The switch Statement – Introduction

if and if/else statements can become quite confusing when nested too deeply, and C++ offers an alternative. suppose a software engineering company wants to screen applicants first for experienced programmers and second for C++ programmers specifically. One possible program is the following:

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

int main()
{
        char programmer, cPlusPlus;
        cout << " Before we consider your application , "
                << " answer the following  " << endl;
        cout << " enter Y for  yes    or  enter N  for no  " << endl;
        cout << "\n Are you a computer programmer?   " ;
        cin >> programmer;
        if (programmer == 'Y')
        {
                cout << "\n Do you program in C++?   " ;
                cin >> cPlusPlus;
                if (cPlusPlus == 'Y')
                        cout << "\n You look like a promising "
                           << "candidate for employment   " << endl;
                else
                if (cPlusPlus == 'N')
                        cout << "\n You need to learn C++ before "
                           << "further consideration  " << endl;
                else
                        cout << "\n You must enter Y or N  " << endl;
        }
        else
        if (programmer == 'N')
                cout << "\n You are not currently qualified"
                   << "for employment  " << endl;
        else
                cout << "\n You must enter Y or N  " << endl;

        return 0;
}
```

Unlike if, which evaluates one value, **switch** statements **allow** you to **branch** on any of a number of different values. The switch statement provides an **alternate way** to handle multi-way choices when the conditions are based on equality. The general form of the switch statement is:

The switch Statement – syntax

switch (*expression*) {

case *constant-expression*: *statements*

...

case *constant-expression*: *statements*

default: *statements*

}

- The expression is an instance of an **ordinal** data type (usually an **integer** or **character**) called a case-label.
- The expression evaluated to an int/char value
- Execution starts at the case labeled with the int value
- Execution starts at default if the int value matches none of the case labels

Semantics of The switch Statement :

- The expression must have int (or char) type
- constant-expression must be a constant : -
  - a literal or named constant
- Statements is one or more statements (no braces needed)
- **default** is **optional**

The switch Statement - Example:

```
int quarter;
...
switch (quarter) {
case 1: cout << "First";
 break;
case 2: cout << "Second";
 break;
case 3: cout << "Third";
 break;
case 4: cout << "Fourth";
 break;
default: cout << "Invalid choice";          }
```

The break Statement

- The break statement causes an immediate exit from the switch.
- Without a break statement, execution continues on to the next set of statements.
- Sometimes this is useful: the textbook has some nice examples.

The switch Statement

- Multiple labels for the same set of statements
- if ch is 'a', it falls through to Option A

```
char ch;
...
switch (ch) {
        case 'a':
        case 'A': cout << "You Typed A";
        break;
        case 'b':
        case 'B': cout << "You Typed B";
        break;
        case 'c':
        case 'C': cout << "You Typed C";
        break;
        default: cout << “Invalid choice”;    }
```

NOTE : It is almost always a good idea to have a **`default case`** in **`switch`** statements. If you have no other need for the `default`, use it to test for the supposedly impossible case, and print out an error message; this can be a tremendous aid in debugging.

It is also important to note that if there is no **`break`** statement at the end of a `case` statement, execution will fall through to the next `case` statement. This is sometimes necessary, but usually is an error. If you decide to let execution fall through, be sure to put a comment, indicating that you didn't just forget the `break`.

Demonstrating the switch statement without break.

```
// This Program is Written By Husain Gholoom
// The function of this program is to demonstrate  a
// a case statement without the break Statements.

#include <iostream>
using namespace std;

int main()
{
  unsigned short int number;
  cout << "Enter a number between 1 and 5: ";
  cin >> number;
  switch (number)
  {
    case 0:   cout << "Too small, sorry!";
              break;
    case 5:  cout << "Good job!\n";  // fall through
    case 4:  cout << "Nice Pick!\n"; // fall through
    case 3:  cout << "Excellent!\n"; // fall through
    case 2:  cout << "Masterful!\n"; // fall through
    case 1:  cout << "Incredible!\n";
             break;
    default: cout << "Too large!\n";
             break;
  }
   cout << "\n\n";
   return 0;
}
```

Output:

```
Enter a number between 1 and 5: 5
Good job!
Nice Pick!
Excellent!
Masterful!
Incredible!

Enter a number between 1 and 5: 8
Too large!
```

Demonstrating the switch statement.

