

C++ Basics

In this week

- More on information representation
 - Two's Complement Binary
- 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
- C++ Primitive Data Types
- **Variables**: declaration, initialization, definition
- Binary Operators: arithmetic expression
- Keyboard/Console Input and Output

Two's Complement Binary Representation

- How can we represent signed decimal numbers in binary such that
 - The system represents zero as a unique bit pattern
 - Arithmetic on the system follows a standard arithmetic procedure with no additional task to handle signs
 - The result of any arithmetic will have the correct sign
- Solution
 - We use **Two's Complement Representation!**

Two's Complement Binary Representation

- An elegant way to avoid the problems of signs and ambiguity in representing zero.
- All modern computers represent numbers with two's complement notation
- To find the two's complement of a given binary number, first flip all the digits and then add 1 to the result
- But how do we perform addition in binary number system?

Two's Complement Binary Representation

- Binary addition (single digit)

$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$
---	---	---	--

- Binary addition (bit patterns)

$\begin{array}{r} 1101 \\ + 1011 \\ \hline 11000 \end{array}$	$\begin{array}{r} 111101 \\ + 101 \\ \hline 1000010 \end{array}$
---	--

- Note: $1 + 1 = 0$ with a carry 1

Two's Complement Binary Representation

- Now, we can easily compute the two's complement of binary numbers
 - **Example:** What is the two's complement of **1011**?
 - **Solution**
 - Step 1. Flip all bits to get **0100**
 - Step 2. Add 1 to the result to get **0100 + 1 = 0101**.
 - Therefore, the two's complement of **1011** is **0101**.
- **Remark:** The two's complement of a binary number must be expressed in the same bit pattern in order to get a correct interpretation of the result. See below.

Two's Complement Binary Representation

- **Example:** Given the nibble **0000**, find its two complement.
 - **Solution**
 - Step 1. Flip all bits to get **1111**
 - Step 2. Add 1 to the result to get **1111 + 1 = 10000**.
 - **Remark:** The left most carry bit is discarded because it goes out of the capacity of a nibble. Therefore, the two's complement of **0000** is **0000**.
- **Example:** Given the nibble **1111**, find its two complement.
 - **Solution**
 - Step 1. Flip all bits to get **0000**
 - Step 2. Add 1 to the result to get **0000 + 1 = 0001**.
 - Therefore, the two's complement of **1111** is **0001**

Two's Complement Binary Representation

- How do we represent signed integers in two's complement?
- Solution
 - **Positive Decimal Numbers and Zero**

Step 1. Their two's complement representation is the same as their unsigned binary representation
 - **Negative Decimal Numbers**

Step 1. Find the unsigned binary representation of the decimal number without the negative sign

Step 2. Flip all the bits in Step 1

Step 3. Add 1 to the result in Step 2

Two's Complement Binary Representation

- Find the binary representation of 53 in two's complement representation as a byte
 - Solution
 - The unsigned binary representation of 53 as a byte is **00110101**. This is also its two's complement representation.
- Find the binary representation of -53 in two's complement as a byte
 - Solution
 - **Step 1.** Find binary representation of 53. It is **00110101**
 - **Step2.** Flip all bits to get **11001010**
 - **Step 3.** Add 1 to the result to get **11001011**
- What about -103 in two's complement as a byte pattern?
 - Solution: Steps 1 through 3 will respectively give the results: **01100111**, **10011000**, and **10011001**. Therefore the answer is **10011001**

Two's Complement Binary Representation

- Add 53 with -53 in two's complement using a byte.
 - Solution:

$$\begin{array}{rcl} 53 & \text{is} & \mathbf{00110101} \\ + -53 & \text{is} & \mathbf{11001011} \\ \hline 0 & & \mathbf{100000000} \end{array}$$

- Once again, the left most carry bit will be **discarded** as it will go **out of the capacity of a byte**. Therefore the answer will be **zero** as expected!

