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

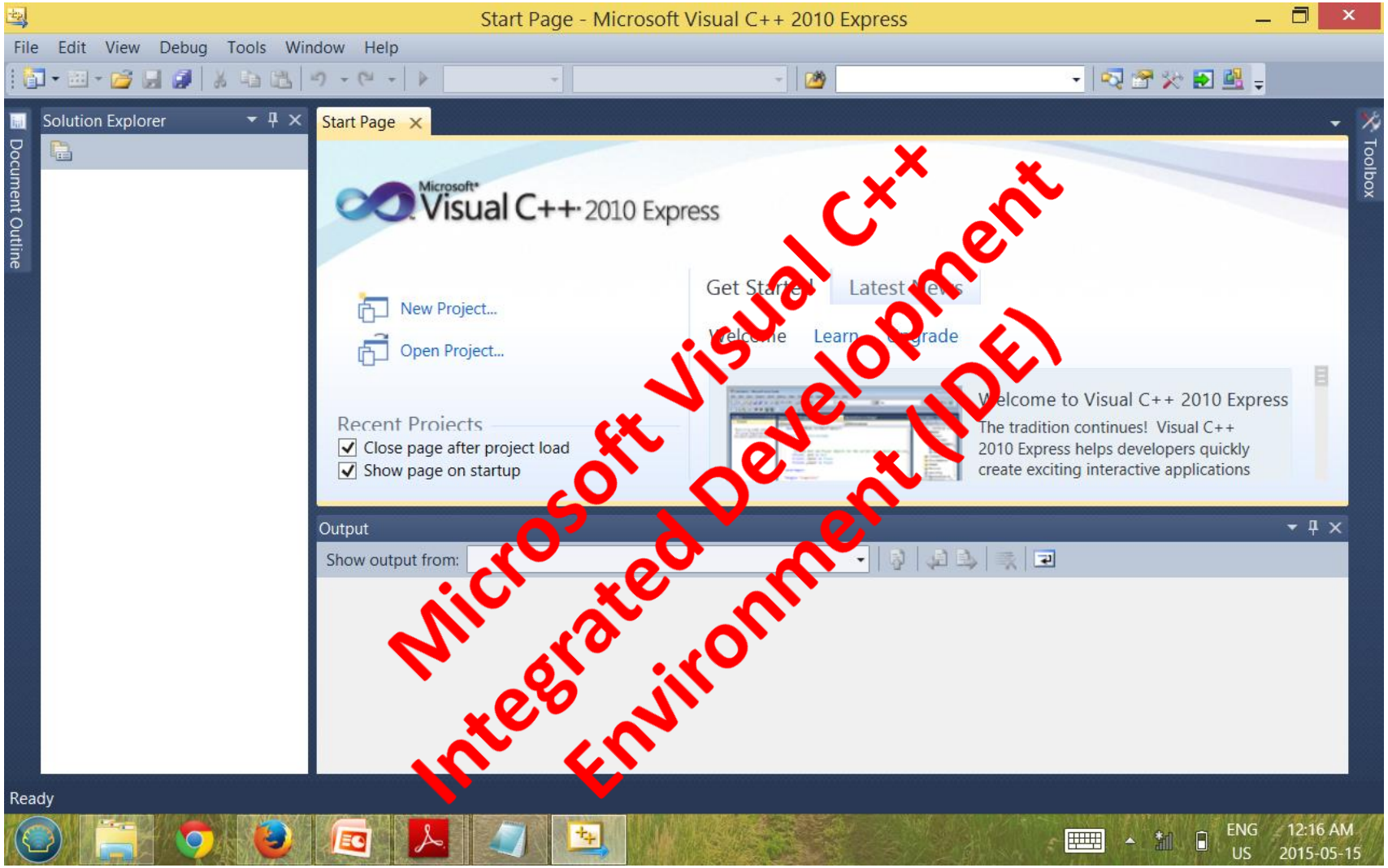
An IDE for C++ Programming

- 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 CMPT 130 course, we will use Microsoft Visual C++ 2010 Express
- The simplest way to get Microsoft Visual C++ 2010 Express in order to use it for your course work is to consult SFU IT administrators who manage course software servers

