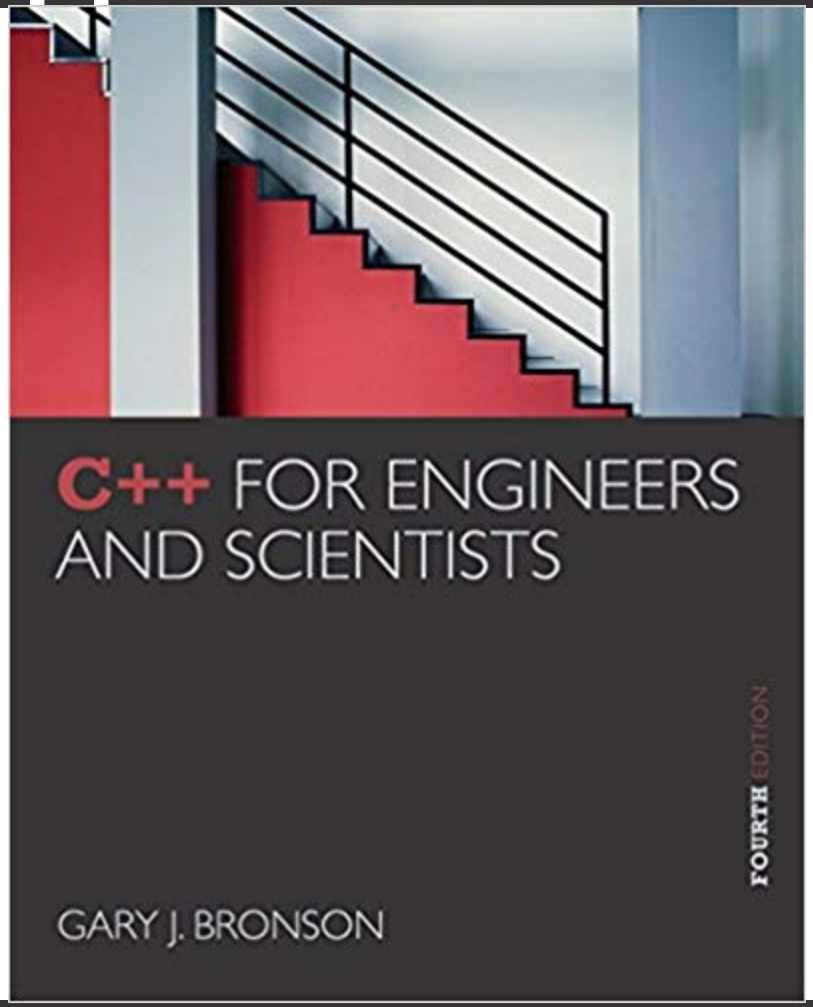
ELEG 1043
Computer Applications in Engineering



Chapter 4:
Selection Structures

Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

Objectives

- In this chapter, you will learn about:
  - Selection criteria
  - The **`if-else`** statement
  - Nested **`if`** statements
  - The **`switch`** statement
  - Program testing
  - Common programming errors

