C++ Foundation with Data Structures

Lecture 2 : Getting Started

First Program

a) About Code blocks

Code blocks is an integrated development environment (IDE) for C/C++.
To set the path of compiler, follow the following steps –

- Click on menu Settings -> Compiler
- Then click on tab **Toolchain executables**
- In the text box under **Compiler's installation directory,** click on button **Auto Detect.** Then you should get the pop-up saying – **Auto detected installation path….**
- If pop-up says – **Couldn't auto detect** …., that means you have downloaded the incorrect setup of code blocks. Uninstall this setup and install the setup with mingw. Then repeat the above steps.

You can create a new file, by clicking on

- File -> New -> Empty File
- Then save that file with extension **.cpp**
- In order to compile and run the program, click on the button right next to a play button which says **build and run**.
- And then you should get your output window.

b) First two lines

We need to include the following two lines in our every C++ file –

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

Under print section we will use cout to print things and cin to take input. These two lines enable us to use them. The first line is required because that iostream file contains the definition of cin and cout. And the second line specifies which cin or cout is to be used. You will be able to understand these better as the course will progress.

c) About Main

Consider the following line of code:

```
int main() {}
```

1. This is the line at which the program will begin executing. This statement is similar to start block in flowcharts. All C++ programs begin execution by calling main()
2. We will understand why we write main this way. For now we should assume that we have to write main as it is.
3. The curly braces {} indicate start and end of main.

d) Print

In order to print things to console (as it is we write) we have to write –

```
cout << "Hello World";
```

We need to have "<<" operator with cout. And the text we want to display should be inside double quotes(""). Whatever is placed inside double quotes(except few special characters) will be displayed as it is.

If we want to add newline after our text, we can use either "\n" or endl. Both serve the same purpose, that is adds newline.

Example Code:

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

int main() {
   cout << "Hello World";
   cout << "Hello World" << "\n";
   cout << "Hello World" << endl;   // both are same
   cout << "Hello World";
}
```

Output:

Hello WorldHello World
Hello World
Hello World

Variables

a) Add two numbers

Consider the following code for adding two numbers

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

int main() {
   int a = 10;
   int b = 5;
   int ans = a + b;
   cout << ans;
}
```

Output:
15

Here, we used variables to store values of two integers and their sum. Thus, a variable is a basic unit of storage in a program.

Syntax for Declaring a Variable :
data_type variable_name [= value];

Here, data_type is one of data types available in C++. The variable_name is the name of a variable. We can initialize the variable by specifying an equal sign and a value (Initialization is optional). However, the compiler assigns a default garbage value if you don't initialize any variable in C++.

While writing variable names you should be careful and follow the rules for naming them. Following are the rules for writing variable names -

1. All variable names may contain uppercase and lowercase letters (a-z, A-Z), underscore (_) and the digits 0 to 9. The dollar sign character is not intended for general use. No spaces and no other special characters are allowed.
2. The variable names must not begin with a number.
3. C++ is case-sensitive. Uppercase characters are distinct from lowercase characters.
4. A C++ keyword (reserved word) cannot be used as a variable name.

b) Data types of variables

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, we can store integers, decimals, or characters in these variables.

The following are built-in data types in C++ :

DATA TYPE	DEFAULT SIZE
bool	1 byte
char	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
Float	4 bytes
Double	8 bytes

Taking Input

Just like cout, we have another shortcut to take input from user, that is cin.

Example Code:

```
//Code for adding two integers entered by the user

#include <iostream>
using namespace std;

int main() {
int a, b;
cin >> a;
cin >> b;
int ans = a + b;
cout << ans;
}
```

Sample Input:
10 5

Output:
15

Here, cin reads the values of variable a and b. We can enter multiple values either separated by space or separated by newline. So value 10 will be assigned to a and value 5 will be assigned to 5. Hence answer will be 15.