Two's Complement Binary Representation

- Now, let us see which **signed** integers can be represented by a given bit pattern
- For this purpose, let us consider a nibble

Decimal	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111

Decimal	-1	-2	-3	-4	-5	-6	-7	-8
Binary	1111	1110	1101	1100	1011	1010	1001	1000

- Notice that the bit pattern **1000** could have been used for **+8** or **-8** equally right. However, computer scientists decided to assign it **-8** for reasons that will be evident soon

Two's Complement Binary Representation

1. The two's complement of **positive** decimal numbers and **zero** always start with a **0** bit. While that of **negative** decimal numbers start with a **1** bit
2. A nibble can represent the signed integers -8 to +7 in two's complement, as shown above. Similarly, a byte can represent signed integers -128 to +127 in two's complement
3. Generally, an **n**-bit pattern can represent the signed integers $-(2^{n-1})$ to $+(2^{n-1}-1)$

Two's Complement Binary Representation

- In order to convert a given binary number in two's complement to decimal, follow these steps:
 - If it starts with a **0**, then it is positive or zero integer.
 1. Convert it to decimal by expansion with powers of 2
 2. The resulting decimal is the required answer
 - Else if it starts with a **1**, then it is negative integer.
 1. Find the two's complement of the given binary number
 2. Convert it to decimal by expansion with powers of 2
 3. The negative of the resulting decimal is the required answer

Two's Complement Binary Representation

- Find the integers represented by the following binary numbers represented in two's complement: **010101**, **11101101**, and **10000000**.
 - **010101** is positive. Expanding it results 21. Therefore the answer is 21.
 - **11101101** is negative. Its two complement is **00010011**. This binary corresponds to the decimal 19. Therefore the answer is -19.
 - **10000000** is negative. Its two complement is **10000000**. This binary corresponds to the decimal 128. Therefore the answer is -128

Two's Complement Binary Representation

- Consider the decimal number 5 in Byte pattern represented in two's complement
 - Its binary representation is **00000101**
 - Its two's complement is **11111011** which is -5
- Now consider the decimal number -5 in byte pattern represented in two's complement
 - Its binary representation is **11111011**
 - Its two's complement is **00000101** which is 5
- Thus, if the two's complement representation of an integer x is y then the representation for $-x$ is given by the two's complement of y

Two's Complement Binary Representation

- Let us perform the operations 3+-5, -3+5 and -3+5 in two's complement in Nibble and Byte forms:

$\begin{array}{r} 3 \quad 0011 \\ + -5 \quad 1011 \\ \hline -2 \quad 1110 \rightarrow -2 \end{array}$	$\begin{array}{r} -3 \quad 1101 \\ + 5 \quad 0101 \\ \hline 2 \quad \textcolor{red}{1}0010 \rightarrow 2 \end{array}$	$\begin{array}{r} -3 \quad 11111101 \\ + 5 \quad 00000101 \\ \hline 2 \quad \textcolor{red}{1}00000010 \rightarrow 2 \end{array}$
---	---	---

- The discarded carry bits are shown in red.
- When a carry over goes beyond the capacity of the bit pattern under consideration, then the operation is said to give rise to an **overflow**