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++



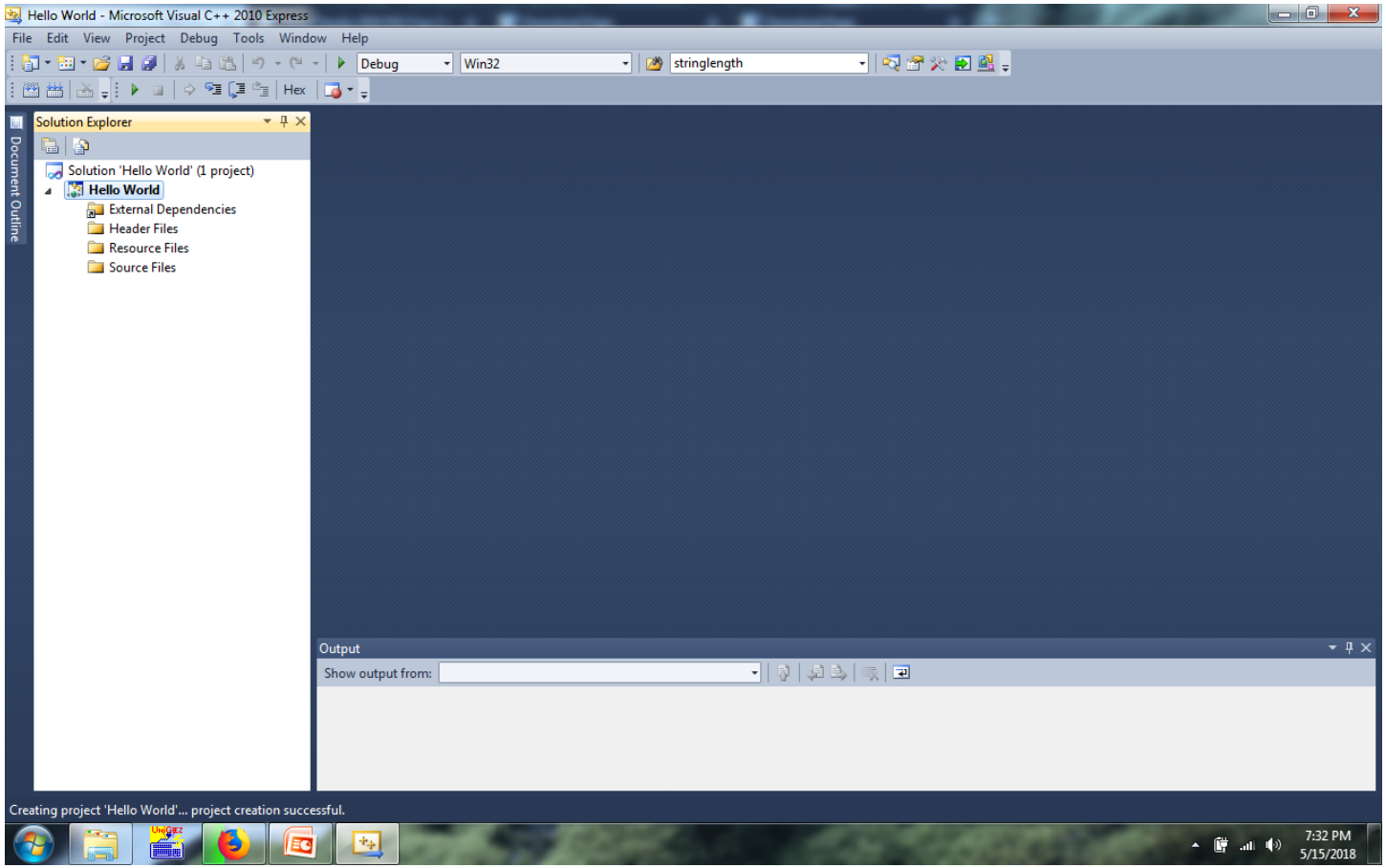
Microsoft Visual C++ Integrated Development Environment (IDE)

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 under Installed Templates select **Visual C++**
  - In the middle, select **Win32 Console Application**
  - Type a name for your project: Name it **MyFirstProgram**
  - Browse to a folder where you would like to save your project and click on **Select Folder**
  - Click **OK** and then click **Next**
  - Under Additional options click on **Empty project**
  - Finally click **Finish**

