C++ Basics

In this week

- MSVC++ 2010 Express: An IDE for C++
- Hello World: Your first C++ program
- Compiling, Linking and Running your program
- The memory unit of a computer
- **Variables**: declaration and initialization
- C++ Primitive Data Types and Type Casting
- Binary Operators: arithmetic expression
- Keyboard/Console Input and Output
- Unary Operators: ++ and -- operators

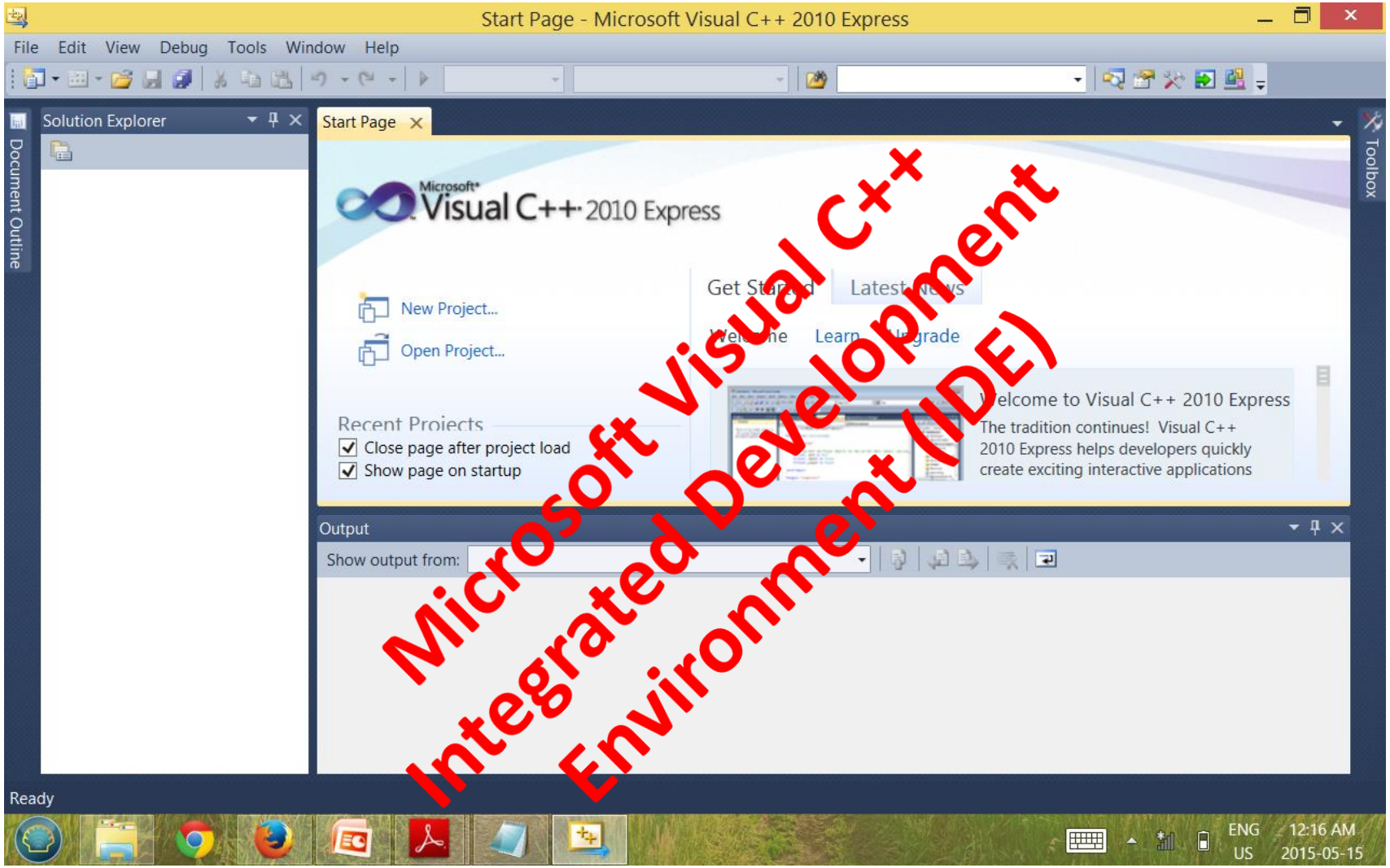
An IDE for C++ Programming

- In order to work with C++ programming language, we need an environment where we write a program, check the correctness of the program, and execute the program
- For this course, we will use Microsoft Visual C++ 2010 Express Edition which is freely available from Microsoft
- The simplest way to get Microsoft Visual C++ 2010 Express Edition is to **Google "Microsoft Visual C++ 2010 Express Edition offline installer"**

Starting Microsoft Visual C++

- In order to start Microsoft Visual C++, click on
  - Start Button
  - All Programs
  - Microsoft Visual Studio 2010 Express
  - Microsoft Visual C++ 2010 Express
- Once you do that Microsoft Visual C++ 2010 Express will start and you will find the following window...