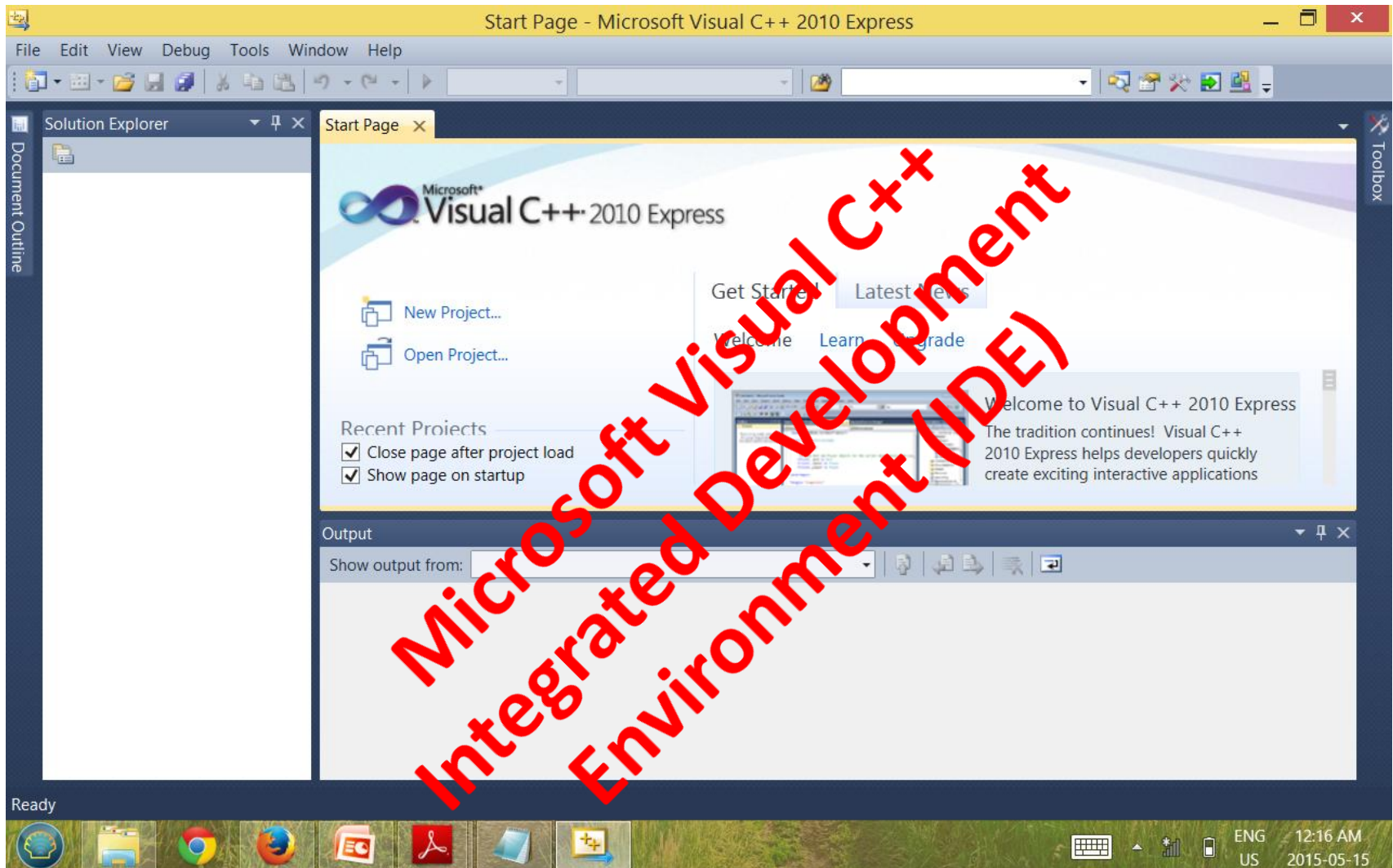
Introduction to C++ programming

- 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 125 course, we will use **Microsoft Visual C++ 2010 Express**
- This software is already installed for us to use in the FIC computer labs
- You may also copy the installation file of the software from FIC computer labs for your use on your laptop (See the course Moodle page for details)

Starting Microsoft Visual C++

- In order to start Microsoft Visual C++, click on
 - Start Button
 - All Programs
 - 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++