Hello World: Your first C++ program

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



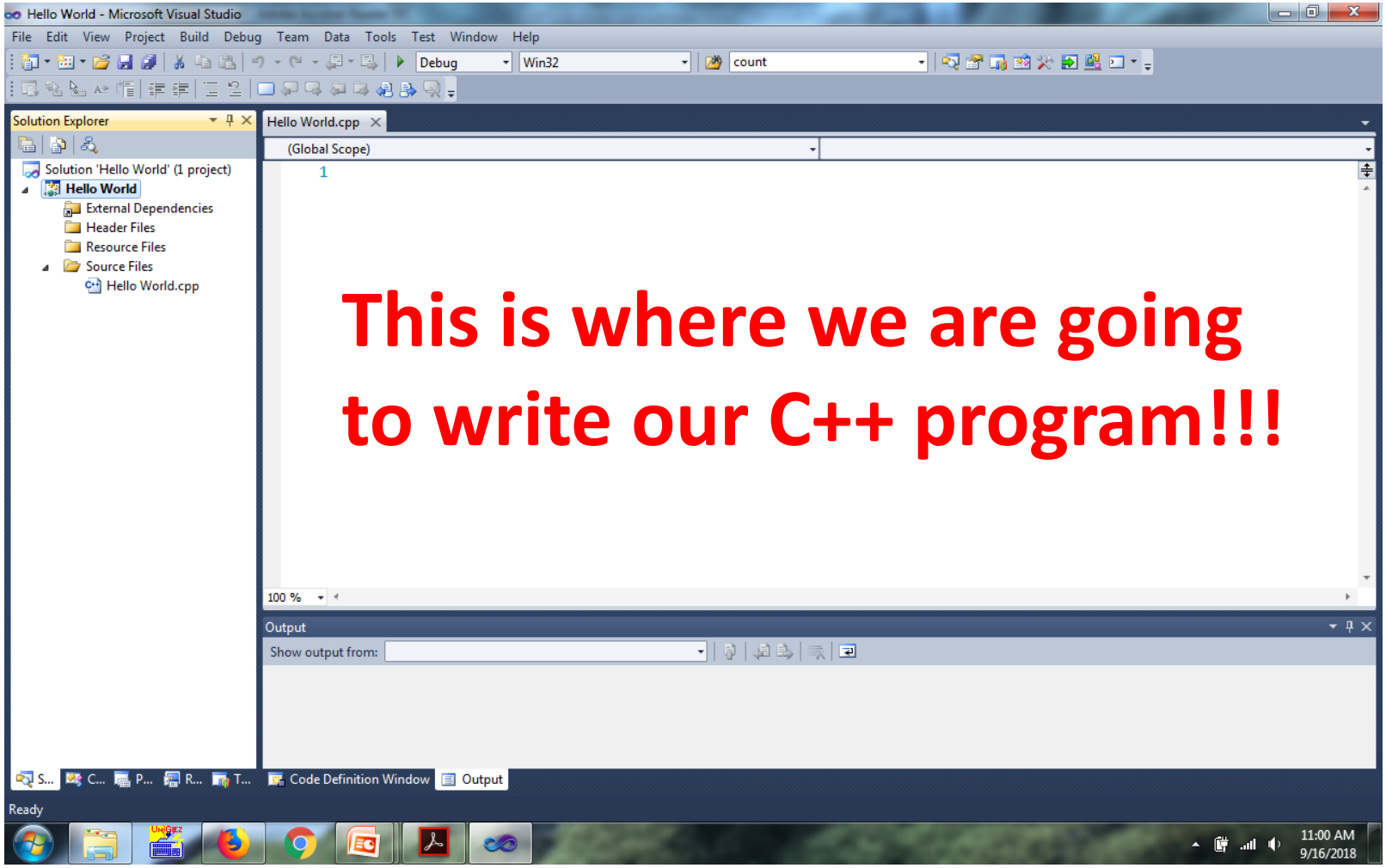
Hello World: Your first C++ program

- 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 **MyFirstProgram**
  - Click **Add**
- Now the middle pane will open an empty editor we just created shown below

