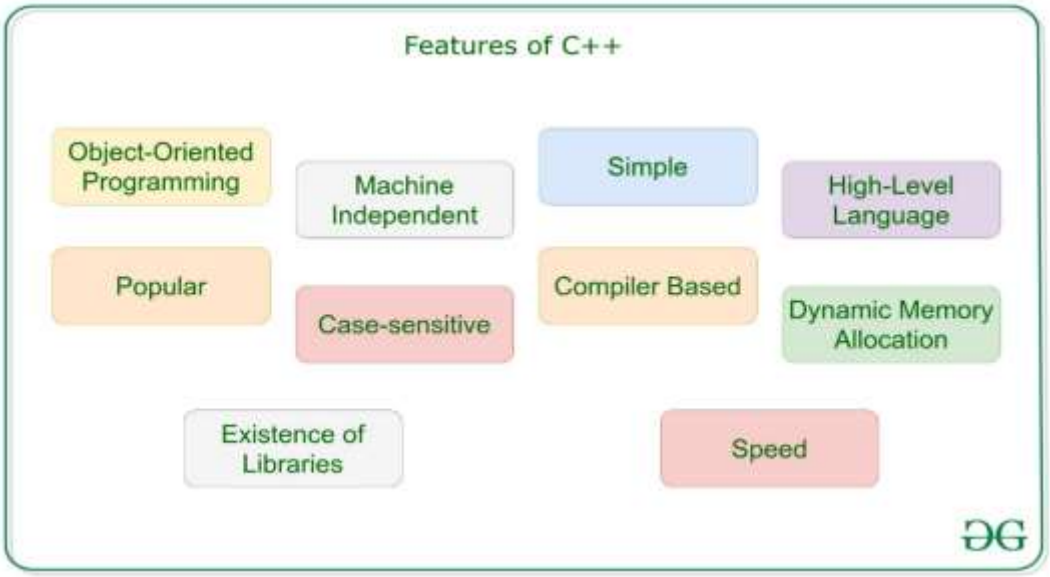
Lecture 3

1.1 Introduction to C++

C++ is a general-purpose programming language that was developed as an enhancement of the C language to include an object-oriented programming. A programming language is a set of rules that provides a way of telling a computer what operations to perform.



- C++ is an Object-Oriented Programming Language. This is the most important feature of **C++**. It can create/destroy objects while programming.
- C++ is a compiler-based language. That is C++ programs used to be compiled and their executable file is used to run it. Due to which C++ is a relatively faster language than Java and Python.
- C++ allows us to allocate the memory of a variable or an array in run time. This is known as **Dynamic Memory Allocation.**
- C++ is a case-sensitive programming language.

1.2 Character set

C++ has the letters and digits, as shown below:

Uppercase: A, B, C, . . ., Z

Lowercase: a, b, c, . . ., z

Digits: 0, 1, 2, . . .,9

Special Characters: All characters other than listed treated as special characters for example:

+	-	*	/	^	
(	[	{	}	]	
)	<	=	>		, (Comma)
“(Double Conations)	. (Dot)	: (Colon)	; (Semicolon)		(Blank Space)

In C++ language, upper case and lower-case letters are distinct and hence there are 52 letters in all. For example, a **bag** is different from **Bag** which is different from a **BAG.**

1.3 C++ Variables

Variables are containers for storing data values. In C++, there are different types of variables (defined with different keywords), for example:

Basic Data Types

The data type specifies the size and type of information the variable will store:

Data Type	Description
int	Stores whole numbers, without decimals
double	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
Boolean	Stores true or false values
char	Stores a single character/letter/number, or ASCII values
string	Stores text, such as "Hello World". String values are surrounded by double quotes.

1.4 C++ program structure

The program in the language (c ++) has a general form when it is written and it is almost constant in its main parts in all programs and the way it is written is as follows

```
#include <iostream>
using namespace std;
// main() is where program execution begins.
int main() {
   cout << "Hello World"; // prints Hello World
   return 0;
}
```

The line **int main()** is the main function where program execution begins.

- The next line **cout << "Hello World";** causes the message "Hello World" to be displayed on the screen.
- The next line **return 0;** terminates the main() function and causes it to return the value 0 to the calling process.

➢ C++ Identifiers

All C++ **variables** must be **identified** with **unique names**. These unique names are called **identifiers**. Identifiers can be short names (like x and y) or more **descriptive** names (age, sum, total volume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

The general rules for constructing names for variables (unique identifiers)are:

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive (myVar and myvar are different variables)
- Names cannot contain whitespaces or special characters like!, #,
- %, etc.
- Reserved words (like C++ keywords, such as int) cannot be used as names.

➢ Constants

When you do not want others (or yourself) to override existing variable values, use the const keyword (this will declare the variable as "constant", which means **unchangeable and read-only):**

Example

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```

➢ C++ User Input

- You have already learned that cout is used to output (print) values. Now we will use cin to get user input.
- cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).
- In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

Example

```
int x;
cout << "Type a number: "; // Type a number and press enter
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

➢ Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax

type variable = value;

Example

```
int myNum = 5;               // Integer (whole number without
decimals)
double myFloatNum = 5.99;    // Floating point number (with
decimals)
char myLetter = 'D';         // Character
string myText = "Hello";     // String (text)
bool myBoolean = true;       // Boolean (true or false)
```

Example

Create a variable called **myNum** of type `int` and assign it the value **15**:

```
int myNum = 15;
cout << myNum;
```

Example

```
int x = 5;
int y = 6;
int sum = x + y;
cout << sum;
```

➢ Declare Many Variables

To declare more than one variable of the same type, use a comma- separated list:

Example

```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

➢ C++ Operators

C++ divides the operators into the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators

➢ Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

➢ Assignment Operators

- Assignment operators are used to assigning values to variables.
- In the example below, we use the **assignment** operator (=) to assign the value **10** to a variable called **x**:
- The **addition assignment** operator (+=) adds a value to a variable:

Example

```
int x = 10;
x += 5;
```

A list of all assignment operators:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

➢ Comparison Operators

Comparison operators are used to comparing two values.

Note: The return value of a comparison is either true (1) or false (0).

A list of all comparison operators:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

In the following example, we use the **greater than** operator (>) to find out if 5 is greater than 3:

Example

```
int x = 5;
int y = 3;
cout << (x > y); // returns 1 (true) because 5 is greater than 3
```

➢ Logical Operators

Logical operators determine the logic between variables or values1:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)