Switch

Lecture 18

`switch` Statements

- Sometimes you need to compare one variable to many different options
- For example, in the fraction lab you need to compare a user input to 4 different characters to determine which operation to perform
- You could use a series of `if`/`else` statements, or you could use a `switch` statement

Example (if-else)

```
int choice;

cout << "Enter 1 to say hello" << endl;
cout << "Enter 2 to say howdy" << endl;
cout << "Enter 3 to say yo" << endl;

cin >> choice;

if ( choice == 1 )
{
     cout << "hello" << endl;
}
else if ( choice == 2 )
{
     cout << "howdy" << endl;
}
else if ( choice == 3 )
{
     cout << "yo" << endl;
}
else
{
     cout << "That is not a valid selection!" << endl;
}
```

Example (`switch`)

```
int choice;

cout << "Enter 1 to say hello" << endl;
cout << "Enter 2 to say howdy" << endl;
cout << "Enter 3 to say yo" << endl;

cin >> choice;

switch ( choice )
{
case 1:
      cout << "hello" << endl;
      break;
case 2:
      cout << "howdy" << endl;
      break;
case 3:
      cout << "yo" << endl;
      break;
default:
      cout << "That is not a valid selection!" << endl;
      break;
}
```

C++ looks at the current value of the **`choice`** variable

Then compares choice to each of the **`case`** statements to see if one is equal to choice

If none of them are equal, it uses the **`default`** entry instead

General Form

```
switch ( VARIABLE )
{
case CONSTANT_1:
    STATEMENTS_IF_VARIABLE_EQUALS_CONSTANT_1
    break;
case CONSTANT_2:
    STATEMENTS_IF_VARIABLE_EQUALS_CONSTANT_2
    break;

    …
case CONSTANT_N:
    STATEMENTS_IF_VARIABLE_EQUALS_CONSTANT_N
    break;
default:
    STATEMENTS_IF_NO_CONSTANTS_EQUAL_VARIABLE
    break;
}
```

Notes

- The values in each `case` statement must be constants, they can't be variables
  - They can be expressions, so long as the expressions consist only of constants
  - This includes fixed values and `const` variables
- You can have any number of `case` statements, but they can't overlap (can't have multiple cases with the same constant value)
- The value in the `switch` statement is usually a variable, but can be an expression also (doesn't need to be constant)

`break`

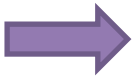
- The break statement tells C++ to "break out" of the `switch` statement
- In other words, when a `break` is executed the program immediately stops the `switch` statement and continues executing the program at the statement after the closing `}` for the `switch` block
- If you leave off the `break` statement, it will continue executing the next statements in the next `case`

Missing a break

```
int choice;

cout << "Enter 1 to say hello" << endl;
cout << "Enter 2 to say howdy" << endl;
cout << "Enter 3 to say yo" << endl;

cin >> choice;

switch ( choice )
{
case 1:
     cout << "hello" << endl;
     break;
case 2:
     cout << "howdy" << endl;
case 3:
     cout << "yo" << endl;
     break;
default:
     cout << "That is not a valid selection!" << endl;
     break;
}
```

Missing a break in case 2

Step-by-Step Example



```
int input;

cout << "Enter a number: ";
cin >> input;

switch ( input )
{
case 1:
     cout << "1" << endl;
case 2:
     cout << "2" << endl;
case 3:
     cout << "3" << endl;
case 4:
     cout << "4" << endl;
     break;
default:
     cout << "You did not enter 1, 2, 3, or 4!" << endl;
     break;
}

cout << "Done." << endl;
```

```
Enter a number: 2
2
3
4
Done.
```

Fall Through Cases

- It is sometimes useful to take advantage of these *fall through cases*
- In general, you might do this if you have the same statements for two (or more) cases
- One of the most common uses of this is when processing character inputs and you want to do the same thing for both uppercase and lowercase letters

Example

```
char choice;
cout << "Enter y or n: ";
cin >> choice;

switch ( choice )
{
case 'y':
case 'Y':
    cout << "You entered yes." << endl;
    break;
case 'n':
case 'N':
    cout << "You entered no." << endl;
    break;
default:
    cout << "That is not a valid selection!" << endl;
    break;
}
```

Exercise

Write a simple calculator that asks the user for two numbers and an operation (+, -, *, /). Use a `switch` statement to determine which operation was entered and compute the result of the operation.

Answer

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

int main()
{
	double input1, input2, answer;
	char op;

	cout << "Enter the first number: ";
	cin >> input1;
	cout << "Enter the operation (+, -, *, /): ";
	cin >> op;
	cout << "Enter the second number: ";
	cin >> input2;

	switch ( op )
	{
	case '+':
		answer = input1 + input2;
		break;
	case '-':
		answer = input1 - input2;
		break;
	case '*':
		answer = input1 * input2;
		break;
	case '/':
		if ( input2 != 0 )
		{
			answer = input1 / input2;
		}
		else
		{
			answer = 0;
		}
		break;
	default:
		cout << "Invalid operation!" << endl;
		return 0;
		break;
	}

	cout << "The answer is: " << answer << endl;

	return 0;
}
```

Wrap Up

- A `switch` statement is useful when you have many different options based on the value of a single variable
  - Menu processing is the most common reason to use a switch statement
- If a matching constant is found in the `case` statements, the program continues executing at that `case` statement
- Otherwise, it continues executing at the `default` case statement
- Except in special cases, each case block should end with a `break` statement