Starting Microsoft Visual C++

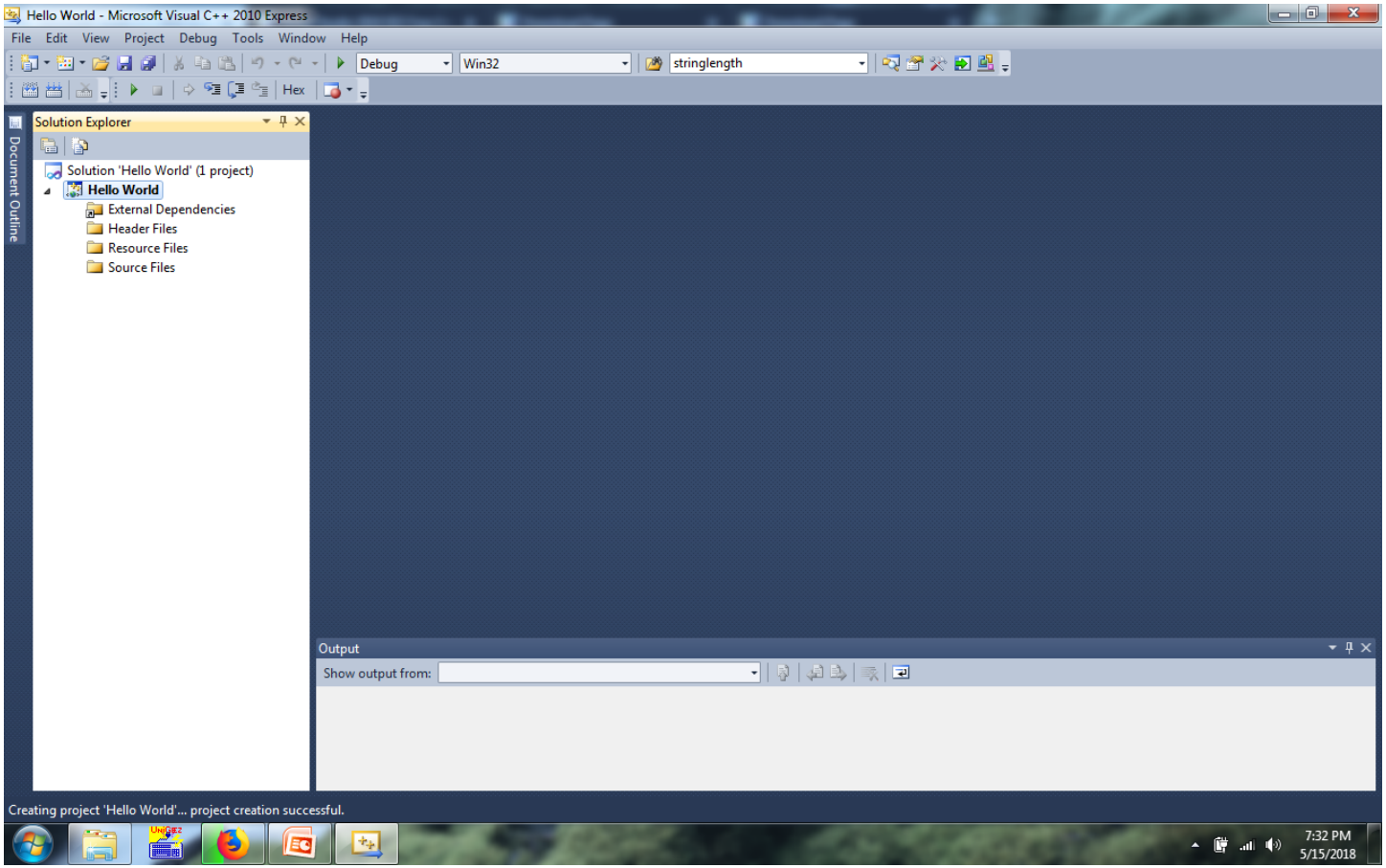


Hello World: Your First C++ Program

- In order to create your first C++ program click on
  - File → New → Project
    - ✓**Alternatively click on New Project on the window**
  - On the left side select **Visual C++** under Installed Templates
  - In the middle, select **Win32 Console Application**
  - Type a name for your project: Name it **Hello World**
  - Browse to a folder where you would like to save your project and click on **Select Folder**
  - Click **OK** and then click **Next**
  - Click on **Empty project** under Additional options
  - Finally click **Finish**

