditional expression also prints “Passed” or “Failed”.

```
grade >= 60 ? cout<< "Passed" : cout<< "Failed" ;
```

- The preceding conditional expression is read, “If grade is greater than or equal to 60,then cout<<"Passed"; otherwise, cout<<"Failed".”

- These are very similar to the if-else statement i.e.

```
if(grade >= 60)
            cout<< "Passed" ;
else
             cout<< "Failed" ;
```

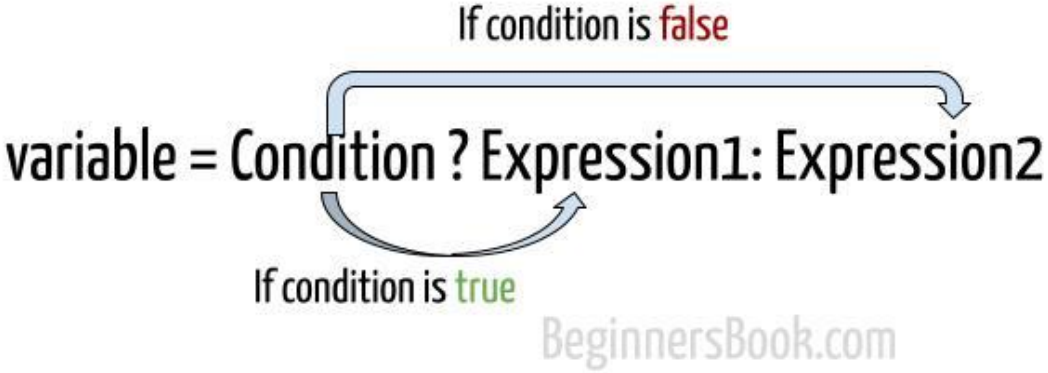
The Conditional Operator (?:)

- **Value Assignment with Conditional Operator**
  - You can also assign a value to a variable with the help of conditional operator.
  - **For example** if a = 10 and b = 5 and result is an integer type variable, then the assignment statement is written as:

```
result = (a>b) ? a : b;
```

  - The (a>b) checks if this is true for a is greater than b.
  - Based on the condition the result will contain max value, which is in this case a i.e. 10.
  - This is equivalent to:

```
if(a>b)
        result =  a;
else
        result = b;
```



The Conditional Operator (?:)

- **Code Example:**

```
#include <iostream>
using namespace std;

int main() {
	int num1, num2;

	cout << "Enter the first number: ";
	cin >> num1;
	cout << "Enter the second number: ";
	cin >> num2;

	int max = (num1 > num2) ? num1 : num2;

	cout << "The maximum value is: " << max << endl;

	return 0;
}
```

1_Conditioanl_Operator_Find_Odd_Even.cpp

2_Conditional_Operator_Find_Max_Value.cpp

The Switch Statement

- **The Switch Statement in C++**
  - The switch statement is a control flow statement that allows a variable to be tested for equality against a list of values.
  - It provides an alternative to using multiple if-else statements and is often simpler and cleaner when working with many conditions.
- **Syntax of switch Statement in C++**

```
switch (expression) {
        case value_1:
                // statements_1;
                break;
        case value_2:
                // statements_2;
                break;
        .....
        .....
        default:
                // default_statements;
                break;
}
```

The Switch Statement

Rules of the switch case statement in C++

- The case value must be either int or char type.
- There can be any number of cases.
- No duplicate case values are allowed.
- Each statement of the case can have a break statement.
- The default Statement is also optional.

```
//switch example with integer
#include<iostream>
using namespace std;

int main()
{
        int day = 0;
        cout << "Please enter your option:" << endl;
        cout << "=========================" << endl;
        cout <<"1: Monday" << endl;
        cout <<"2: Tuesday" << endl;
        cout <<"3: Wednesday" << endl;
        cout <<"4: Thursday" << endl;
        cout <<"5: Friday" << endl;
        cout << "=========================" << endl;
        cin >> day;
        switch(day)
        {
        case 1:
                cout << "Monday" << endl;
                break;
        case 2:
                cout << "Tuesday" << endl;
                break;
        case 3:
                cout << "Wednesday" << endl;
                break;
        case 4:
                cout << "Thursday" << endl;
                break;
        case 5:
                cout << "Friday" << endl;
                break;
        default:
                cout << "Invalid Input" << endl;
        }
        return 0;
}
```

