OBJECT ORIENTED PROGRAMMING WITH C++

Procedural Programming: Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Languages used in Procedural Programming: FORTRAN, ALGOL, COBOL, BASIC, Pascal and C.

Object Oriented Programming: Object oriented programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object oriented programming, computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Languages used in Object Oriented Programming: Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala.

Difference between Procedural Programming and Object Oriented Programming:

Procedural Oriented Programming	Object Oriented Programming
In procedural programming, program is divided into small parts called *functions*.	In object oriented programming, program is divided into small parts called ***objects***.
Procedural programming follows *top down approach*.	Object oriented programming follows ***bottom up approach***.
There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way for hiding data so it is *less secure*.	Object oriented programming provides data hiding so it is ***more secure***.

In procedural programming, overloading is not possible.	Overloading is possible in object oriented programming.
In procedural programming, function is more important than data.	In object oriented programming, data is more important than function.
Procedural programming is based on *unreal world*.	Object oriented programming is based on ***real world***.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

Advantage of OOPs over Procedure oriented programming language:

1. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
2. OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
3. OOPs provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

C++ Programming Language

C++ is a special-purpose programming language was developed in **1980** by **Bjarne Stroustrup** at bell laboratories of AT&T (American Telephone & Telegraph), located in U.S.A.

C++ language is very similar to C language, and it is so compatible with C that it can run 99% of C programs without changing any source of code though C++ is an object-oriented programming language, so it is safer and well-structured programming language than C.

C++ programming is "relative" (called a superset) of C, it means any valid C program is also a valid C++ program.

Programming languages that were developed before C++ language:

Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
K & R C	1978	Kernighan & Dennis Ritchie
C++	1980	Bjarne Stroustrup

C++ Features

C++ is object oriented programming language. It provides a lot of **features** that are given below.

1. Simple
2. Machine Independent or Portable
3. Mid-level programming language
4. Structured programming language
5. Rich Library
6. Memory Management
7. Fast Speed
8. Pointers
9. Recursion
10. Extensible
11. Object Oriented
12. Compiler based

1) **Simple:** C++ is a simple language in the sense that it provides structured approach (to break the problem into parts), rich set of library functions, data types etc.

2) **Machine Independent or Portable:** Unlike assembly language, c programs can be executed in many machines with little bit or no change. But it is not platform-independent.

3) **Mid-level programming language:** C++ is also used to do low level programming. It is used to develop system applications such as kernel, driver etc. It also supports the feature of high level language. That is why it is known as mid-level language.

4) **Structured programming language:** C++ is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

5) **Rich Library:** C++ provides a lot of inbuilt functions that makes the development fast.

6) **Memory Management:** It supports the feature of dynamic memory allocation. In C++ language, we can free the allocated memory at any time by calling the free() function.

7) **Speed:** The compilation and execution time of C++ language is fast.

8) **Pointer:** C++ provides the feature of pointers. We can directly interact with the memory by using the pointers. We can use pointers for memory, structures, functions, array etc.

9) **Recursion:** In C++, we can call the function within the function. It provides code reusability for every function.

10) **Extensible:** C++ language is extensible because it can easily adopt new features.

11) **Object Oriented:** C++ is object oriented programming language. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

12) **Compiler based:** C++ is a compiler based programming language, it means without compilation no C++ program can be executed. First we need to compile our program using compiler and then we can execute our program.

C++ Program

#include<iostream.h> includes the **standard input output** library functions. It provides **cin** and **cout** methods for reading from input and writing to output respectively.

#include <conio.h> includes the **console input output** library functions. The getch() function is defined in conio.h file.

The **main() function is the entry point of every program** in C++ language.

The getch() function asks for a single character**.** Until we press any key, it blocks the screen.

C++ Basic Input/Output

C++ I/O operation is using the stream concept. Stream is the sequence of bytes or flow of data. It makes the performance fast.