Hello World: Your first C++ program

- At this point, MS VC++ will show the start page of your project as follows



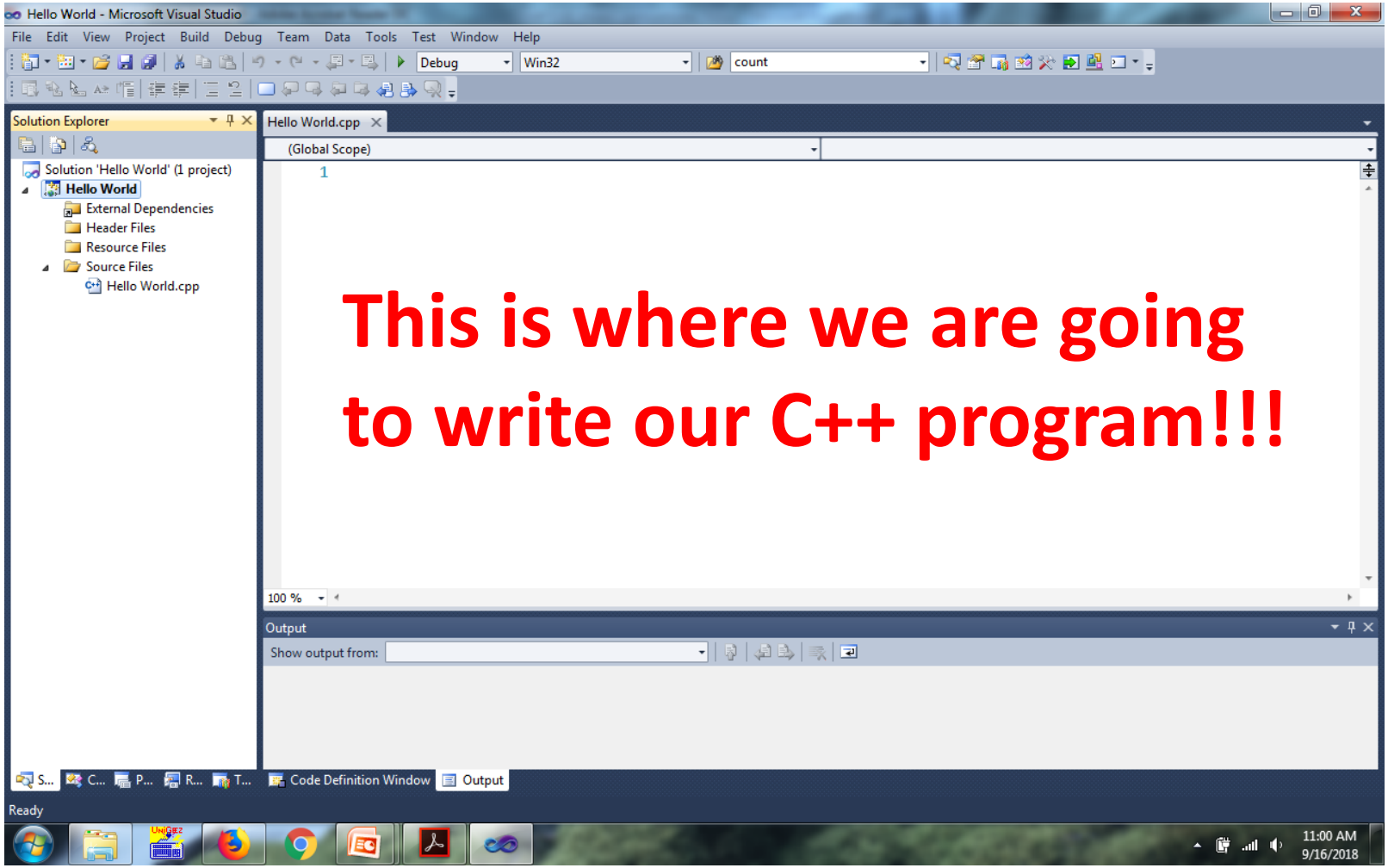
Hello World: Your first C++ program

- The new window will show three panes:
  - On the left pane will be a list of folders. You don't have to understand the details of all the folders for the time being
  - The middle pane is blank because we haven't created any C++ program yet. Once we create a C++ program, it will be opened in this middle pane
  - The bottom pane is called the output pane and will be discussed later...

Hello World: Your first C++ program

- Now, let us create a new C++ program inside our project
- To do so follow the following steps:
  - Click on **Project** on the menu bar
  - Click on **Add New Item**
  - Select **C++ File (.cpp)**
  - Type a name for the program. Name it **Hello World**
  - Click **Add**
- Now the middle pane will open an empty editor we just created shown below