```
// This Program is Written By Husain Gholoom
// The function of this program is to simulate a
// Simple calculator

#include<iostream>
Using namespace std;
int main( )
{
        float Operand1, Operand2,  Result;
        char Operator;
        cout<<"The function of this program is to simulate a Simple calculator. \n";
        cout<< "Enter the First Number ";
        cin>>Operand1;
        cout<< "Enter the Second Number ";
        cin>>Operand2;

        cout<< "Enter the Mathematical Operation Desired ( + , - , * , / ) ";
        cin>>Operator;

        switch ( Operator ) { // consider operation
        case '+': Result = Operand1 + Operand2;
              cout<<"The Result of Adding  "<<Operand1 <<"  and  "<< Operand2 <<
                  "  is  "<< Result<<endl;
                        break;
        case '-':  Result = Operand1 - Operand2;
                        cout<<"The Result of Subtracting  "<<Operand1 <<"  and  "<<
                    Operand2 << "  is  "<< Result<<endl;
                        break;
        case '*': Result = Operand1 * Operand2;
              cout<<"The Result of Multiplying  "<<Operand1 <<"  and  "<<
              Operand2 << "  is  "<< Result<<endl;
                        break;
        case '/': Result = Operand1 / Operand2; // no protection against Op2 == 0
              cout<<"The Result of Dividing  "<<Operand1 <<"  and  "<< Operand2
                        << "  is  "<< Result<<endl;
                        break;
        default:  Result = INT_MIN; // return flag if operation is invalid
                    cout<<"The Invalid Mathematical Operator  "<<endl;
                    break;  }


         return 0;
}
```

Menus

You can use switch statement to create menu-driven programs. A menu-driven program allows the user to determine the course of action by selecting it from a list of actions.

Example

```
// - Menu with switch control structure.

#include <iostream>
using namespace std;
int main()
{
        int choice;

        cout << "MENU\n\n";

        cout << "1." << "\t" << "Lobster\n";
        cout << "2." << "\t" << "Steak\n";
        cout << "3." << "\t" << "Turkey\n";
        cout << "4." << "\t" << "Hamburger\n";
        cout << "5." << "\t" << "Vegetarian\n\n";

        cout << "Choose your dinner entree: ";
        cin >> choice;

        cout << endl;

        switch (choice)
        {
                case 1: cout << "Lobster is my favorite! Dig in!\n\n";
                break;

                case 2: cout << "Yummy! Steak is great...\n"
                   << "but limit yourself to once a week," << endl
                   << "or risk the chance of high cholesterol!\n\n";
                break;

                case 3: cout << "Turkey is healthier than steak! ...Enjoy!\n\n";
                break;

                case 4: cout << "Hamburger is another form of steak. :-)\n\n";
                break;

                case 5: cout <<  "Finally, a vegetarian is in the house!\n\n";
                break;

                default: cout << "Invalid number, please enter a number"
                    << " from the entries above.\n\n";
                break;  }
        return 0;               }
```

Sample Run

```
MENU

1.      Lobster
2.      Steak
3.      Turkey
4.      Hamburger
5.      Vegetarian

Choose your dinner entree: 2

Yummy! Steak is great...
but limit yourself to once a week,
or risk the chance of high cholesterol!

Press any key to continue . . .
```

Sample Run

```
MENU

1.      Lobster
2.      Steak
3.      Turkey
4.      Hamburger
5.      Vegetarian

Choose your dinner entree: 6

Invalid number, please enter a number from the entrees above.

Press any key to continue . . .
```

```
//   Menu using if, else if, else statements.

#include <iostream>
using namespace std;

int main()
{
      int num;

      cout << "MENU\n\n";
      cout << "1." << "\t" << "Lobster\n";
      cout << "2." << "\t" << "Steak\n";
      cout << "3." << "\t" << "Turkey\n";
      cout << "4." << "\t" << "Hamburger\n";
      cout << "5." << "\t" << "Vegetarian\n\n";
      cout << "Choose your dinner entree: ";
      cin >> num;

      cout << endl;

      if (num == 1)
            {
               cout << "Lobster is my favorite! Dig in!\n\n";
            }

      else if (num == 2)
            {
               cout << "Yummy! Steak is great...\n"
                    << "but limit yourself to once a week,\n"
                    << "or risk the chance of high cholesterol!\n\n";
            }

      else if (num == 3)
            {
               cout << "Turkey is healthier than steak! ...Enjoy!\n\n";
            }

      else if (num == 4)
            {
               cout << "Hamburger is another form of steak. :-)\n\n";
            }

      else if (num == 5)
            {
               cout << "Finally, a vegetarian is in the house!\n\n";
            }

      else
            {
               cout << "Invalid number, please enter a number"
                    << " from the entrees above.\n\n";
            }
      return 0;
}
```

Sample Run

```
MENU

1.      Lobster
2.      Steak
3.      Turkey
4.      Hamburger
5.      Vegetarian

Choose your dinner entree: 5

Finally, a vegetarian is in the house!

Press any key to continue . . .
```

Sample Run

```
MENU

1.      Lobster
2.      Steak
3.      Turkey
4.      Hamburger
5.      Vegetarian

Choose your dinner entree: 0

Invalid number, please enter a number from the entrees above.