Hello World: Your first C++ program

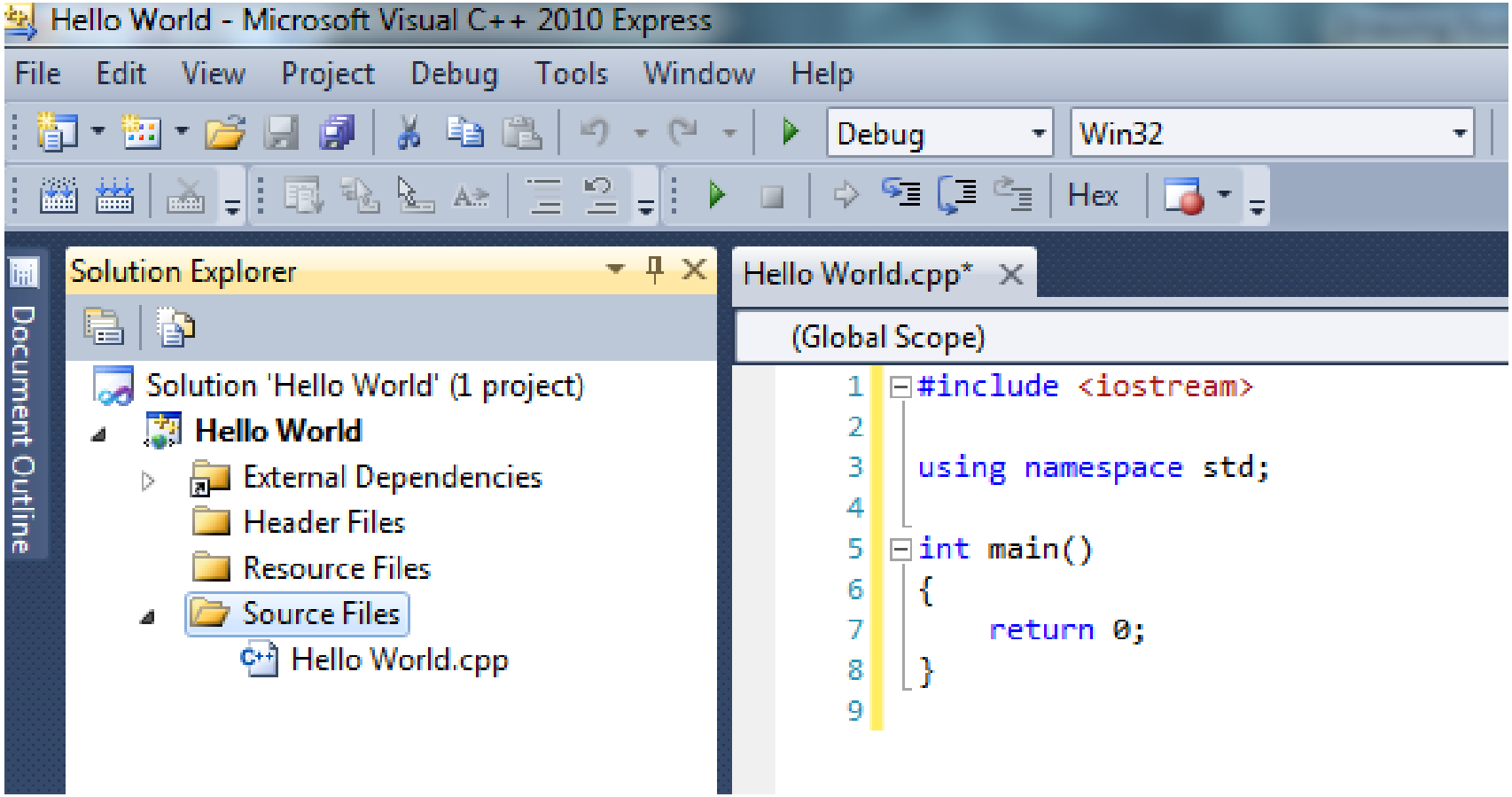


This is where we are going to write our C++ program!!!

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 press **F7** key on the keyboard
- You should get the message

  ===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
- Now press **F5** key on the keyboard in order to run (execute) the program
- A black screen will appear and close immediately
- This black screen is known as the **output console (window)** of the program
- In order to stop the output window from closing immediately, we will write a code that tells C++ to pause execution of the program before closing the output window

Hello World: Your first C++ program

(Global Scope)

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

- Now press **F7** to check the correctness of the program and once we get a succeeded message press **F5** to execute the program 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 F7 and then 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 some computations
- 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 in the program

C++ Main Program Block

- As shown in the program above, 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
- Thus the code that will form a C++ program will always 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 **F7**, C++ will first check our program for any **syntax errors**
- If there is any syntax error, it will be shown on the bottom pane 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 **F7**, 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**
- Finally we press **F5** in order to link our program to some libraries so that to create an executable file (alternatively called an **application** or simply an **app**)
- The part of the C++ programming language that links our program to libraries in order to create an executable file is called **C++ linker**
- Thus when we press **F5**, C++ will first link our program in order to create an executable file and then execute the executable file
- If any of the libraries required in order for our program to be converted to executable file are missing; then the C++ linker will report errors which we call **linking errors**
- Thus 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 some computation
- 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 a C++ program is made up of one or more C++ statements separated by semicolons
- **When you execute a C++ program, the statements in 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

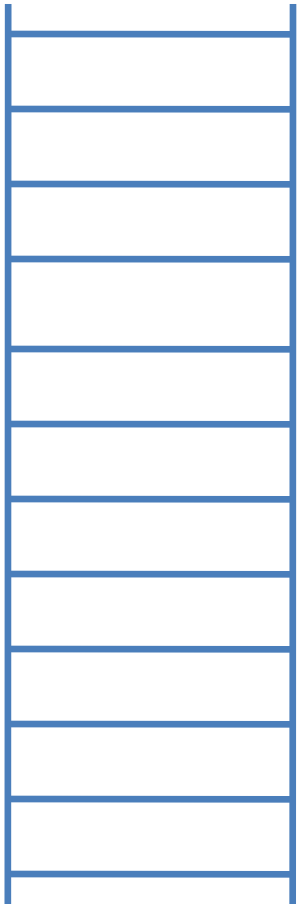
- **Syntax**

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

  cout [<< 'ONE CHARACTER'] [<< 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 or double 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 << '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

Storing and Processing Data in C++ Programs
The Memory Unit of a Computer

- In order process data in our C++ programs, we first need to store the data in the computer's memory unit
- 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
- Each box is one byte in size



Variables in C++
Declaration and Initialization

- In C++, a **variable** is a name we assign to a memory space in the computer's memory unit which allows us to refer to the same memory space again and again using the variable name
- For example, we can assign a value to the variable in order to store the value in the memory space the variable refers to and then use the value stored in the memory using the variable name we have chosen
- That is, in order to identify memory spaces we give them names that we can easily remember and then refer to the memory using the name
- 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