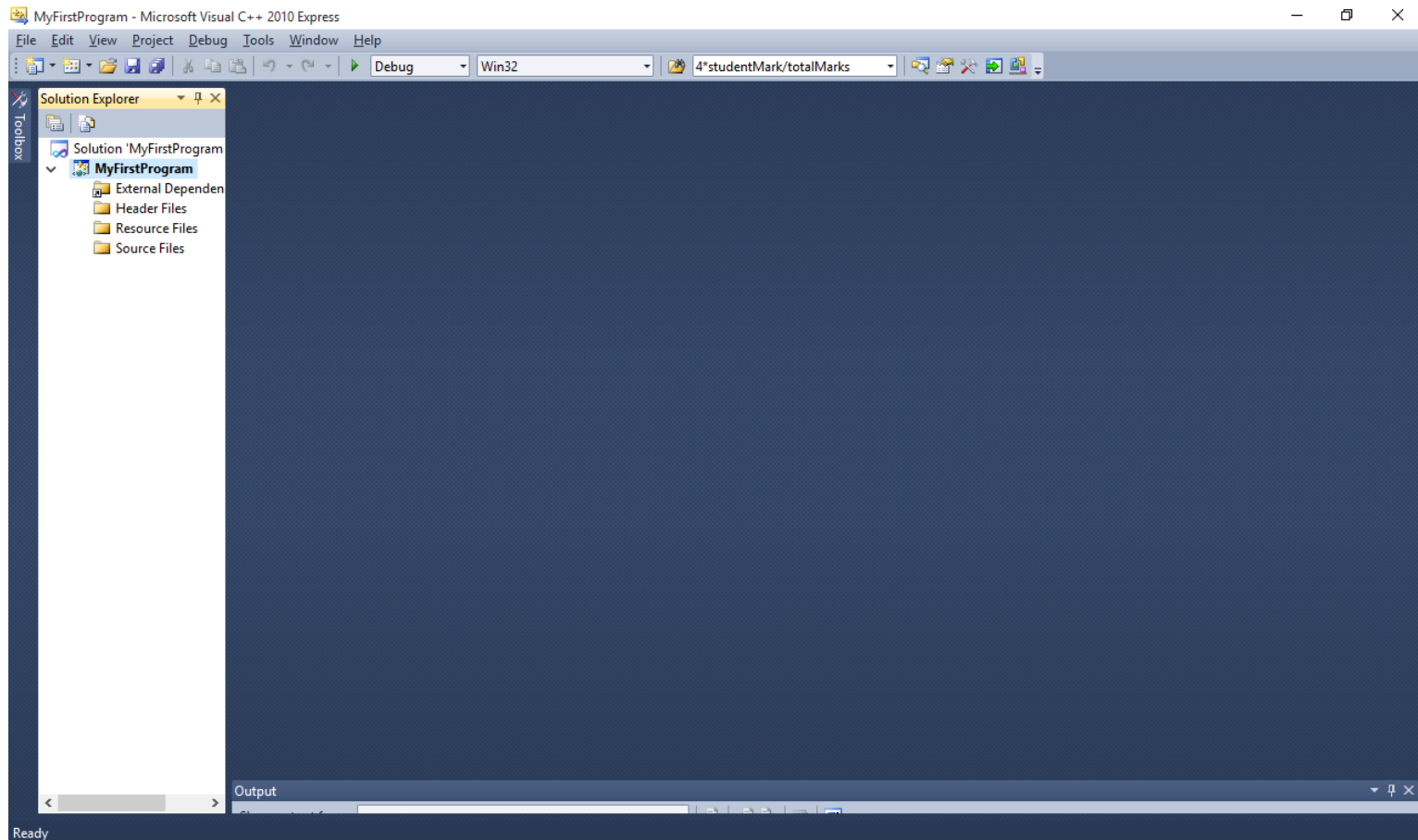


Hello World: Your First C++ Program

- In order to create your first C++ program click on the **New Project** short cut shown on the window
- A new project dialog box will be displayed. Perform the following steps carefully
 - On the left side under **Installed Templates** select **Visual C++**
 - In the middle section, select **Empty Project**