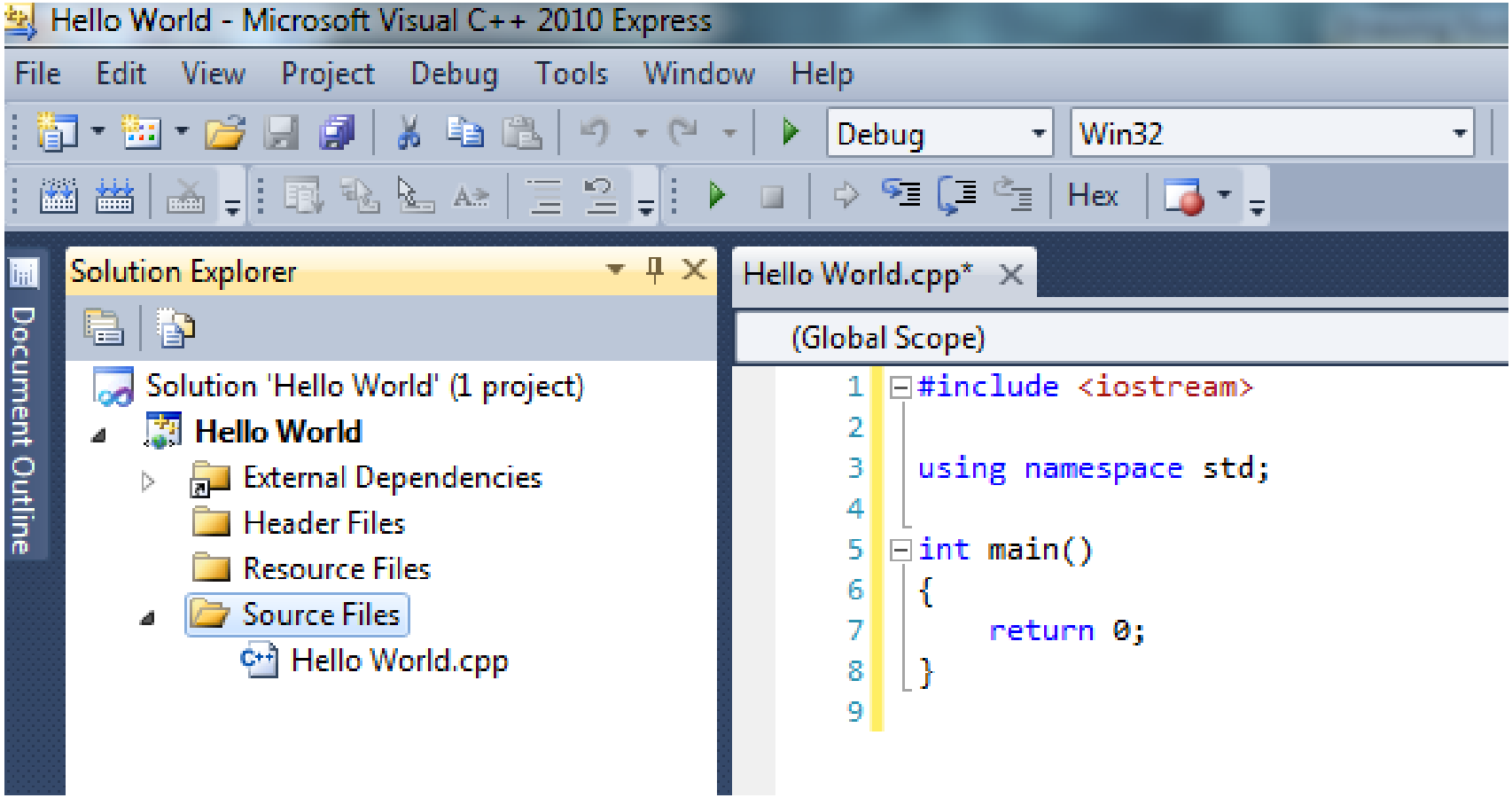
Hello World: Your first C++ program



Hello World: Your first C++ program

- Now let us write our first C++ program
- In C++ programming language, every program starts with what is known as **Main Program**
- So let us write the simplest possible C++ main program in order to demonstrate what a C++ main program looks like
- This program, shown below, shows a complete C++ main program that does nothing

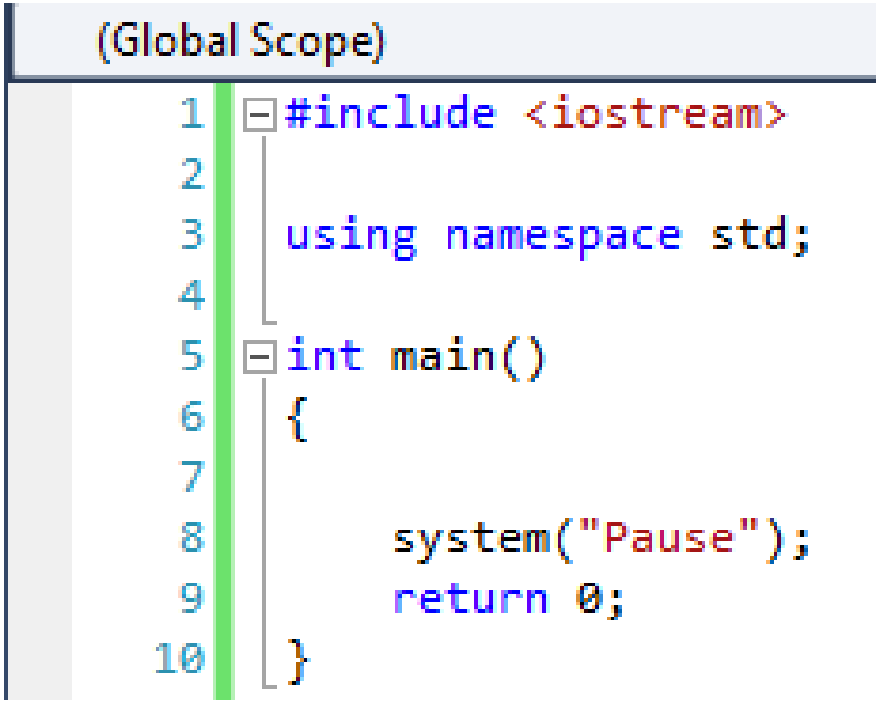
Hello World: Your first C++ program



Hello World: Your first C++ program

- In order to check the correctness of the program and execute (run) the program, press F5
- You will see a black screen flash and close automatically
- This black screen is known as the **output console (window)** of the program
- In order to see the output window, we will write a code that tells C++ to pause before closing the output window as shown below

Hello World: Your first C++ program



- Now press F5 and you will see the output window and it will ask you to press any key to close the program
- Press any key on the keyboard and this will terminate (close) the program