If bytes flow from main memory to device like printer, display screen, or a network connection, etc, this is called as **output operation.**

If bytes flow from device like printer, display screen, or a network connection, etc to main memory, this is called as **input operation.**

I/O Library Header Files

Let us see the common header files used in C++ programming are:

Header File	Function and Description
<iostream>	It is used to define the **cout, cin and cerr** objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.
<iomanip>	It is used to declare services useful for performing formatted I/O, such as **setprecision and setw.**
<fstream>	It is used to declare services for user-controlled file processing.

Standard output stream (cout): The **cout** is a predefined object of **ostream** class. It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream insertion operator (<<) to display the output on a console.

Example of standard output stream (cout):

```
#include <iostream>
using namespace std;
int main( )
{
cout << "Welcome to C++ Programming.";
}
```

Output: Welcome to C++ Programming

Standard input stream (cin): The **cin** is a predefined object of **istream** class. It is connected with the standard input device, which is usually a keyboard. The cin is used in conjunction with stream extraction operator (>>) to read the input from a console.

Example of standard input stream (cin):

```
#include <iostream>
using namespace std;
int main( ) {
  int age;
   cout << "Enter your age: ";
   cin >> age;
   cout << "Your age is: " << age;
}
```

Output:

```
Enter your age: 22
Your age is: 22
```

Standard end line (endl): The **endl** is a predefined object of **ostream** class. It is used to insert a new line characters and flushes the stream.

Example of standard end line (endl):

```
#include <iostream>
using namespace std;
int main( )
{
cout << "Bengali";
cout << " English"<<endl;
cout << "Mathematics"<<endl;
}
```

Output:

```
Bengali English
Mathematics
```

C++ Variable

A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times. It is a way to represent memory location through symbol so that it can be easily identified.

syntax to declare a variable: type variable_list;

Example of declaring variable is given below:

```
int x;
float y;
char z;
```

Here, x, y, z are variables and int, float, char are data types.

C++ Identifiers

C++ identifiers in a program are used to refer to the name of the variables, functions, arrays, or other user-defined data types created by the programmer. They are the basic requirement of any language. Every language has its own rules for naming the identifiers.

In short, we can say that the C++ identifiers represent the essential elements in a program which are given below:

- **Constants**
- **Variables**
- **Functions**
- **Labels**
- **Defined data types**

Constants are the identifiers that refer to the fixed value, which do not change during the execution of a program.

Keywords are the reserved words that have a special meaning to the compiler. They are reserved for a special purpose, which cannot be used as the identifiers. For example, 'for', 'break', 'while', 'if', 'else', etc. are the predefined words where predefined words are those words whose meaning is already known by the compiler. Whereas, the identifiers are the names which are defined by the programmer to the program elements such as variables, functions, arrays, objects, classes.

There are a total of 95 reserved words in C++. A list of 32 Keywords in C++ Language which are also available in C language are given below.

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

A list of 30 Keywords in C++ Language which are not available in C language are given below.

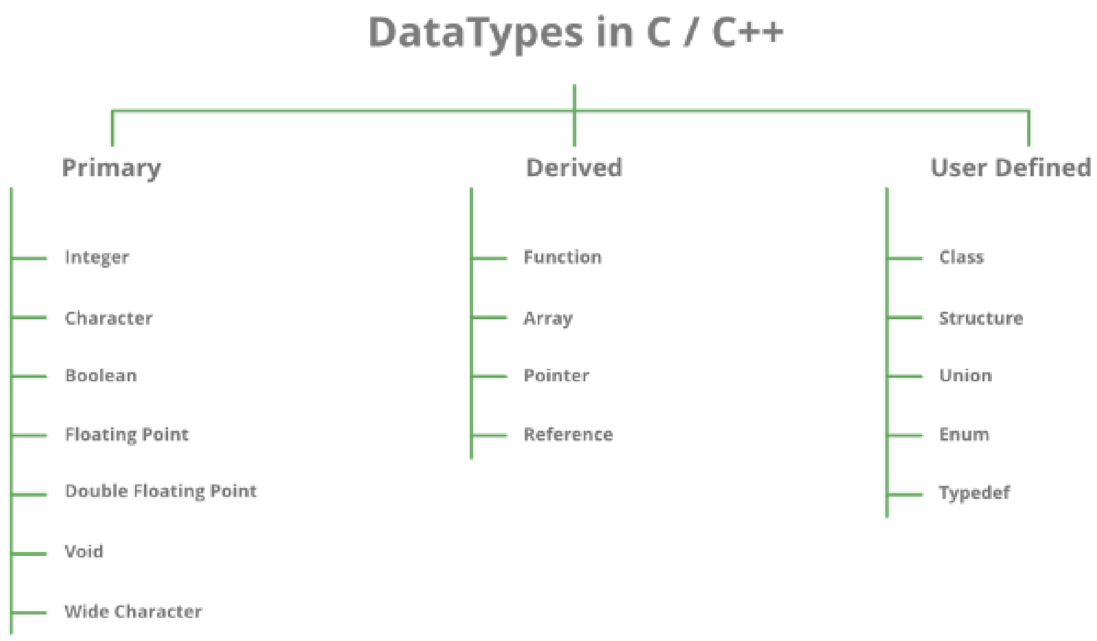
asm	dynamic_cast	namespace	reinterpret_cast	bool
explicit	new	static_cast	false	catch
operator	template	friend	private	class
this	inline	public	throw	const_cast
delete	mutable	protected	true	try
typeid	typename	using	virtual	wchar_t

