# Introduction to C++ Syntax

## Comments

Two Styles

// the remainder of the line is ignored by the compiler

/* */ all of the text between the /* and */ is ignored by the compiler

## Preprocessor Directive

```
#include<iostream>
```

we will use this at the beginning of every source file that we create in which we would like to use the **cin** and **cout** objects to read input from the standard input device and output to the standard output device

# main

```
int main() {
   return 0;
}
```

- main – the name of the function
- int – the return type of the main function
- () – the (empty) parameter list for the main function. Note: the main function can also take exactly two parameters (int, char *)
- return 0; – the main function returns a 0 to signify successful program completion

## The cout Object and Stream Insertion Operator

- Stream Insertion Operator, <<, used to insert data into the output stream
- cout – object in the iostream header file used to output data to the screen

  Example: cout << “Hello”;

## The cout Object and Stream Insertion Operator

- Place the following preprocessor directive in all programs which use the cout object.

  `#include<iostream>`

- Place the following line in all programs which use the cout object.

  `using std::cout;`

  or

  `using namespace std;` - actually… NEVER do this!

## Some Escape Sequences

- \n – Go to the next line
- \r – Return to the beginning of the current line
- \t – Tab to the next default tab stop
- \a – Ring the system bell
- \” – Used to output a “
- \\ - Used to output a \

## The endl stream manipulator

- not an escape sequence – don’t put in quotes
- flushes the output stream – output will be generated immediately
- The endl stream manipulator is also a member of the std namespace, so place using std::endl; in programs that use it

## Variables vs. Constants

- Variable – label for memory that holds a value that may change
- Constant – a value that is fixed.

## Variable Data Types

int – integers (positive and negative whole numbers)

float – real numbers

double – real numbers, more space than a float

char – any ASCII character

bool – logical, true or false

## Declaring a Variable

*DataType variablename*;

- This tells the compiler that *variablename* will be a variable to hold values of the type *DataType*
- The variable name must be a valid identifier
- No value will be placed in the variable

## Valid Identifiers

- identifier – name for a variable, function or constant. May be any combination of alphanumeric characters and the underscore that does NOT begin with a numeral and is not a C++ keyword.

  Note: C++ is CASE SENSITIVE!

  – *Valid* identifiers – Joe, x, august1st, _underscore,OneMore

  – *Invalid* identifiers – 4thOfJuly, c++, int, one more

## Valid Identifiers

- identifier – name for a variable, function or constant. May be any combination of alphanumeric characters and the underscore that does NOT begin with a numeral and is not a C++ keyword.

  Google Style Guide – begin with a lowercase letter and use underscores between words

## Initializing Variables

- <u>initialize</u> – giving a value before use. Example: x = 3;
- <u>assignment operator</u> (=) commutes from right to left. The value on the LHS will be changed.
- cascading assignment operations
  x = y = z = 0;
- initializing and declaring in one statement

  Example: int x = 3;

# Constant Variables

- Placing the const qualifier in front of a variable's declaration ensures that the variable's value will not change during the program execution.
- Constant variables must be initialized with their declaration.
- Constant variables allow for easier program modification.
- Naming convention *kCamelCase*

# Arithmetic Operations

- *, /, %, +, –
- Follow the Order of Operations
- Note: You can put an arithmetic operation in a cout statement.  For example,

```
cout << x / 2;
```

will print the value of x/2.

## Arithmetic Operations

Operations are type dependent

| | | |
|---|---|---|
| 1 / 4 | evaluates to | 0 |
| 1 / 4.0 | evaluates to | 0.25 |
| 1.0 / 4 | evaluates to | 0.25 |

## More Assignment Operators

+=, -=, *=, /=, %=

| | | |
|---|---|---|
| x += y; | is equivalent to | x = x + y; |
| x -= y; | is equivalent to | x = x - y; |
| x *= y; | is equivalent to | x = x * y; |
| x /= y; | is equivalent to | x = x / y; |
| x %= y; | is equivalent to | x = x % y; |

## Increment and Decrement Operators

++, --

++  Adds one to the variable

--  Subtracts one from the variable

## Increment and Decrement Operators

- Pre-Increment, Pre-Decrement

  When the operator precedes the variable, the new value is returned by the operation

  Examples: ++i or --i

## Increment and Decrement Operators

- Post-Increment, Post-Decrement

  When the operator follows the variable, the old value is returned by the operation

  Examples:  i++ or i--

## Increment and Decrement

Examples

```
int x = 3, y;
y = x++ + 2;
cout << x << “ “ << y;
//
```

## Increment and Decrement

Examples

```
int x = 3, y;
y = x++ + 2;
cout << x << “ “ << y;
// Output: 4 5
```

## Increment and Decrement

Examples

```
int x = 3, y;
y = ++x + 2;
cout << x << “ “ << y;
//
```

## Increment and Decrement

<u>Examples</u>

```
int x = 3, y;
y = ++x + 2;
cout << x << “ “ << y;
// Output: 4 6
```

# Increment and Decrement

```
int x = 1;
cout << x++ << “\n”;        //
   //

cout << ++x << “\n”;        //
   //

cout << x-- << “\n”;        //
   //

cout << --x << “\n”;        //
   //
```

# Increment and Decrement

```
int x = 1;
cout << x++ << “\n”;      // Output: 1
   // x becomes 2

cout << ++x << “\n”;      // Output: 3
   // x becomes 3

cout << x-- << “\n”;      // Output: 3
   // x becomes 2

cout << --x << “\n”;      // Output: 1
   // x becomes 1
```

# Type Casting

- When performing an arithmetic operation on integer and real variables, the integer variable is <u>implicitly cast</u> as a real variable.

## Type Casting

- Methods for explicitly casting variables:

  We'll use:

  static_cast< *type* > ( *variable* )

  or

  ( *type* ) *variable*

  Others:

  const_cast< *type* > ( *variable )*

  dynamic_cast < *type* > ( *variable )*

  reinterpret_cast < *type* > ( *variable )*

# The cin Object and the Stream Extraction Operator

- stream extraction operator, >>, used to obtain data from the input stream
- cin - object in the iostream header file. Used to obtain input from the  standard input device (keyboard). The value of the variable on the right hand side will be replaced

Example: cin >> x;

## The cin Object and the Stream Extraction Operator

- The cin object is also a member of the std namespace, so place using std::cin; in files that use this object.

# Terminology

- Syntax Error – error in usage of the language (like a grammatical error). Our compiler will catch our syntax errors.
- Logic Error – syntactically correct code that does not perform the desired task.

# Additional Notes on Writing C++ Code

- Don’t pass column 80 in your source code
- Indent code two spaces inside of a block (e.g. in the body of a function)
- Indent subsequent lines of a command if the command takes more than one line