Hello World: Your first C++ program

- Next, let us add a code segment to print a message to the output window
- Modify the program as shown below and press F5 and see…
- This completes your first Hello World C++ Program

```
(Global Scope)
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World" << endl;
8      system("Pause");
9      return 0;
10 }
```

Few Terminologies

- As you see, a C++ program starts with an **#include** directive
- Include directives allow us to import C++ libraries that will allow our C++ program perform things
- For example **#include <iostream>** allows our program to perform printing to the output window
- Also in C++, **libraries are packaged together inside namespaces** and therefore whenever we include a library that is found inside a namespace, we need to tell our program to **use** that namespace
- For example, the **iostream** library is found inside a namespace named **std** and therefore we use that namespace as shown by **using namespace std;**
- Finally, in C++ printing a message to the output screen is achieved by **cout** command as shown above

C++ Main Program Block

- As shown every C++ main program starts with **int main()**
- It is followed by opening curly bracket **{**
- It also has a corresponding closing curly bracket **}**
- The part of C++ main program between **{** and **}** is known as the **block** of the main program
- The C++ code that will form the main program will be placed inside the main program block

Syntax of C++ Programs

- In order to write a good essay in English language, we follow the grammar of the English language
- Similarly in order to write a correct C++ program, we must follow the grammar of C++ programming language!!!
- The grammar of a programming language is known as the **syntax** of the programming language
- Therefore in order to write a correct C++ program, we must follow the syntax of C++ programming language
- After writing a C++ program and press F5, C++ will first check our program for any **syntax error**
- If there is any syntax error, the program will not run; instead it will show the error messages below the middle pane of the IDE

Compiling and Linking C++ Programs

- The C++ program that we type (edit) in the MSVC++ IDE is called the **source code** of our program
- When we press F5, C++ tests our source code for any syntax errors
- The part of C++ programming language that checks our source code for any syntax error is known as the **C++ compiler**
- After the compiler asserts that there are no any syntax errors in our source code, C++ converts our program to an executable file (alternatively called an **application** or simply an app)
- The part of the C++ programming language that converts our source code to an application is called the **C++ linker**
- Thus when we press F5, C++ will first perform compilation of our source code using the compiler and then, if there are no any syntax errors, it will link our program in order to create an executable file (or an application) that runs on the computer
- Any syntax error is caught by the compiler; while any linking error (this is more advanced topic) is caught by the linker

C++ Statements

- Just like an English language paragraph is made up of sentences, C++ program is made up of **C++ statements**
- A C++ statement is one line of code that does something
- Every C++ statement is terminated by a **semicolon**
- For example **cout << "Hello World" << endl;** is a C++ statement that prints a message to the output window
- Thus any C++ program is made up of one or more C++ statements separated by semicolons
- **When you execute a C++ program, the statements of the program will be executed one after the other starting from the first statement all the way down to the last statement**
- The last statement of our C++ programs is **return 0;** which informs the operating system our program has finished execution

cout statement

- **Syntax**

  cout [<< "SOME MESSAGE"] [<< endl];

  cout [<< 'ONE CHACTER'] [<< endl];
- Square bracket means, what is inside a square bracket is optional
- The message to be printed, if there is any, must be placed between **double quotes** if it contains more than one characters; and it may be placed between single quotes if it contains **only one** character
- **endl** stands for **end of line** (same as pressing the Enter key). Thus
  - **cout << "Hi" << endl;** prints the message **Hi** and moves the cursor to the next line.
  - **cout << "Hi";** prints the message **Hi** and keeps the cursor on the same line
  - **cout << endl;** prints no message and moves the cursor to the next line
  - **cout;** prints no message and keeps the cursor in the same line
  - **cout << 'h';** prints the character **h** and keeps the cursor on the same line
  - **cout << 'h' << endl;** prints the character **h** and moves the cursor to the next line

Practice Exercises

- Now, create a new C++ project and write a C++ program that prints the following message on to the output window

  Fraser International College

  CMPT 130 Course

  Instructor Yonas

- Your output must be in the same format as shown here

The Memory Unit of a Computer

- The memory unit of a computer is simply millions or billions of transistors
- As such, memory unit of a computer can store binary information
- The memory unit is organized as a long series of memory boxes as shown here

Variables in C++
Declaration and Initialization

- In C++, variables are names we assign to memory locations
- We can then assign a value to a variable which will be stored in the memory location of the variable
- That is, instead of remembering memory locations, we give the memory locations names we choose (like English names see below)
- Variable names are formed by using English alphabets, digits or the underscore character
- Variable names must begin with an alphabet or underscore
- Variable names are case sensitive
- Some examples of valid variable names are: **x**, **y**, **a1**, **b5**, **age**, **studentId**, **student_id**, **myAge**, **my_age**, **numberOfStudents**, **_size**, **solution**, **bit**, **number**, **num**, **_num1**, **num2**, **number1**,…
- Some examples of invalid variable names include **my-age**, **x+y**, **4x**, **age$2**, **number of students**,…

Variables in C++
Declaration and Initialization

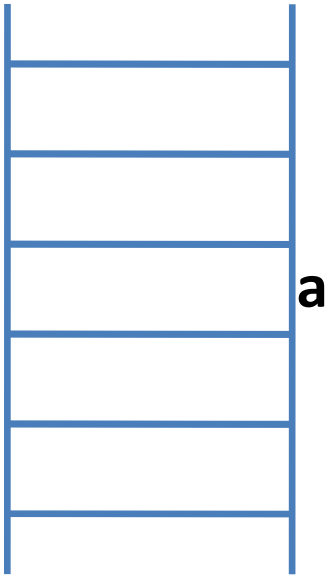
- Variable names must be different from C++ keywords
- Keywords have predefined meanings and are reserved words for C++ language
- Some C++ keywords: auto, break, case, float, if, goto, signed, default, continue, char, int, void, typedef, volatile, while, do, extern, asm, protected, class, new, throw, try, virtual,....