❖ **Differences between Identifiers and Keywords**

Identifiers	Keywords
Identifiers are the names defined by the programmer to the basic elements of a program.	Keywords are the reserved words whose meaning is known by the compiler.
It is used to identify the name of the variable.	It is used to specify the type of entity.
It can consist of letters, digits, and underscore.	It contains only letters.
It can use both lowercase and uppercase letters.	It uses only lowercase letters.
No special character can be used except the underscore.	It cannot contain any special character.
The starting letter of identifiers can be lowercase, uppercase or underscore.	It can be started only with the lowercase letter.
It can be classified as internal and external identifiers.	It cannot be further classified.
Examples are test, result, sum, power, etc.	Examples are 'for', 'if', 'else', 'break', etc.

C++ Data Types

All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store. Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires a different amount of memory.



Data types in C++ is mainly divided into three types:

1. **Primitive Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc. Primitive data types available in C++ are:
   - Integer
   - Character
   - Boolean
   - Floating Point
   - Double Floating Point
   - Valueless or Void
   - Wide Character

2. **Derived Data Types:** The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

- Function
- Array
- Pointer
- Reference

3. **Abstract or User-Defined Data Types**: These data types are defined by user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:
   - Class
   - Structure
   - Union
   - Enumeration
   - Typedef defined DataType

Primitive Data Types

- **Integer**: Keyword used for integer data types is **int**. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.

- **Character**: Character data type is used for storing characters. Keyword used for character data type is **char**. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

- **Boolean**: Boolean data type is used for storing boolean or logical values. A boolean variable can store either *true* or *false*. Keyword used for boolean data type is **bool**.

- **Floating Point**: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is **float**. Float variables typically requires 4 byte of memory space.

- **Double Floating Point**: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is **double**. Double variables typically requires 8 byte of memory space.

- **void**: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.

- **Wide Character**: Wide character data type is also a character data type but this data type has size greater than the normal 8-bit datatype. Represented by **wchar_t**. It is generally 2 or 4 bytes long.