The `switch` Statement

https://www.programiz.com/c-programming/c-switch-case-statement

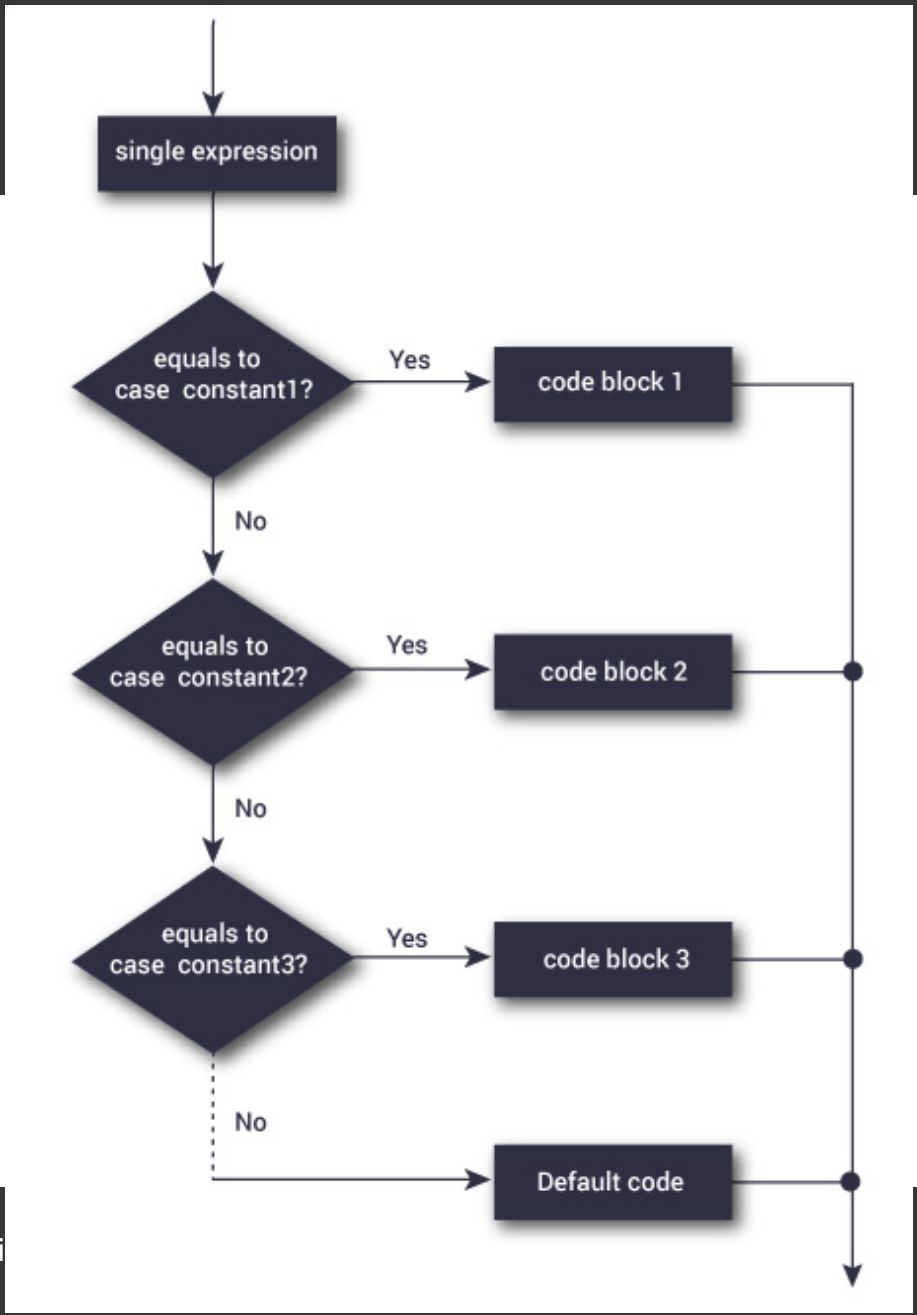
The `switch` Statement

- **`switch`** statement: Provides for one selection from many alternatives
- **`switch`** keyword starts the statement
  - Is followed by the expression to be evaluated
- **`case`** keyword identifies a value to be compared to the switch expression
  - When a match is found, statements in this **`case`** block are executed

The `switch` Statement (continued)

- **`default`** case is executed if no other case value matches were found
- **`default`** case is optional

The `switch` Statement



The `switch` Statement (continued)

```
 switch(expression)
{
    case constant-expression :
        statement(s);
        break;
    case constant-expression :
        statement(s);
        break;
     …
    default :
        statement(s); }
```

Example 4

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

int main()
{
     int num=2;

    switch(num)
    {
          case 1:
                cout<<"Case1: Value is: ”<<num;  break;
          case 2:
                cout<<"Case2: Value is: ”<<num;  break;
          case 3:
                cout<<"Case3: Value is: ”<<num;  break;
          default:
                cout<<“Default: Value is: ”<<num;
    }

    return 0;

}
```

A Closer Look: Program Testing

- Theory: A comprehensive set of test runs would test all combinations of input and computations, and would reveal all errors
- Reality: There are too many combinations to test for any program except a very simple one
- Example:
  - One program with 10 modules, each with five `if` statements, always called in the same order
  - There are 2^5 paths through each module, and more than 2^{50} paths through the program!

A Closer Look: Program Testing (continued)

- Conclusion: there is no error-free program, only one in which no errors have recently been encountered

Common Programming Errors

- Using the assignment operator (=) instead of the relational operator (==) for an equality test
- Placing a semicolon immediately after the condition
- Assuming a structural problem with an **`if-else`** causes the error instead of focusing on the data value being tested

Summary

- Relational expressions, or conditions, are used to compare operands
- If the relation expression is true, its value is `1`; if false, its value is `0`
- Use logical operators `&&` (AND), `||` (OR), and `!` (NOT) to construct complex conditions
- `if-else` allows selection between two alternatives

Summary (continued)

- An `if` expression that evaluates to `0` is false; if non-zero, it is true
- `if` statements can be nested
- Chained `if` statement provides a multiway selection
- Compound statement: contains any number of individual statements enclosed in braces

Summary (continued)

- **`switch`** statement: Provides a multiway selection
- **`switch`** expression: Evaluated and compared to each **`case`** value
  - If a match is found, execution begins at that case's statements and continues unless a **`break`** is encountered