C++ Data Types

- In order to use a variable in C++ program, **it must be declared before its use**
- Variable declaration means assigning data type for the variable
- Example

```
int a;
```

- In this example **a** is a variable that can hold an integer value
- So here, **a** is nothing but a name we assigned to a memory location as shown here
- Which memory location? We don't know!
- C++ searches for a free memory location, grabs it, and gives it to us to use



C++ Data Types

- C++ supports the following basic data types

Data Type	Data Size (Byte)	Minimum Value	Maximum Value
char	1	-128	127
short	2	-32,768	32,767
int	4	-2,147,483,648	2,147,483,647
long	4	-2,147,483,648	2,147,483,647
float	4	-3.4 E +38	3.4 E +38
double	8	-1.7 E +308	1.7 E +308
long double	10	-3.4 E +4932	3.4 E +4932
unsigned char	1	0	255
unsigned short	2	0	65,535
unsigned int	4	0	4,294,967,295
unsigned long	4	0	4,294,967,295
bool	1	true or false	true or false

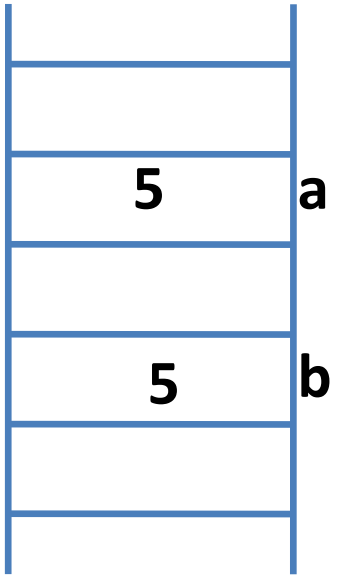
The Assignment Operator

- In C++ a variable is assigned (given) a value by using the assignment operator
- **Syntax**

  variable = value;
- The assignment operator has two operands (one on the left and the other on the right)
- The left hand side operand must be a variable name
- The right hand side can be a value or a variable that has already been assigned a value
- **Example**

```
int a;
a = 5;
int b;
b = a;
```

- Now, the variable **a** is assigned a value 5
- The variable **b** is assigned **a copy of the value** of **a** which is 5



Printing the Values of Variables

- In order to print the value of a variable, we use cout command
- For Example:

```
int a;
a = 5;
cout << a << endl;
```

- Note that there is no quotation mark because if you put quotation mark, you will print the message containing the symbol **a** but not the value of the variable **a**
- In order to print a message and the value of a variable, just use << to separate them
- For Example

```
int a;
a = 5;
cout << "The value of a is " << a << endl;
```

- In order to print the values of more than one variables, some messages and some computations on the variables, just separate them with <<
- For Example

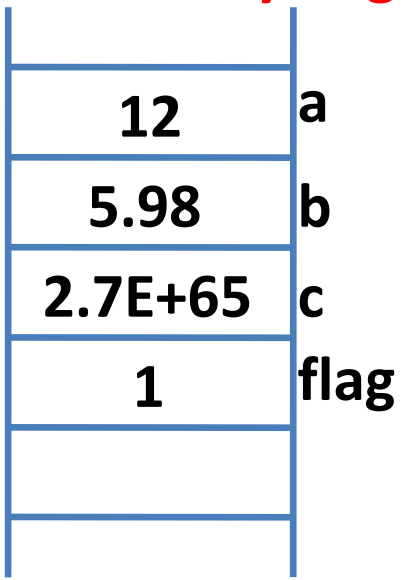
```
int a;
a = 5;
int b;
b = 7;
cout << "a is " << a << " and b is " << b << " and  a + b is " << a + b << endl;
```

Variables and Data Types: Example

```
int main()
{
    int a;
    float b;
    double c;
    bool flag;

    a = 12;
    b = 5.98;
    c = 2.7E+65;
    flag = true;

    cout << "Value of a = " << a << endl;
    cout << "Value of b = " << b << endl;
    cout << "Value of c = " << c << endl;
    cout << "Value of flag = " << flag << endl;

    system("pause");
    return 0;
}
```

Remark

Although flag is declared as boolean variable and assigned the value true, when you print flag, what do you get?