 - For the **Name**, type **MyFirstProgram**
 - For the **Location**, click on the **Browse** button and select the folder in your computer where you would like to save your project
 - Click on the **OK** button

Hello World: Your first C++ program

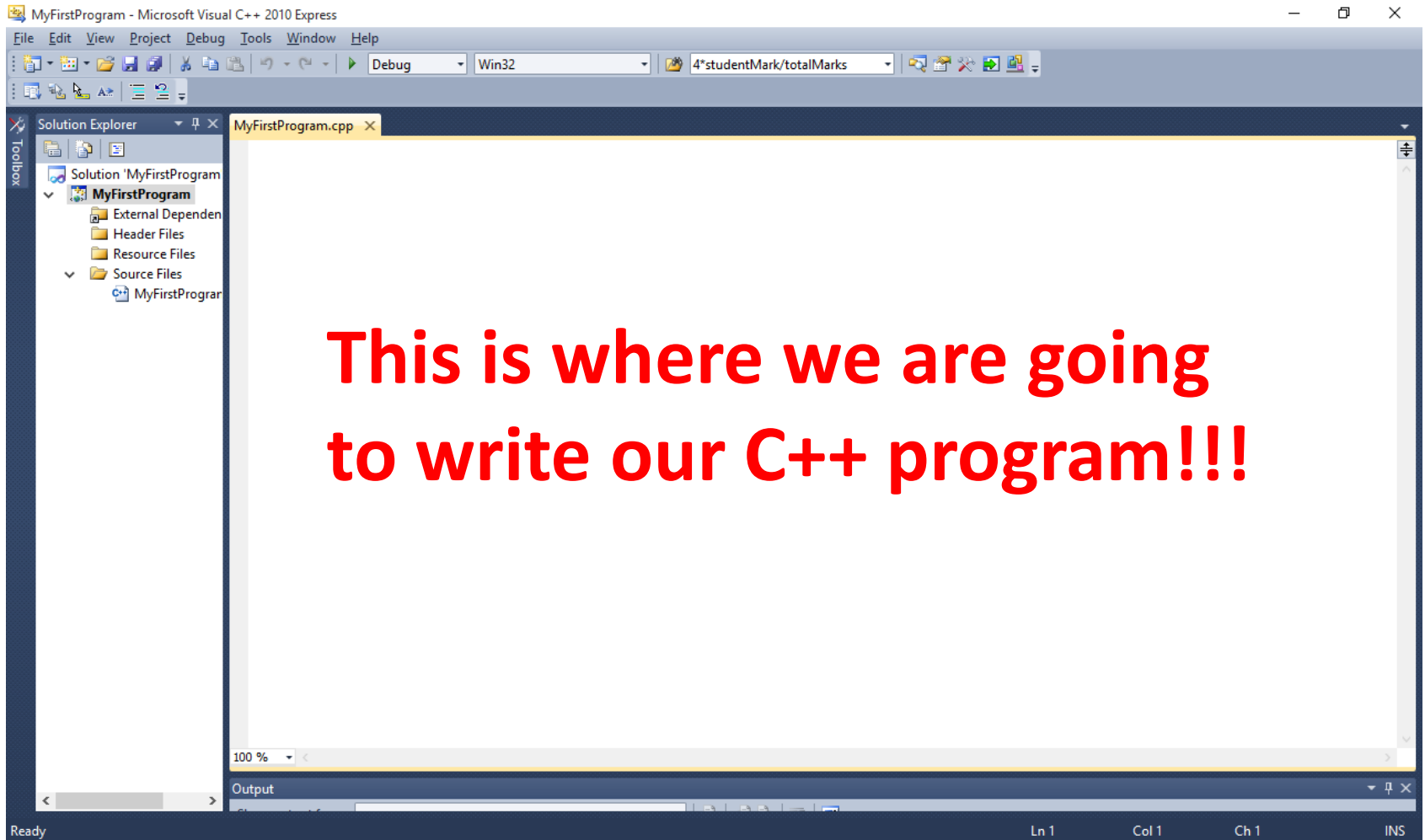
- Now MS VC++ will show the start page of your project similar to the window shown below