Press any key to continue . . .
```

Nested Switch :

It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

```
int x = 2 , y = 3;

switch(x) {
        case 1:  cout << "x is 1\n"; break;
        case 2:
        case 3:  switch (y) {
                    case 1 :  cout << "x is 2 or 3 , y is 1 ";  break;
                    case 2 :  cout << "x is 2 or 3 , y is 2 ";  break;
                    default:  cout << "x is 2, or 3 , y is not 1 or 2 ";
                                    break;

                 }  break;

  default:  cout << "x is not 1, 2, or 3";  break ;  }
```

Rewrite the above code segment using if .. else statement !!

Switch and Range

Consider the following if statement

```
cout << "Enter your test score and I will tell you\n";
      cout << "the letter grade you earned: ";
      cin >> testScore;
      if ( testScore >=0  && testScore < 60)
               cout << "Your grade is F.\n";
      else if (testScore >=60  &&  testScore < 70)
               cout << "Your grade is D.\n";
      else if (testScore >=70  &&  testScore < 80)
               cout << "Your grade is C.\n";
      else if (testScore >=80  && testScore < 90)
               cout << "Your grade is B.\n";
      else if (testScore >=90  && testScore <= 100)
       cout << "Your grade is A.\n";
else  cout << "That is not a valid score.\n";
```

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

int main() {

        int testScore;
        cout << "Enter your test score and I will tell you\n";
        cout << "the letter grade you earned: ";
        cin >> testScore;
        switch (testScore) {
        case  0 ...  59:
                cout << "Your grade is F.\n";
                break;
        case 60 ... 69 :
                cout << "Your grade is D.\n";
                break;
        case  70 ... 79:
                cout << "Your grade is C.\n";
                break;
        case 80 ... 89 :
                cout << "Your grade is B.\n";
                break;
        case 90 ...  100:
                cout << "Your grade is A.\n";
                break;
        default:
                cout << "That score isn't valid\n";
        }
        return 0;
}
```

More about blocks and scope

1. The scope of a variable is the part of the program where the variable may be used.
2. The scope of a variable is the innermost block in which it is defined, from the point of definition to the end of that block.
3. Note: the body of the main function is just one big block.

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

int main()
{

        double income; //scope of income is red + blue!
        cout << "What is your annual income? ";
        cin >> income;
        if (income >= 35000)
        {
           int years; //scope of years is blue;
           cout << "How many years at current job? ";
           cin >> years;
           if (years > 5)
              cout << "You qualify.\n";
           else
             cout << "You do not qualify.\n";
        }
  else
          cout << "You do not qualify.\n";  // Can Not Access
          cout << "Thanks for applying.\n"; // years


        return 0;
}
```

Variables with the same name

1. In an inner block, a variable is allowed to have the same name as a variable in the outer block.
2. When in the inner block, the outer variable is not available (it is hidden).
3. Not good style: difficult to trace code and find bugs

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

int main()
{

	int number;
	cout << "Enter a number greater than 0: ";
	cin >> number;
	if (number > 0)
	{
		int number; // another variable named number
		cout << "Now enter another number ";
		cin >> number;
		cout << "The second number you entered was ";
		cout << number << endl;
	}

	cout << "Your first number was " << number << endl;

	return 0;
}
```

Sample Run

```
Enter a number greater than 0: 10
Now enter another number 20
The second number you entered was 20
Your first number was 10
```

Exercises :

What is the value of the variable c after the switch statement below?

```
int x = 3;

switch ( x ) {
  case 1:  c = 'A'; break;
  case 2:  c = 'B'; break;
  case 3:  c = 'C'; break;
  default: c = 'F'; break;
}

cout << c << endl;
```

Is the following a valid switch statement. If not , explain why ?

```
int n = 0;

switch ( n <= 2)  {
 case 0 : cout << " Draw " << endl ; break;
 case 1 : cout << " Win "  << endl ; break;
}
```

What is the output of the following code :

```
int  Z  = 3 ?
int  b;

switch(Z)

{
        case 1: b = 3;      break;
        case 2: b = 4;      break;
        case 3: b = 5;
        case 4: b = 6 ;
        break;
        default: b = 7;
}

cout << b ;
```

Rewrite the above code segment using if .. else statement.

Convert the following if/else if statement into a switch statement:

```
if (choice == 1)     {
   cout << fixed << showpoint << setprecision(2);     }
else if ((choice == 2) || (choice == 3))     {
   cout << fixed << showpoint << setprecision(4);     }
else if (choice == 4)     {
   cout << fixed << showpoint << setprecision(6);     }
else {
   cout << fixed << showpoint << setprecision(8);     }
```

Convert the following if/else if statement into a switch statement:

```
if (choice >=  1   &&   <= 10)     {
   cout << fixed << showpoint << setprecision(2);     }
else if ((choice == 11)      {
   cout << fixed << showpoint << setprecision(4);     }
else if (choice >11  &&   < =20 )     {
   cout << fixed << showpoint << setprecision(6);     }
else {
   cout << fixed << showpoint << setprecision(8);     }
```