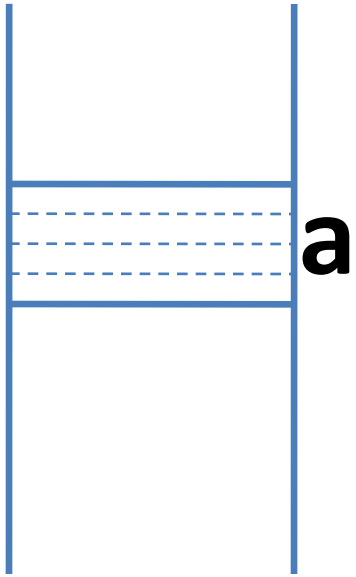
- Variable names must be different from C++ keywords
- Keywords are some names that have predefined meanings in C++ programming language and are reserved words for the language
- Thus we can not use any keyword as a variable name in our C++ programs
- Some C++ keywords: auto, break, case, float, if, goto, signed, default, continue, char, int, void, typedef, volatile, while, do, extern, asm, protected, class, new, throw, try, virtual,....

Variables in C++
Declaration and Initialization

- In order to use a variable in C++ program, **it must first be declared before its use**
- Variable declaration means assigning data type for the variable
- The data type of a variable specifies the number of Bytes used by the variable and the information representation used when storing data in the variable
- Example

  int a;

- Now **a** is a variable 4 bytes in size and can store an integer value in two's complement representation
- 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



Variables in C++
Declaration and Initialization

- C++ supports the following basic data types

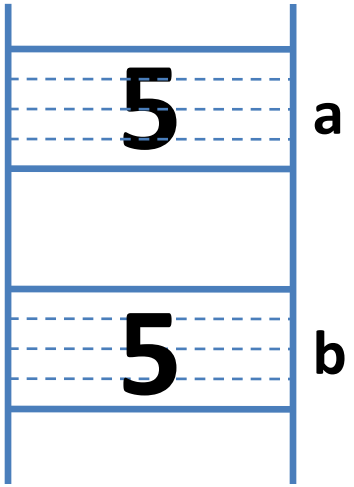
Data Type	Memory Size in Bytes	Information Representation	Minimum Value	Maximum Value
char	1	Two's complement	-128	127
short	2	Two's complement	-32,768	32,767
int	4	Two's complement	-2,147,483,648	2,147,483,647
long	4	Two's complement	-2,147,483,648	2,147,483,647
float	4	Not Discussed	-3.4 E +38	3.4 E +38
double	8	Not Discussed	-1.7 E +308	1.7 E +308
long double	10	Not Discussed	-3.4 E +4932	3.4 E +4932
unsigned char	1	Unsigned Binary	0	255
unsigned short	2	Unsigned Binary	0	65,535
unsigned int	4	Unsigned Binary	0	4,294,967,295
unsigned long	4	Unsigned Binary	0	4,294,967,295
bool	1	Not Discussed	true or false	true or false

The Assignment Operator

- In C++ a variable is assigned a value using the assignment operator
- **Syntax**

  variable = value;
- The assignment operator has two operands (one on the left hand side and the other on the right hand size)
- The left hand side operand must be a variable name
- The right hand side can be a literal value or a variable that has already been assigned a value in which case its value will be used for the assignment operation
- **Example**

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



- Now, the variable **a** is assigned the literal 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 the **cout** command
- For Example:

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

Output

```
5
```

- Note that the variable name is not placed inside single or double quotes because if we do that then the cout statement will interpret it as a message containing the symbol **a** but not the value of the variable **a**
- In order to print some message and the value of a variable, use << to separate them
- For Example

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

Output

```
The value of a is 5
```

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

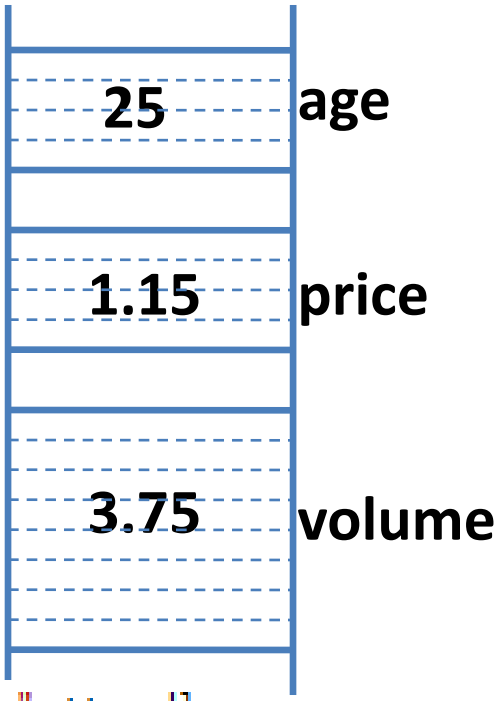
```
int a;
a = 5;
int b;
b = 7;
cout << "The value of a is " << a << " and that of b is " << b << endl;
```

