C++ Programming Basics

C++ Lecture 1

Nick Matthews

Adam Kohl

Course Goals

- Introduction to integrated development environments (IDEs)
- Crash course in C++ programming
- Workable understanding of variables, functions, and objects

Class Structure

- Class time from 9-11am and 2-4pm on Monday, Wednesday, and Friday
- Class time is used for lectures and worktime
- Daily activities reinforcing concepts
- Concepts will build on each other so ask questions early

Motivation

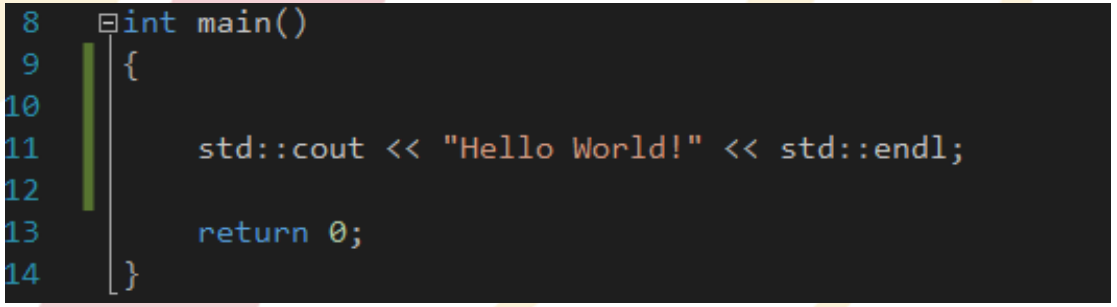
- C++ is a challenging but powerful language
- Basis of many major software packages
- Concepts in this class extend to many other languages
- Today is devoted to the basic building blocks
  - Setting up the IDE
  - C++ Syntax
  - Using the "Includes" Statement
  - Commenting Code
  - Output to the Command Line

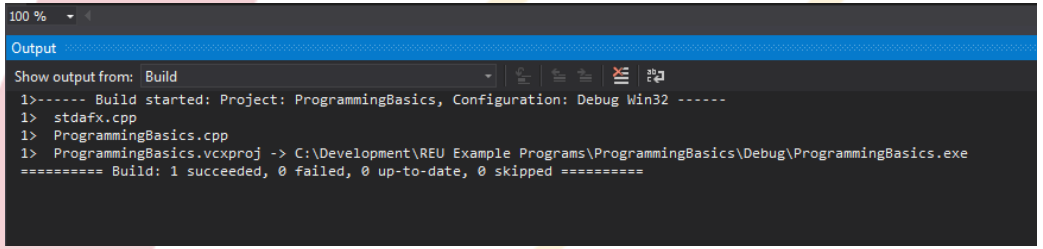
What is an IDE?

- Definition: An **Integrated Development Environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development.
  An IDE normally consists of a **source code editor**, **build automation tools** and a **debugger**.

