Additional Logic Handling Techniques

CS 10A – PROGRAMMING LOGIC PART 2

Go To and Labels

- The most reviled form of logic. The greatest source of bugs in all of programming.
- Code in all languages run from top to bottom in a sequential order. Go To hijacks this order by allowing code to jump forward or back to specified Labels.
  - goto someLabel; // Now go to where someLabel is
    - someLabel: // Program continues from here, placed anywhere
- Very rarely is Go To ever preferred, let alone needed, over other forms of logic control. NEVER USE THIS IF YOU CAN AVOID IT. (Do not use Go To in any assignment I hand out.)

How to Use Go To and Labels

Program

```
int y = 0;
int main()
{
start_point:
        y++;
        if(y < 5)
                goto start_point;
        cout << y << endl;
        return 0;
}
```

Console

- ./a.exe

```
5
```

Switch-Case Logic

- Switch-Case is a more minimalistic version of if/else if/else logic. Useful for handling logic blocks containing numerous else if statements and are readily expandable.
- If your logic consists only checking the specific value of a variable, such as selecting menu options, then switch-case is recommended over if/else if/else.
- Marginally faster (by insignificant amounts) than if/else
- Their main drawback is that they can only handle int and char variables for use. They cannot handle additional Boolean logic either. This is true for C/C++ only.

Switch-Case Syntax

```
switch(variable)                    // Specify the variable you want to use, either an int type or char type
{                                   // Use brackets to encase the switch-case block of logic
        case value0:                // Block to execute based on variable value
                                    // If variable == value0, then the code here executes
                break;              // break is a keyword that signifies the end of a case, the remaining cases are skipped
        case value1:                // You can include as many cases as you want
                                    // Of course, you can include as many lines to execute as you want
                break;
        case value2:                // Cases are basically the same as labels, with switch acting like a Go To
                                    // If there is no break, then the next lines (regardless of being in other cases) execute as well
        case value3:                // The switch case block only exits upon hitting a break
                                    // So if variable == value2, then case value3 gets executed as well since
                break;              // no break ends a case of value2.
        default:                    // default is basically the else of switch-case

                break;
}
```

Utilizing Switch-Case

Switch-Case Example

```
int x = 0;
int main()
{
        switch(x)
        {       // If using a char type, use single quotes, i.e. 'a'
                case 0:
                        cout << "x is 0" << endl;
                        break;
                case 1:
                        cout << "x is 1" << endl;
                        break;
                default:
                        cout << "x is something else" << endl;
                        break;
        }
        return 0;
}
```

If/Else If/Else Equivalent

```
int x = 0;
int main()
{
        if(x == 0)
                cout << "x is 0" << endl;
        else if(x == 1)
                cout << "x is 1" << endl;
        else
                cout << "x is something else" << endl;
        return 0;
        /*
          Additional cases is the same as additional else if statements.
          Switch-case statements are much easier to read since they only
          allow one kind of logic, which is ==.
        */
}
```

Multiple Choice on Switch-Case

```
switch(variable)
{
        case ‘A’:
        case ‘a’:
                // You can stack multiple cases over a set of commands
                // A case is just a label, so they alone execute nothing and won’t impact the program
                break;
        case ‘B’:
        case ‘b’:
        case ‘C’:
        case ‘c’:
                // This allows you to allow multiple ways to select that particular block
                // Commonly used to disable case sensitive user inputs
                break;
}
```