Variable Declaration, Initialization and Definition

- In C++, a variable may be declared and subsequently initialized
- Example

```
int a;          //variable declaration
⋮
a = 5;          //variable initialization
```

- Alternatively, a variable may be initialized during its declaration which is known as variable definition
- Example:

```
int a = 5;      //variable definition
```

Character Data Type

- A C++ **char** variable uses one byte of memory and can store signed integers from -128 to 127 using two's complement representation
- But although the actual value stored inside a C++ char variable is a signed integer represented in two's complement; when you print a C++ char variable using **cout** command, you will not see a signed integer
- Instead, the **cout** command will treat the char variable as if it is in **unsigned binary representation**, computes its unsigned decimal value **(which will always be between 0 and 255)** and finally prints the symbol (**character**) whose **Ascii code** is the unsigned decimal number computed
- Moreover, a C++ char can be assigned a character value such as alphabet, digit, or any special symbol such as ; , $, #, @, !, *,... from the keyboard
- In this case, the Ascii code of the character value will be stored in the char variable
- Example

```
char ch = 'f';
```

- Now, ch is assigned the integer value 102 which is ascii code of 'f'
- Note that in C++ characters are enclosed in single quotes

Character Data Type

- What if we want **cout** to print the actual signed integer, represented in two's complement representation, from a char data type variable?
- One way to achieve this is to tell **cout**, the variable under consideration is a numeric type variable that supports the unary operator **+**

Multiple Variables Declaration and Initializations

- C++ allows to declare and/or initialize multiple variables in one statement. Example

  int a, b = 6, c, d, e, f = 9;
- Multiple variables declared in the same statement must all be of the same data type
- C++ allows to initialize some of them and leave others uninitialized
- If a variable is not initialized, then it is undefined!
- Once variables are declared we can assign multiple variables with the same value in one statement as follows

  a = c = d = e = 5;
- Alternatively, we may assign them using a value of a variable that has been already initialized as follows

  a = c = d = e = b;
- In this case all a, c, d and e will get value of b which is 6.

Data Assignment Rules

- As a general rule, avoid putting a value of some data type in a variable of different data type!
- Assigning a float or double value, into an integer variable will truncate the data and put only the integer part to the variable. Example

  int x = 2.76;

  will put only the value 2 in the variable **x**.
- Putting an int value in a float or double variable is generally safe. Example

  float b = x;

  where x is as defined above will put 2.0 to the float variable b.

Constant Variables

- Sometimes we may need some variables to be initialized with some value and never want to change the value throughout the program
- In this case, the variable may be declared as constant
- Example

```
const int SIZE = 24;
```

- Constant variables are also called **named constants**
- Named constants can be used anywhere in our program to mean the value that is assigned to them
- It is customary to use all capital letters variable name for constant variables (although it is not a must)

C++ Arithmetic Binary Operators

- C++ supports the following arithmetic operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo (remainder) *Defined for integer operands only

- Always use brackets to make sure computations are performed in the order you would like them
- Remark: C++ does not have exponent operator!

Arithmetic Expressions

- C++ statements made up of arithmetic operators are known as arithmetic expressions. Example

```
int main()
{
    int a = 5, b = 3;

    cout << a << " + " << b << " = " << a + b << endl;
    cout << a << " - " << b << " = " << a - b << endl;
    cout << a << " * " << b << " = " << a * b << endl;
    cout << a << " / " << b << " = " << a / b << endl;
    cout << a << " % " << b << " = " << a % b << endl;

    system("pause");
    return 0;
}
```

Precision of Arithmetic Operations

- As can be seen in the previous program, sometimes C++ arithmetic expressions may not evaluate to what we would expect
- In the previous program **a / b** is evaluated to **1**; and not to **1.67**
- This is because the two operands, that is the variables **a** and **b** are integers. Therefore the division operation is done in integer domain
- In C++, the highest precision operand determines the result of an operation. Example

```
float a = 5.0;
int b = 3;
cout << a/b << endl;
```

←←←← **PRINTS 1.67**

would print **1.67** because the highest precision operand which is the variable a is float and therefore the operation is performed in float.

Order of Precedence of Binary Arithmetic Operators

- Given the following arithmetic expression, what would be the result?

7 - -6 + 4 * 5 / 6 % (4 + 3) / 2

- C++ has the following order of precedence shown in the following table

Operator	Order of Precedence
Bracket	Highest (Performed first)
* / %	
+ -	Lowest (Performed last)

- Ope are exe
- Therefore the above expression is evaluated to integer 14.

Type Casting

- Sometimes, we might need to convert the data type of the values of certain variables during an arithmetic expression
- Suppose, we have two **int** variables but we would like to print the division of the variables performed in floating data type; then we would convert the data type for the operation
- Example

```
int a = 5, b = 3
float c = static_cast<float>(a) / b
```

- Now, **c** will be assigned 5.0/3 = 1.67
- We say the variable **a** is casted from **int** to **float**

Type Casting

- When a variable is casted for an operation, then the variable still remains unchanged. Only a temporary value of the casted type is returned for the computation purposes
- In the previous example the variable **a** is still an int variable with a value 5 even after the operation (see example program below)

Type Casting

```
int main()
{
    int a = 5, b = 3;

    cout << "static_cast<float>(a) / b = " << static_cast<float>(a) / b << endl;
    cout << "static_cast<float>(a) / static_cast<float>(b) = " <<
        static_cast<float>(a) / static_cast<float>(b) << endl;
    cout << "static_cast<float>(a / b) = " << static_cast<float>(a / b) << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    system("pause");
    return 0;
}
```

- In this case, the first and second expressions both give 1.67 that is to say 5.0/3 = 5.0/3.0 = 1.67
- While the third line performs the division 5/3 to get the integer 1 and then this integer result is casted to float 1.0
- The variables a and b still remain int data types after the computation and their values unchanged!