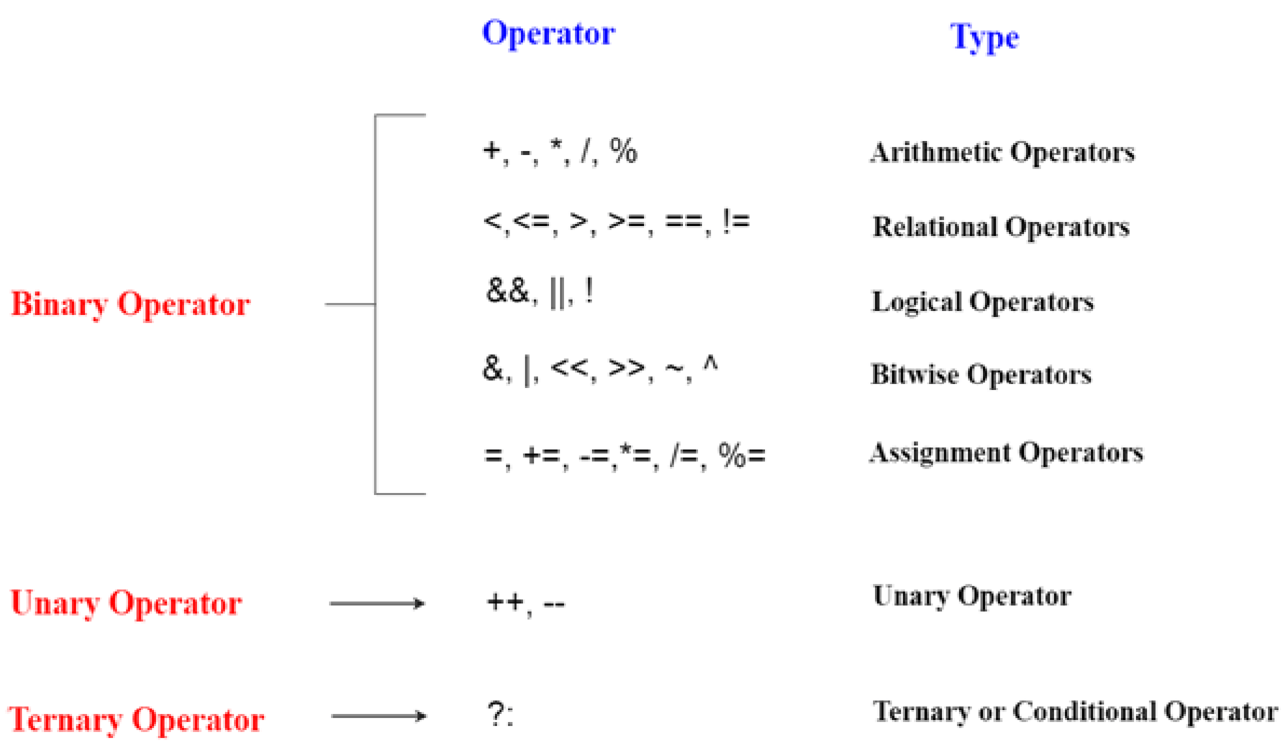
Data Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	8	-2,147,483,648 to 2,147,483,647
unsigned long int	8	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

C++ Operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

There are following types of operators to perform different types of operations in C language.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operator
- Unary operator
- Ternary or Conditional Operator



	Operator	Type
Binary Operator	+, -, *, /, %	Arithmetic Operators
	<,<=, >, >=, ==, !=	Relational Operators
	&&, ||, !	Logical Operators
	&, |, <<, >>, ~, ^	Bitwise Operators
	=, +=, -=,*=, /=, %=	Assignment Operators
Unary Operator	++, --	Unary Operator
Ternary Operator	?:	Ternary or Conditional Operator

Precedence of Operators in C++

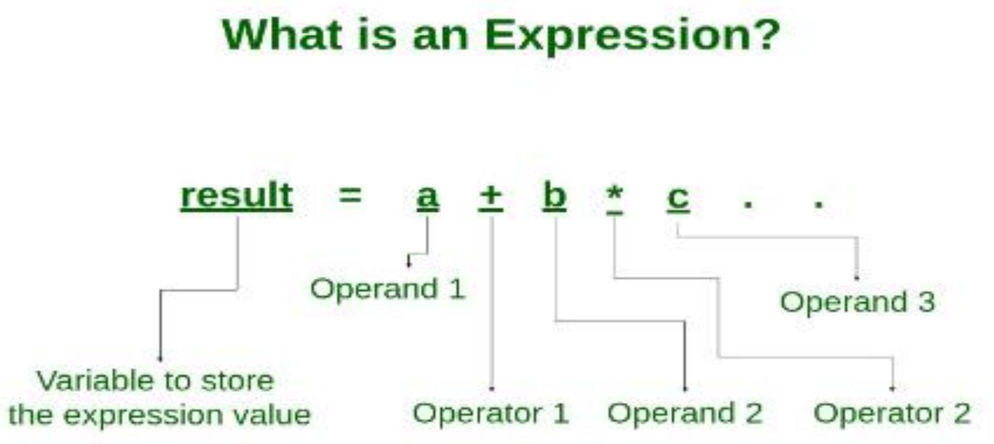
The precedence of operator species that which operator will be evaluated first and next. The associativity specifies the operators direction to be evaluated, it may be left to right or right to left.