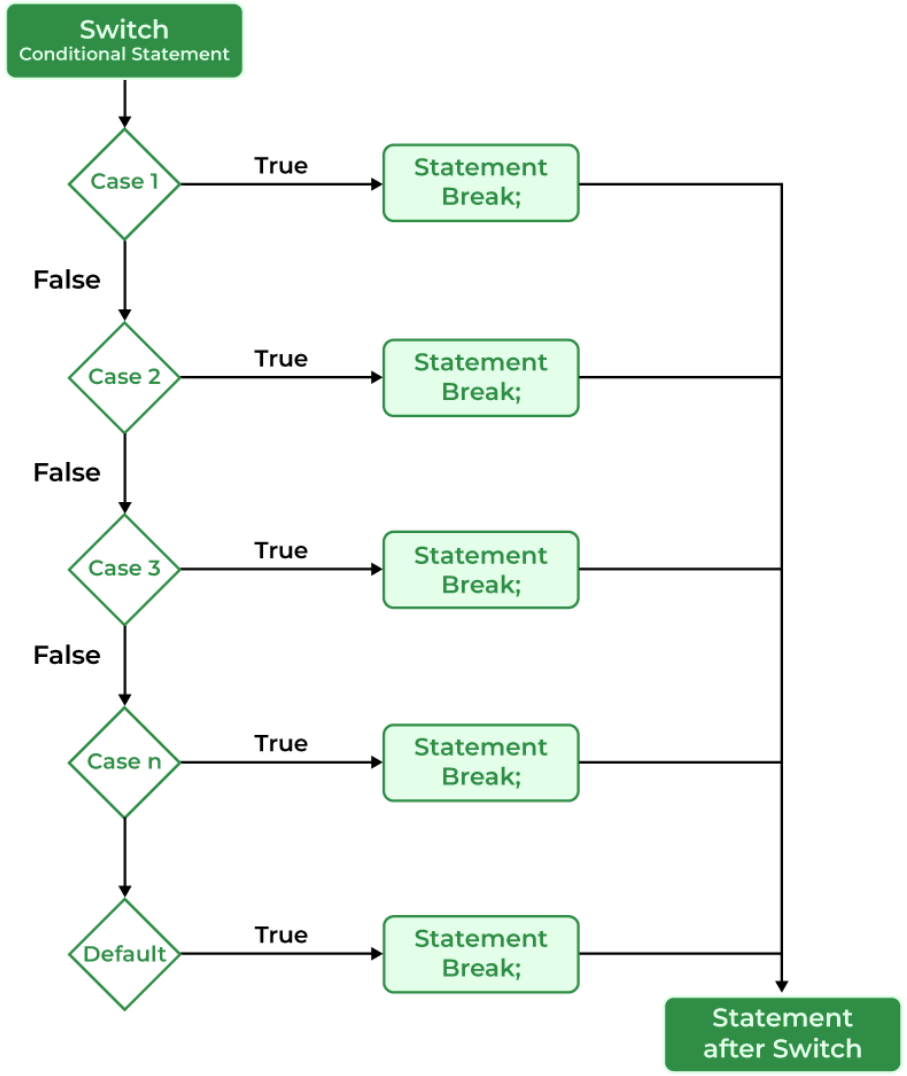
The Switch Statement

- **Working of switch Statement in C++**
  - **Step 1:** The switch expression is evaluated.
  - **Step 2:** The evaluated value is then matched against the present case values.
  - **Step 3A:** If the matching case value is found, that case block is executed.
  - **Step 3B:** If the matching code is not found, then the default case block is executed if present.
  - **Step 4A:** If the break keyword is present in the block, then program control comes out of the switch statement.
  - **Step 4B:** If the break keyword is not present, then all the cases after the matching case are executed.
  - **Step 5:** Statements after the switch statement is executed.

```
#include<iostream>
using namespace std;
void main()
{char ch;
 cout<<"Enter a charater to check it is vowel or not";
 cin>>ch;
     switch(ch)
            {
             case'a': case'A':
             cout<<ch<<" is a Vowel";
             break;
                 case 'e': case'E':
                 cout<<ch<<" is a Vowel";
                 break;
                     case 'i': case'I':
                     cout<<ch<<" is a Vowel";
                     break;
                         case 'o': case'O':
                         cout<<ch<<" is a Vowel";
                         break;
                             case 'u': case'U':
                             cout<<ch<<" is a Vowel";
                             break;
             default:
             cout<<ch<<" is a consonant character not a vowel";
            }
}
```

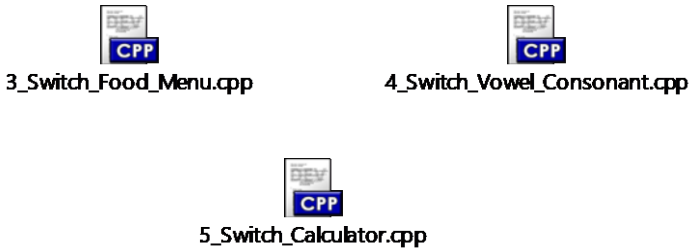
The Switch Statement

- **Flowchart of Switch Statement in C++**



The Switch Statement

Code Example




```
#include <iostream>
using namespace std;

int main()
{
	// switch variable
	char x = 'A';

	// switch statements
	switch (x) {
		case 'A':
			cout << "Choice is A";
			break;
		case 'B':
			cout << "Choice is B";
			break;
		case 'C':
			cout << "Choice is C";
			break;
		default:
			cout << "Choice other than A, B and
C";
		break;
		}
	return 0;
}
```

The ‘goto’ Statement

- In C++, the goto statement transfers the program's control flow from one part to another.
- It allows the users to jump to another part of the program while skipping the current part.
- Although it is generally advised to use structured control flow statements for readability and maintainability, there are many scenarios where the goto statement can be useful.
- **Syntax**

```
int main() {
        // ... code before goto

        goto label; // Jump to the label

        // ... code that will be skipped by the goto

        label:
        // ... code after the label

return 0;
}
```

The ‘goto’ Statement

- **C++ Program to Utilize the goto Statement**

6_Goto_Switch_Calculator.cpp

```
#include <iostream>
using namespace std;
int main() {
        int num1, num2, result;
        char op;
        rerun:
                cout<<“Enter Num 1: ”;          cin>>num1;
                cout<<“Enter Operator: ”;       cin>>op;
                cout<<“Enter Num 2: ”;          cin>>num2;

                switch(op){
                        // statements here
                }
                cout<<“Do you want another Operation\nEnter Y/y for Yes.”;
                char choice;
                cin>>choice;
                if (choice == ‘y’ || choice == ‘Y’)
                        goto rerun;
        return 0;
}
```

The ‘goto’ Statement

- **Utilize goto for error handling**

7_Goto_Error_Handling.cpp

```
#include <iostream>
using namespace std;
int main() {
    double numerator = 10.0;
    double denominator = 0.0;
    double result;

    if (denominator == 0) {
        // Jump to cleanup if denominator is 0
        goto cleanup;
    }

    // Perform the division
    result = numerator / denominator;
    cout << "Result: " << result << endl;
    return 0;

cleanup:
    cerr << "Error: Division by zero is not allowed." << endl;
    cout << "Program has finished executing." << endl;
    return 0;
}
```

Thank You