Hello World: Your first C++ program

- Now, let us create a new C++ program inside our project
 - Click on **Project** from the menu bar
 - *If you don't see the **Project** menu on the menu bar, then click on **View** on the menu bar, and then click on **Other Windows** and finally click on **Solution Explorer**. You will now see the **Project** menu on the menu bar.*
 - Click on **Add New Item**
- A new item dialog box will be displayed. Perform the following steps carefully
 - On the left side, select **Visual C++**
 - In the middle section, select **C++ File (.cpp)**
 - For the **Name**, type **MyFirstProgram**
 - For the **Location**, it is already set and therefore don't modify it
 - Click on the **Add** button
- Now the middle pane will open an empty editor where we will be typing our program as shown below

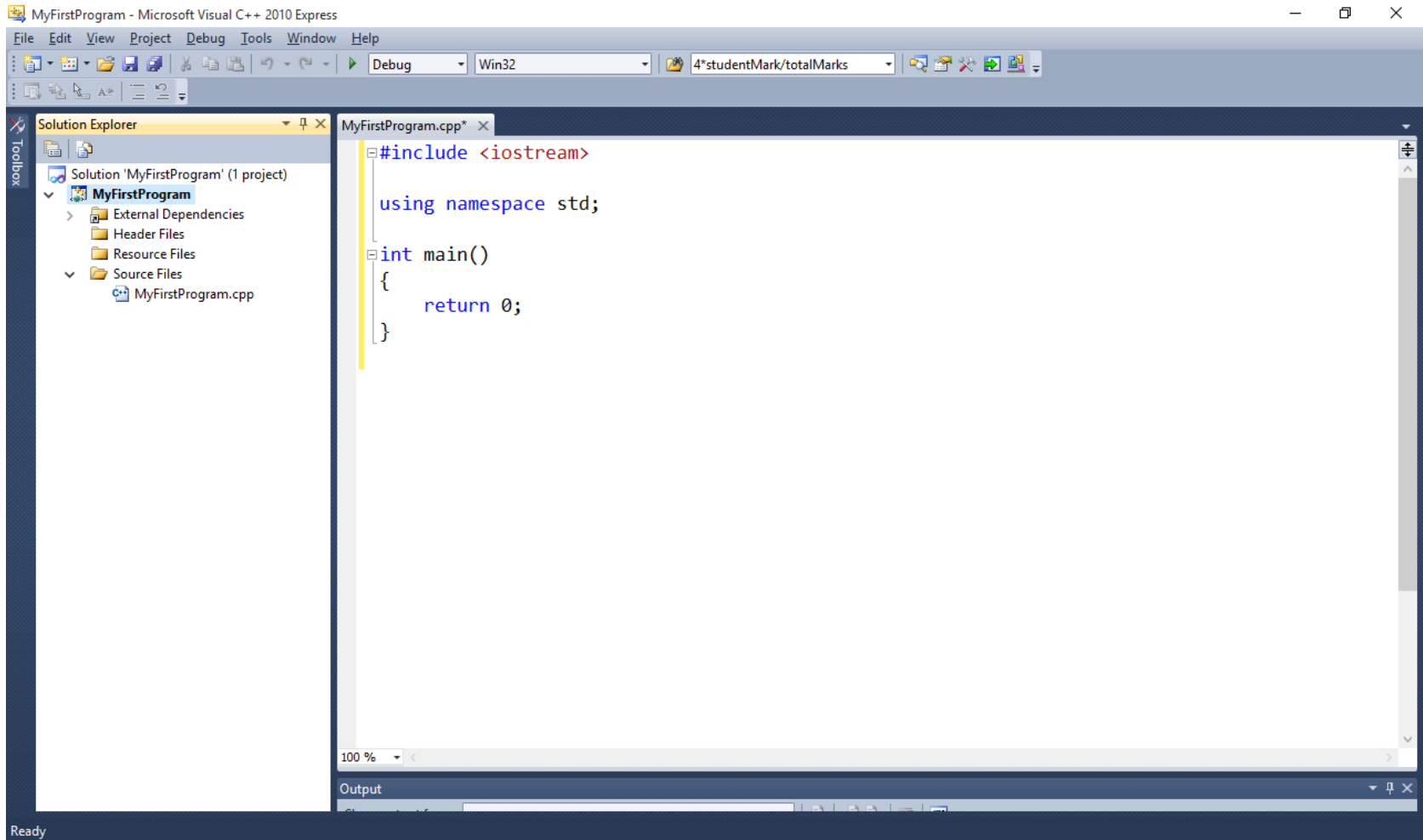
Hello World: Your first C++ program



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

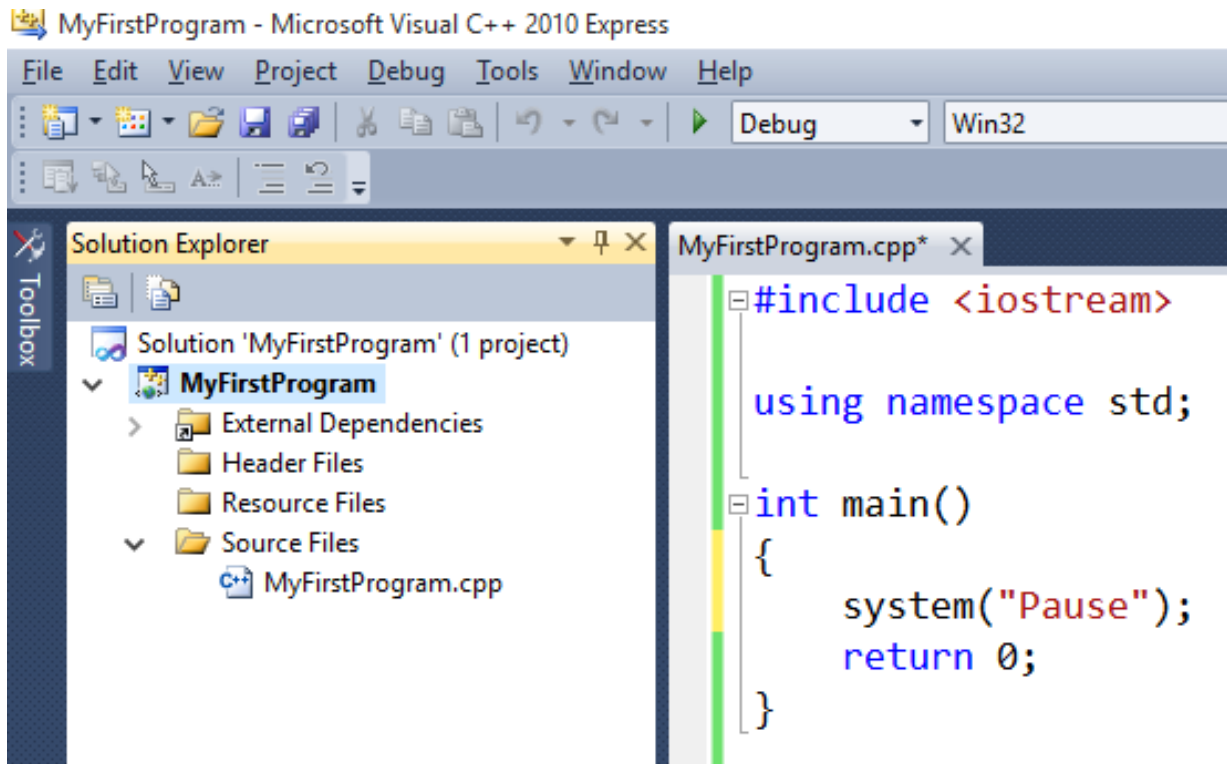
Hello World: Your first C++ program



Hello World: Your first C++ program

- In order to check the correctness of the program 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