❖ **The precedence and associativity of C++ operators is given below:**

Category	**Operator**	**Associativity**
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Right to left
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=/td>	Right to left
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Right to left
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

C++ Expression

Expression: An expression is a combination of operators, constants and variables. An expression may consist of one or more operands, and zero or more operators to produce a value.



Examples of C++ expression:

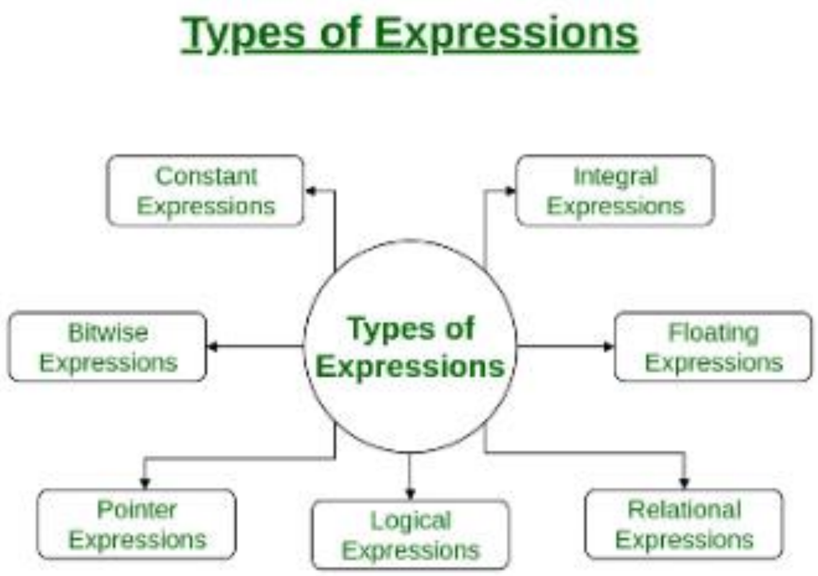
(a+b) - c

(x/y) -z

4a2 - 5b +c

(a+b) * (x+y)

Types of Expressions: Expressions may be of the following types:



- **Constant expressions**: Constant Expressions consists of only constant values. A constant value is one that doesn't change.
  Examples: 5, 10 + 5 / 6.0, 'x'

- **Integral expressions**: Integral Expressions are those which produce integer results after implementing all the automatic and explicit type conversions.
  Examples: x, x * y, x + int(5.0)
  where x and y are integer variables.

- **Floating expressions**: Float Expressions are which produce floating point results after implementing all the automatic and explicit type conversions.
  Examples: x + y, 10.75
  where x and y are floating point variables.

- **Relational expressions**: Relational Expressions yield results of type bool which takes a value true or false. When arithmetic expressions are used on either side of a relational operator, they will be evaluated first and then the results compared. Relational expressions are also known as Boolean expressions.
  Examples:x <= y, x + y > 2

- **Logical expressions**: Logical Expressions combine two or more relational expressions and produces bool type results.
  Examples: x > y && x == 10, x == 10 || y == 5

- **Pointer expressions**: Pointer Expressions produce address values.
  Examples:&x, ptr, ptr++
  where x is a variable and ptr is a pointer.

- **Bitwise expressions**: Bitwise Expressions are used to manipulate data at bit level. They are basically used for testing or shifting bits.
  Examples:
  x << 3

  shifts three bit position to left

  y >> 1

  shifts one bit position to right.

  Shift operators are often used for multiplication and division by powers of two.

An expression may also use combinations of the above expressions. Such expressions are known as **compound expressions**.