Intro to Programming
CGS 3406
Computer Programming

- Process to instruct computers to perform a given task
- Instructions are generally in the form of a programming language
- Source code is a set of instructions in a given programming language that expresses the task (e.g., computation)
- We are using the C++ language and standard library
- Programming is a process
  - Sequence of steps to achieve an end
  - Includes
    - Problem solving (designing a solution)
    - Refinement
      - Debugging
      - Adding features
- Many steps in the programming process are similar regardless of the language
Computer Programming in C++
Statement

- Complete executable unit of a program
- Typically composed of one or more expressions
- Simple statement
  - End in a semicolon
  - expression;
  - cout << "hello world\n";
- Compound statement
  - Set of zero or more statements inside curly braces {}
  - { statement₁; statement₂; statement₃; }
Declaration Statements

- Introduce a name into a program
- Names cannot be keywords
- Keywords
  - Reserved names that are part of the language
  - Have special meaning
  - Cannot be used as identifiers in your program
- Examples
  - while
  - if
  - else
  - ...

Keywords

- while
- if
- else
Declaration Statements

- Declare before use

- Variables
  - type and identifier
    ```
    int x;
    ```
  - `const` declares that the variable cannot change
    ```
    const int x = 5;
    ```
  - Attempting to modify a `const` variable will result in a compiler error
  - Note that literals are also constant

- Functions
  - Declaration
  - Definition
Declaration Statements

type name ( parameter1, parameter2, ... )
{
    statements
}

• Functions
  – Declaration
    • Specifies the types input output types
  – Definition
    • Statements that define what the function does
Control Structures

- Default mode of execution is sequential order
- Alter sequential execution
  - Selection
    - Statements executed depending on the program state
    - Select code to execute based on test
  - Iteration
    - Repeated execution code until program reaches a given state
if-else
(selection)

• if X is true do Y else do Z
• if a<b update current maximum

• Syntax

  if (expression)
      statement₁
  else if (expression)
      statement₂
  ...
  else
      statementₙ
**switch**

*(selection)*

- Select code based on case
- Often convenient for occasions in which there are multiple cases to choose from
- Syntax

```java
switch (expression) {
    case constant:
        statement(s)
    case constant:
        statement(s)
    ...
        // (as many case labels as needed)
    default: // optional label
        statements
}
```
switch (selection)

switch (expression) {
    case constant:
        statement(s)
    case constant:
        statement(s)

    ...  (as many case labels as needed)

    default:          // optional label
        statements
}

- Evaluates the expression, and then compares it to the values in the case labels
  - If it finds a match, execution of code jumps to that case label
The values in case labels must be constants, and may only be integer types, which means that you
- integer, char, or enumerations (not yet discussed)
- case label must be a literal or a variable declared to be const
- Note: You may not have case labels with regular variables, strings, floating point literals, operations, or function calls

If you want to execute code only in the case that you jump to, end the case with a `break` statement, otherwise execution of code will "fall through" to the next case