Output

```
The value of a is 5 and that of b is 7
```

Variables and Data Types: Example

```
#include <iostream>

using namespace std;

int main()
{
    int age;
    float price;
    double volume;

    age = 25;
    price = 1.15;
    volume = 3.75;

    cout << "I am " << age << " years old." << endl;
    cout << "The price of orranges is $" << price << " per killogram." << endl;
    cout << "We are left with " << volume << " liters of gas." << endl;

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



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

  int b(5); --------- > variable definition
- Moreover several variables of the same data type can be declared or defined together in one statement
- Example:

  double x, y, z; -- > several variables declaration

  float a = 2.25, b(1.15), c = 1.5; -- > several variables definition

  int p1, p2 = 1.1, p3; -- > several variables declaration or definition

Variable Assignment Rules

- Generally speaking a variable should always be assigned a value that is the same data type as the variable itself
- But sometimes, we may wish to assign a variable a different data type value
- In such cases, either C++ will automatically adjust the assigned value or give syntax error if the value can not be adjusted
- Examples:

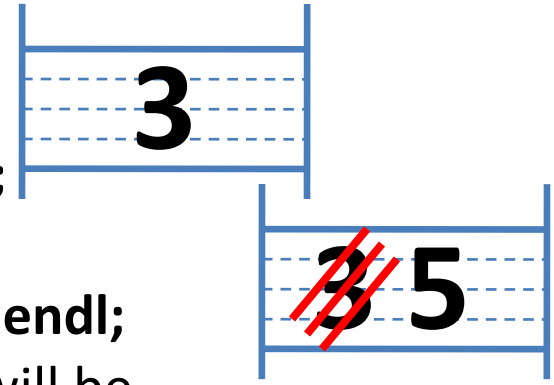
  int a = 3.5; ---- > the variable a is assigned the integer value 3. We say the value 3.5 is truncated to integer 3

  float b = 5; ---- > the variable b is assigned the value 5.0

Modifying the value of variables

- The value of a variable can be modified as many times as we wish
- The assignment operator is used to assign a new value to a variable
- Whenever a new value is assigned to a variable, then the old value is deleted from the memory and replaced with the new value
- Example

```
int x = 3;
cout << "The value of x is " << x << endl;
x = 5;
cout << "Now the value of x is " << x << endl;
```



- This output from this sample code fragment will be

```
The value of x is 3
Now the value of x is 5
```

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: Example

```
#include <iostream>
using namespace std;
int main()
{
    int a = 15, b = 6;
    float c = 2.4;

    int s = a + b;
    float d = c - b;
    float p = b * c;
    float q1 = a / c;
    int q2 = a / c;
    int q3 = a / b;
    float q4 = a / b;
    int r = a % b;

    cout << "The sum of " << a << " and " << b << " is " << s << endl;
    cout << "The difference between " << c << " and " << b << " is " << d << endl;
    cout << "The product of " << b << " and " << c << " is " << p << endl;
    cout << "Dividing " << a << " by " << c << " gives the quotient " << q1 << endl;
    cout << "Dividing " << a << " by " << c << " gives the quotient " << q2 << endl;
    cout << "Dividing " << a << " by " << b << " gives the quotient " << q3 << endl;
    cout << "Dividing " << a << " by " << b << " gives the quotient " << q4 << endl;
    cout << "Dividing " << a << " by " << b << " gives a remainder of " << r << 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 for example, when we divide the integer value 15 by the integer value 6 then the division operation is performed in integer domain and will give an integer result
- The highest precision operand of division operation will always determine the result of the division operation
- Here is a guide for division operation in C++

Division Operation Data Types	Result
int / int	int
int / float, float / int, float / float	float
int / double, double / int, double / double	double
float / double, double / float	double

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)

- Operators on the same row have the same order of precedence and are executed from left to right
- Therefore the above expression is evaluated to integer 14.

Reading User Input Values

- In order to read values from keyboard, we use **cin** command
- **<u>Syntax</u>**

  cin >> variableName;

- The cin command will pause execution of a program and will wait until the user enters value from the keyboard
- Once the user types a value and presses the Enter Key then the value will be assigned to the variable and execution of the program will proceed to the next statement in the program

Practical Examples: Example 1

- Write a C++ program that will ask the user to enter the length and width of a rectangle and then computes and prints the area and the perimeter of the rectangle.

```
#include <iostream>
using namespace std;
int main()
{
    float length, width, area, perimeter;
    cout << "Please enter the length ";
    cin >> length;
    cout << "Please enter the width ";
    cin >> width;

    area = length * width;
    perimeter = 2 * (length + width);

    cout << "Area is " << area << endl;
    cout << "Perimeter is " << perimeter << endl;

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

Reading User Input Values

- Multiple inputs for multiple variables can also be read in one **cin** statement as follows
- Example

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

- Now, if we enter **6 8 14** from the keyboard then the variable a will be assigned the value 6, the variable b will be assigned the value 8 and the variable c will be assigned the value 14
- Multiple inputs from the keyboard must be separated by one or more spaces or tabs but not any other separator
- The variables in the cin statement can also be of different data types; in which case the user input values must match the data types of the variables

Practical Examples: Example 2

- Write a C++ program that reads the coefficients of a quadratic equation

$$ax^2 + bx + c = 0$$

and then computes and prints the discriminant of the quadratic equation.