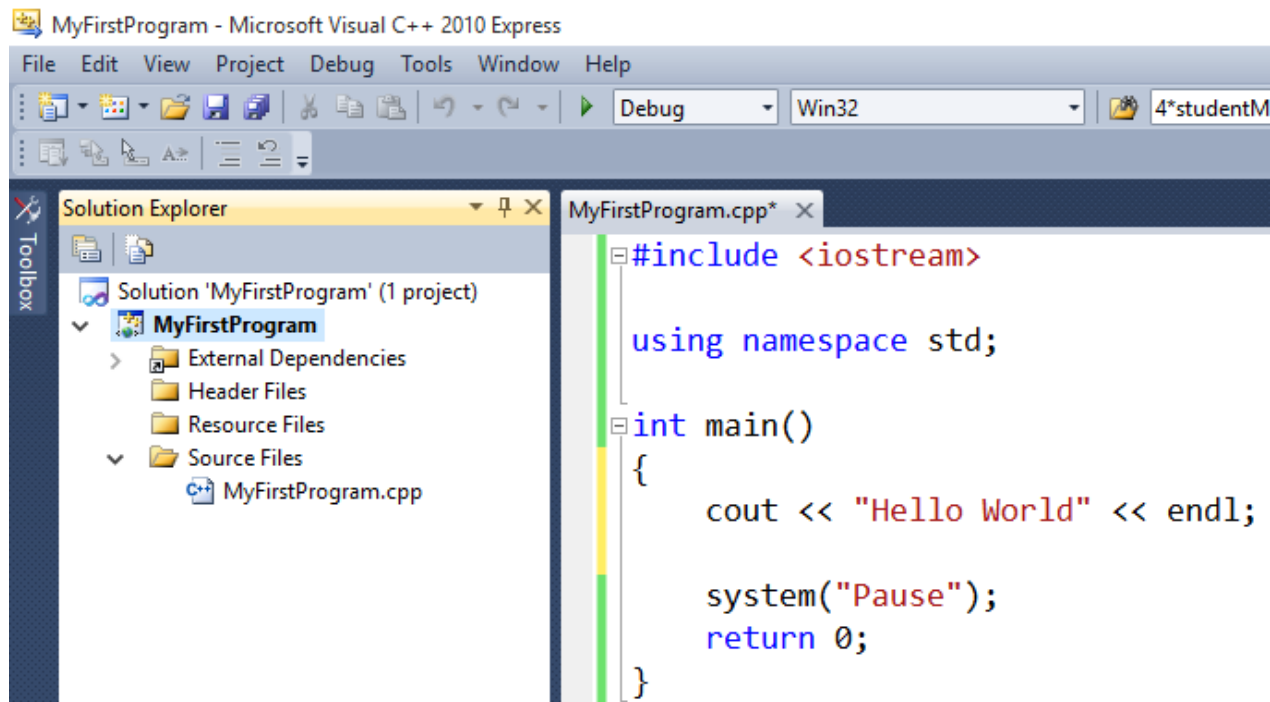
Hello World: Your first C++ program



- Now press **F7** to check the correctness of the program and once we get the message **Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped** 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
- This program will print the message **Hello World** on the output window and then ask you to press any key to close the program
- This completes your first Hello World C++ Program



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];

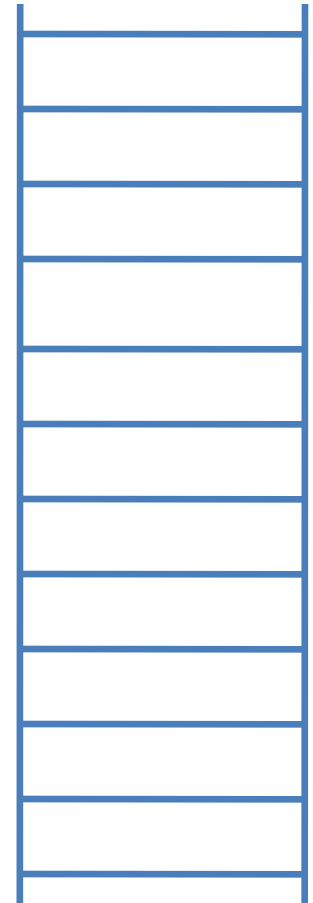
cout [<< 'ONE CHARACTER'] [<< endl];

- 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,....