We can take multiple inputs in one line also. **S**o we could have combined statements cin>>a; and cin>>b; into one statement i.e cin >> a>> b;

How is Data Stored ?

a) How are integers stored ?

The most commonly used integer type is int which is a signed 32-bit type. When you store an integer, its corresponding binary value is stored. The way integers are stored differs for negative and positive numbers. For positive numbers the integral value is simple converted into binary value and for negative numbers their 2's compliment form is stored.

Let's discuss How are Negative Numbers Stored?

Computers use 2's complement in representing signed integers because:

1. There is only one representation for the number zero in 2's complement, instead of two representations in sign-magnitude and 1's complement.

2. Positive and negative integers can be treated together in addition and subtraction. Subtraction can be carried out using the "addition logic".

Example:

int i = -4;

Steps to calculate Two's Complement of -4 are as follows:

Step 1: Take Binary Equivalent of the positive value (4 in this case)
0000 0000 0000 0000 0000 0000 0000 0100

Step 2: Write 1's complement of the binary representation by inverting the bits

1111 1111 1111 1111 1111 1111 1111 1011

Step 3: Find 2's complement by adding 1 to the corresponding 1's complement

```
 1111 1111 1111 1111 1111 1111 1111 1011
+0000 0000 0000 0000 0000 0000 0000 0001
 -----------------------------------------------------------
 1111 1111 1111 1111 1111 1111 1111 1100
```

Thus, integer -4 is represented by the binary sequence (1111 1111 1111 1111 1111 1111 1111 1100) in C++.

b) Float and Double values

In C++, any value declared with decimal point is by default of type double (which is generally of 8 bytes). If we want to assign a float value (which is generally of 4 bytes), then we must use 'f' or 'F' literal to specify that current value is "float".

Example:

```
float float_val = 10.4f;            //float value
double val = 10.4;          //double value
```

c) How are characters stored

Every character has a unique integer value, which is called as ASCII value. As we know system only understands binary language and thus everything has to be stored in the form binaries. So for every character there is corresponding integer code – ASCII code and binary equivalent of this code is actually stored in memory when we try to store a char.

Example code:

```
int main(){
        char ch1, ch2;
        // We can assign any integer value to character variable also. Here it
is ASCII value for 'X'
        ch1 = 88;
        ch2 = 'Y';
        cout << ch1 << " " << ch2;
}
```

Output:
X Y

d) Adding int to char

When we add int to char, we are basically adding two numbers i.e. one corresponding to the integer and other is corresponding code for the char.

Example code:

```
int main() {
        cout << ('a' + 1);
}
```

Output:
98

Here, we added a character and an int, so it added the ASCII value of char 'a' i.e 97 and int 1. So, answer will be 98.

Similar logic applies to adding two chars as well, when two chars are added their codes are actually added i.e. 'a' + 'b' wil give 195.

Operators

a) Arithmetic operators

Arithmetic operators are used in mathematical expression in the same way that are used in algebra.

OPERATOR	DESCRIPTION
+	Adds two operands
-	Subtracts second operand from first
*	Multiplies two operands
/	Divides numerator by denominator
%	Calculates Remainder of division

b) Relational operators

Relational Operators are the operators that used to test some king of relation between two entities. The following table lists the relation operators supported by C++.

OPERATOR	DESCRIPTION
==	Check if two operands are equal
!=	Check if two operands are not equal.
>	Check if operand on the left is greater than operand on the right
<	Check if operand on the left is smaller than right operand
>=	Check if left operand is greater than or equal to right operand
<=	Check if operand on left is smaller than or equal to right operand

c) Logical operators

C++ supports following 3 logical operators. The result of logical operators is a Boolean i.e. true(which is any non zero value) or false(which is 0).

OPERATOR	DESCRIPTION
&&	Logical AND
||	Logical OR
!	Logical NOT

Example:
Suppose a = true and b= false, then:
(a && b) is false i.e 0
(a || b) is true i.e. 1
(!a) is false i.e. 0