```
#include <iostream>
using namespace std;
int main()
{
    double a, b, c;
    cout << "Please enter the coefficients a, b, c of a quadratic equation ";
    cin >> a >> b >> c;

    double discriminant = b * b - 4 * a * c;

    cout << "The discriminant of the quadratic equation is " << discriminant << endl;

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

Practical Examples: Example 3

- Write a C++ program that asks the user to enter his/her birth day, month and year and then computes and prints the number of days since the user's birth until January 1, 2020. Assume the user input is an earlier date than January 1, 2020. For simplicity, assume there are 30 days in every month and there are 12 months in every year.

```
#include <iostream>
using namespace std;
int main()
{
    int d, m, y;
    cout << "Please enter the day, month and year of your birth date in that order ";
    cin >> d >> m >> y;

    int days = (2020 - y) * 360 + (1 - m) * 30 + (1 - d);

    cout << "There are " << days << " days since the day you were born until January 1, 2020." << endl;

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

Practical Examples: Example 4

- Write a C++ program that asks the user to enter the number of days since he/she was born until today and then computes and prints how old the user is in the format of years, months and days. For simplicity, assume there are 30 days in every month and there are 12 months in every year.

```
#include <iostream>
using namespace std;
int main()
{
    int days;
    cout << "Please enter the number of days since you were born until now ";
    cin >> days;

    int y = days / 360;
    days = days % 360;
    int m = days / 30;
    days = days % 30;
    int d = days;

    cout << "You are " << y << " years, " << m << " months, and " << d << " days old." << endl;

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

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
- Example

  char ch1 = -191;

- Now ch1 is assigned the integer value -191 whose two's complement binary representation is shown above
- Moreover, a C++ char can be assigned a character value such as an alphabet, a digit, or any special symbol placed in single quotes in which case the ASCII code of the character value will be stored in the char variable using two's complement representation
- Example

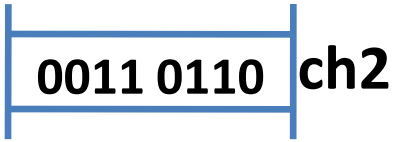
  char ch2 = 'A';

- Now ch2 is assigned the integer value 65 (ASCII code of 'A')

Character Data Type Input

- A char variable can also be assigned an input value from the keyboard using **cin** command
- In this case the user will have to put one character input from the keyboard (such as an alphabet, a digit or a special symbol) and then the ASCII code of the character input will be assigned to the char variable in two's complement representation
- Single quotes are not needed when typing an input from the keyboard
- Example

```
char ch;
cout << "Please enter a character ";
cin >> ch; 6 ↵
```



- In this case ch will be assigned the integer value 54 as shown above in its two's complement representation

Character Data Type Output

- Although the actual value stored inside a C++ char variable is a signed integer represented in two's complement; printing a C++ char variable using **cout** command will not print the signed integer stored in the variable
- Instead, the **cout** command will treat the char variable as if it is in **unsigned binary representation**, compute its unsigned decimal value **(which will always be between 0 and 255)** and finally print the symbol (**character**) whose **ASCII code** is the unsigned decimal number computed
- Example

```
char ch1 = -191, ch2 = 'A';
cout << "ch1 is " << ch1 << " and ch2 is " << ch2 << endl;
```

Output

```
ch1 is A and ch2 is A
```

Character Data Type Output

- 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 inform the **cout** command that the char variable under consideration is a numeric type information using the **+** or **-** unary operators
- For example, consider the following program and determine it's output

```
int main()
{
    int a = -191;
    char b = -191;

    cout << a << endl;      //This print -191
    cout << +a << endl;     //This print -191
    cout << -a << endl;     //This print  191

    cout << b << endl;      //This print A
    cout << +b << endl;     //This print 65
    cout << -b << endl;     //This print -65

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

Character Data Type Arithmetic

- Moreover when a char type variable is used in arithmetic operations then it behaves as numeric information and uses the integer value stored in the variable for the arithmetic operations
- Analyze the following program and determine its output

```
int main()
{
    char ch1 = -206, ch2 = 322, ch3;
    int num1 = -206, num2, num3;

    ch3 = ch1 + ch2;
    num2 = ch1 + ch2;
    num3 = num1 + ch2;  //Note that highest precision is int

    cout << ch1 << ", " << +ch1 << ", " << -ch1 << endl;
    cout << ch2 << ", " << +ch2 << ", " << -ch2 << endl;
    cout << ch3 << ", " << +ch3 << ", " << -ch3 << endl;
    cout << num1 << ", " << +num1 << ", " << -num1 << endl;
    cout << num2 << ", " << +num2 << ", " << -num2 << endl;
    cout << num3 << ", " << +num3 << ", " << -num3 << endl;