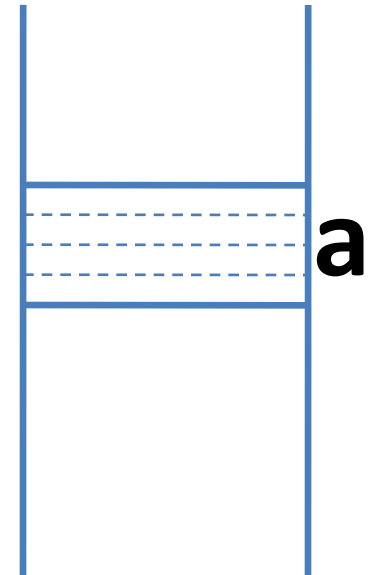
Variables in C++

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

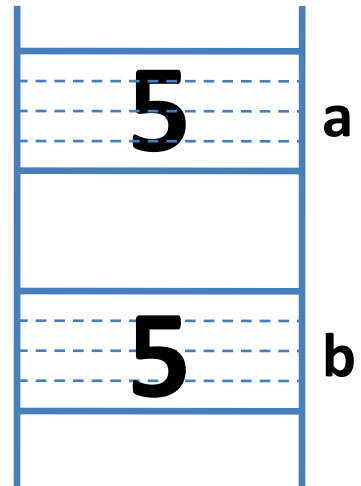
The Assignment Operator

- 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 side)
- 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;
```

5

Output

- 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;
```

The value of a is 5

Output

- 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;
```

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

Output

Variables and Data Types: Example

```
#include <iostream>

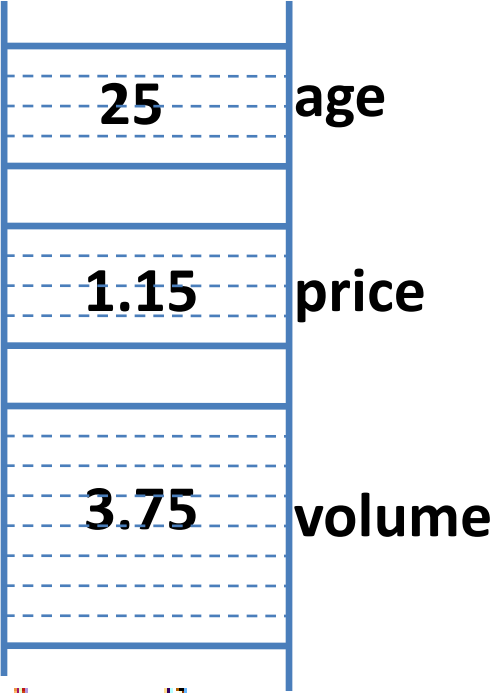
using namespace std;

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 oranges is $" << price << " per killogram." << endl;
    cout << "We are left with " << volume << " liters of gas." << endl;

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



The diagram illustrates the memory layout of the variables defined in the code. It consists of a vertical stack of three rectangular boxes, each representing a variable's memory space. The top box contains the value '25' and is labeled 'age' to its right. The middle box contains the value '1.15' and is labeled 'price' to its right. The bottom box contains the value '3.75' and is labeled 'volume' to its right. Each box is divided into three horizontal sections by dashed lines, suggesting a segmented or padded memory structure.

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, 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 = 3.9;` ---- > the variable a is assigned the integer value 3. We say the value 3.9 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;
```

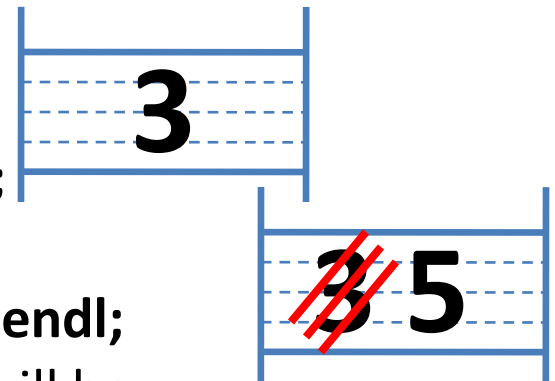
```
x = 5;
```

```
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

- **Syntax**

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;
}
```

Compound Arithmetic Operators

- C++ supports the following compound binary operators

Compound Operator	Example	Description
+=	a += b	a = a + b
-=	a -= 5	a = a - 5
*=	b *= c	b = b * c
/=	a /= 4.0	a = a / 4.0
%=	a %= b	a = a % b

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

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

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

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
 - Begin variable names with lower case
 - Variable names with two or more words should either be capitalized or separated by underscore character.
For example,
int numberOfStudents, my_age;
 - Insert as much needed as comments as possible
 - Always remember C++ is case sensitive