Unary Arithmetic Operators

- In addition to the binary operators, C++ supports unary (that is only one operand) operators for integer, short, and long variables
- These operators are **++** and **--**
- They increment and decrement the value of a variable by 1
- Example:

```
int a = 4, b = 7, c = 10, d = 1;
cout << a++ << endl; ←prints 4 and makes a to have the value 5
cout << ++b << endl; ←prints 8 and makes b to have the value 8
cout << c-- << endl;  ← prints 10 and makes c to have the value 9
cout << --d << endl;  ← prints 0 and makes d to have the value 0
cout  << a << "  " << b << "  " << c << "   " << d << endl
```

Unary Arithmetic Operators

- When inside binary operators, the unary operators will have different effects depending on which side (left or right) of the variable they are placed
- Example:

```
int a = 7, b = 5, c = 1, d = 9;
cout << ++a + b << endl;
```

This first increments a to 8 then does the addition giving the output 13

```
cout << c-- + d << endl;
```

This first does the addition 1+9 printing 10 then decrements the value of c to 0

```
cout << a << endl;
```
←←← prints 8

```
cout << c << endl;
```
←←← prints 0

Unary Arithmetic Operators

- When used inside binary operations, the unary operators are as if there are two statements. Example suppose we have

  int a = 6, b = 7, c = 4, d = 9, e, f;
- Then the expression

  e = a + b++;

  is equivalent to

  e = a + b;

  b = b + 1;
- While the expression

  f = --c * d;

  is equivalent to

  c = c – 1;

  f = c * d;

Post Increment and Pre Increment

- The unary operators are also known as post increment, post decrement, pre increment or pre decrement based on which side (left or right) the unary operator is place
- **Post Increment**: x++

  uses the current value of x then increments it
- **Pre Increment**: ++x

  increments the x first and then uses the new value
- Similarly for the post-decrement and pre-decrement
- When the unary operators are alone in a statement then pre or post does not make any difference. Example

```
int a = 3, b = 8;
a++; ← increment a by 1
cout << a << endl; ← prints 4
--b; ← decrements b by 1
cout << b << endl; ← prints 7
```

Unary Arithmetic Operators

- Analyse the following C++ program and determine the output without writing the code in a computer. Instead trace the output of the program using pencil and paper

```
int main()
{
    int a = 5, b = 3, c = 8, d = 9;

    cout << "a=" << a << ", b=" << b << ", c=" << c << ", d=" << d << endl;
    cout << "a++ + b = " << a++ + b << endl;
    cout << "--c + d++ = " << --c + d++ << endl;
    cout << "a=" << a << ", b=" << b << ", c=" << c << ", d=" << d << endl;

    int e = 12, f = 5;
    e++;
    cout << e << endl;
    --f;
    cout << f << endl;

    system("pause");
    return 0;
}
```

cin: Reading Values from Keyboard

- In order to read values from keyboard, we use cin command
- **Syntax**

  cin >> variableName;

- For example, see the following program

```
#include <iostream>
using namespace std;
int main()
{
    int number1, number2, result;
    cout << "Please enter the first number: ";
    cin >> number1;
    cout << "Please enter the second number: ";
    cin >> number2;
    result = number1 * number2;
    cout << "The product of the two numbers entered is " << result << endl;
    return 0;
}
```

Cascading Input using cin

- Just like cout can output multiple tokens separated by the streaming operator <<
- Multiple inputs for multiple variables can be cascaded in a single cin
- Example

```
int a, b, c;

cout << "Enter three values ";
cin >> a >> b >> c;
```

- Now, if we enter 6 8 14 from the keyboard then a = 6, b = 8 and c = 14
- **Do not put commas to separate multiple inputs!**

C++ Comments

- C++ comments are description sentences we may want to put in our programs but they are NOT part of the program. They are there only to explain some things we may want to explain
- C++ provides two types of comments
- **Single line comments**

  Every line that starts with // is a comment. Such comments are single line comments
- **Multiple line comments**

  If a line starts with /* then all C++ statements below it until a closing */ are considered comments
- The following program demonstrates this

C++ Comments

```
int main()
{
    /* This program
    asks the user to input an integer and then
    prints the square of the integer.
    Everything in this section is comment
    */
    int num1;

    //Enter the first number
    cout << "Please enter the first number: ";
    cin >> num1;    //cin allows to input data
    //Now print the square of num1
    cout << "The square of " << num1 << " is " << num1*num1 << endl;

    system("pause");
    return 0;
}
```

C++ Program Styling

- While it is not a must rule, C++ developers use some conventions to make programs easily readable and understandable. These include
  - Put each statement on separate line
  - Put the curly braces on their own line
  - Use all upper case names for constant variables
  - Begin variable names with lower case
  - Variable names with two or more words should be capitalized. Example **int numberOfStudents**
  - Insert as much needed as comments as possible
  - Always remember C++ is case sensitive