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

Constant Variables

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

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

- Once a variable is defined as a constant variable then attempting to modify the value of the variable will cause a syntax error
- 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 named constants (although it is not a must)

Type Casting

- Sometimes we may need to convert the data type of the values of certain variables during an arithmetic expression
- For example suppose that we have two **int** variables and we would like to perform division operation between the values of the variables 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 value of the variable **a** is casted from **int** to **float**

Type Casting

- When the value of a variable is casted for an operation then the variable still remains unchanged
- Instead the value of the variable is copied temporarily and this temporary value is casted as required and finally this casted temporary value is used for the computation purposes
- Thus the data type of the variable **a** is still an int and its value is still the integer 5 even after the operation. See also examples below.

Type Casting

```
int main()
{
    int a = 5, b = 3;
    cout << static_cast<float>(a) / b << endl;
    cout << a / static_cast<float>(b) << endl;
    cout << static_cast<float>(a) / static_cast<float>(b) << endl;
    cout << static_cast<float>(a / b) << endl;
    cout << a << endl;
    cout << b << endl;
    system("Pause");
    return 0;
}
```

- In this case, the first, second and third expressions all give 1.67 that is to say 5.0/3 = 5/3.0 = 5.0/3.0 = 1.67
- While the fourth 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 operators for integer, short, and long data type variables
- Unary operators have only one operand
- These operators are **++** and **--**
- They increment/decrement the value of their operand by 1
- These operators can be placed either on the left hand side or the right hand side of their operands as shown below

```
int a = 4, b = 7, c = 10, d = 1;
a++;        ←increment value of a by 1 (same as a = a + 1;)
++b;        ←increment value of b by 1 (same as b = b + 1;)
c--;        ←decrement value of c by 1 (same as c = c - 1;)
--d;        ←decrement value of d by 1 (same as d = d - 1;)
cout  << a << "  " << b << "  " << c << "   " << d << endl;
```

Unary Arithmetic Operators

- When unary operators are placed inside **cout** statement then they will be executed differently depending on which side of their operands they are placed as shown below

```
int a = 4, b = 7, c = 10, d = 1;
cout << a++ << endl; ←print value of a then increment it
cout << ++b << endl; ←increment value of b then print it
cout << c-- << endl; ←print value of c then decrement it
cout << --d << endl; ←decrement value of d then print it
cout  << a << "  " << b << "  " << c << "   " << d << endl;
```

Output

```
4
8
10
0
5  8  9  0
```

Unary Arithmetic Operators

- Similarly when unary operators are placed inside binary operators then they will be executed differently depending on which side of their operands they are placed as shown below

```
int a = 7, b = 5, c = 1, d = 9;
int sum1 = ++a + b;
```

This first increments the value of a to 8 and then computes the sum which is 13

```
int sum2 = c-- + d;
```

This first computes the sum which is 10 and then decrements the value of c to 0

Pre Increment and Post Increment

- The unary operators are also known as pre increment, pre decrement, post increment or post decrement based on which side (left or right) the unary operator is place
  - **<u>Pre Increment</u>**: ++x
    Increments the value of x first and then uses the new value
  - **<u>Post Increment</u>**: x++
    Uses the current value of x and then increments it
  - **<u>Pre Decrement</u>**: --x
    Decrements the value of x first and then uses the new value
  - **<u>Post Decrement</u>**: x--
    Uses the current value of x and then decrements it
- When these unary operators form a statement on their own then the pre and post operations will perform the same computation
- Example:

```
int a = 3, b = 3;
a++;
cout << a << endl;
++b;
cout << b << endl;
```

<u>Output</u>

```
4
4
```

Programming Errors

- As a programmer; quite often than not, you will be facing with four types of errors:
  - syntax errors: The code in a program does not adhere to the language rules (grammar)
  - run-time errors: Some part of the code in a program causes the program to crash
  - semantic errors: The output (result) from a program is logically wrong, and
  - linking errors: One or more library used in a program is missing.
- The following program demonstrates the first three types of errors.

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

int main()
{
        int a = 5, b = 0, c;
        cout << a << " - " << b << " is " << a_b << endl;       //syntax error
        cout << a << " - " << c << " is " << a-c << endl;       //run-time error
        cout << a << " - " << b << " is " << a*b << endl;       //semantic error
        system("Pause");
        return 0;
}
```

Comments in C++

- Comments are plain English language description sentences we may want to put in our programs but that 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 the lines below it until a closing */ are considered comments
- Comments are automatically ignored by the compiler and linker. See the example below.

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 named constants
  - 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