What is an IDE

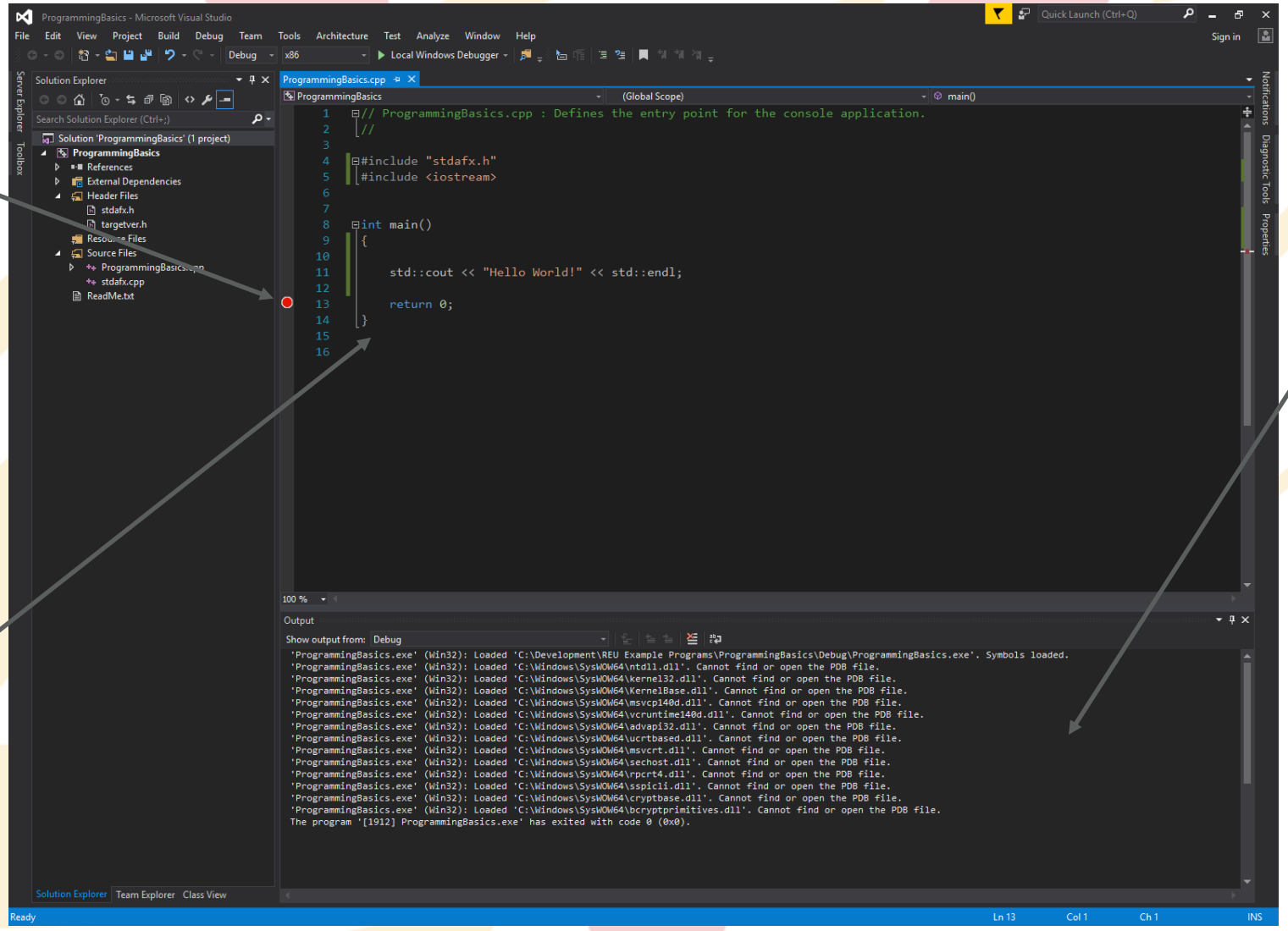
- Source Code – The letters, numbers, and symbols that make up a program
- Build Automation – Translates source code into computer speak
- Debugging – Looking though code while running to gain understanding





Visual Studio

Debugging Breakpoint



Source Code

Build Output

Checkpoint

- Open Visual Studio and and create a new visual C++ Win32 Console Application

C++ Syntax

- Program statements are executed line-by-line
- Lines are terminated with the a ;
- Everything starts in the main function
- Source code is translated into machine code prior to execution
- Syntax similar to other "curly brace languages" (C, Java, C#)

Checkpoint

- Obligatory "Hello World!" application. Enter code into IDE and run.
- Delete a ;
- What happens?
- Remove "return 0;"
- What happens?

```
4   #include "stdafx.h"
5   #include <iostream>
6
7
8   int main()
9   {
10
11      std::cout << "Hello World!" << std::endl;
12
13      return 0;
14  }
```

Code Comments

- Good programmers comment their code
- Comments explain in plain language what a portion of code does
- Comments are helpful to yourself and others when reading code

```
int main()
{

    std::cout << "Hello World!" << std::endl;   // This is a one line comment

    /*Tthis is a multi line comment.
    I
    I can
    I can go
    I can go on
    I can go on forever
    I can go on forever and
    I can go on forever and ever.......*/

    return 0;
}
```

Checkpoint

- Add your own comments to your Hello World program
- What happens when you run the program without the // or /**/ with the text?

Include Statements

- Remember C++ complies syntax line-by-line
- If we want to do something we need the syntax or machine code
- There are common operations that users want to perform
- Don't want to reinvent the wheel every time we want a printout
- Sooooo we use the standard library and include statements

Include Statements

- To specify what standard features we want we use #include
- Let's call and use all the functionality in our own program without having to write the code ourselves
- For example, in our Hello World program we used #include <iostream> to print to the command line

Checkpoint

- Uncomment the #include <iostream> and try to run the program

```
4   #include "stdafx.h"
5   //#include <iostream>
6
7
8   int main()
9   {
10
11      std::cout << "Hello World!" << std::endl;   // This is a one line comment
12
13      /*Tthis is a multi line comment.
14      I
15      I can
16      I can go
17      I can go on
18      I can go on forever
19      I can go on forever and
20      I can go on forever and ever.......*/
21
22      return 0;
23  }
```

Questions?

Assignment

- Play with different wording in your Hello World application
- Challenge: Can you split the words onto different lines?