UNIT -1

* **C++ :-** C++ is an object oriented programming language. It was developed at AT & T lab, bell laboratories in early 1979. It was rename as C++. C++ language is an extension to C programming language and supports classes, inheritance, functions, loading and operator overloading which where not supported by C programming language.

The object oriented features in C++ is helpful in large program with extensibility and easy to maintain the software after sale to costumer. It is helpful to make the real world problems properly.

C++ is a middle level programming language developed by Bjarne Stroustrup starting in 1979 at Bell labs. C++ runs on a variety of platforms such as Windows, MAC OS and a various versions of UNIX.

Object oriented programming including the four pillers of object oriented development -

1- Encapsulation
2- Data hiding
3- Inheritance
4- Polymorphism

* **Learning C++ :-** The most important thing while learning C++ is to focus on concepts. The purpose of learning a programming language is to become more effective at designing and implementing new systems and at maintaining old ones.

* **Characters used in C++ :-**

1- Alphabats
2- Uppercase letters A to Z.
3- Lowercase letters a to z.
4- Numbers 0 to 9
5- Special characters -

+	Plus	"	Double quotes
-	Minus	&	Ampersand
*	Asterisk	#	Hash
^	Caret	$	Dollar
/	Slash	%	Percent
|	Vertical bar	>	Greater than
~	Tilde	<	Less than
)	Close parenthesis	=	Equal to
(	Open parenthesis	!	Exclamation mark

White space character - A character that is used to produce blank space when printed in C++ is called white

space character. These are space tags, new lines and comments.

* 60 Keywords or reserve words :- Keywords are reserved words. C++ language use the following keywords which are not available to users to use them as variables, function names. Generally all keywords are in lower case although uppercase of same name can be use as an identifier.

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	public	try
catch	float	protected	typedef
char	for	register	union
class	friend	return	unsigned
const	goto	short	virtual
continue	if	signed	void
default	inline	sizeof	volatile
do	int	static	while
delete	long	struct	throw

Add by ANSI C++ new keywords –

bool	export	reinterpret_cast	typename
const_cast	false	static_cast	using
dynamic_cast	mutable	true	wchar

Keywords are predefine reserve words used in programming that have special meaning to the compiler. Keywords are the part of the syntax and they cannot be used as an identifier.

For example – Keyword

int house → variable

Here int is a keyword that indicates house is a variable identifier to type integer.

Keyword cannot be changed by the user. They are always written in lower case.

* **Identifiers :-** Identifiers are user-defined words. Identifiers refers to name given to entities such as variables, fuctions, arrays etc.

Some rules of naming identifiers are –

1- First character should be an alphabat or underscore.
2- The name should not be a keyword.
3- Since C++ is case sensitive the uppercase and lowercase characters are consider different.
~~4~~- For ex – Code, code, CODE
4- A valid identifier can have character both upper and lower case, digits and

underscore.

Some examples of valid identifiers are - Nirbhay, Tech_fully, Youtube123.

Some examples of invalid identifier names are -

(i) 1234Youtube → first character should be an alphabat
(ii) rec# → # is a special character.
(iii) account no → blank space is not allowed.

* **Variable** :- A variable is a name of memory location. It is used to store data. It's value can be changed and it can be used many times. It is a way to represent memory location through symbol, so that it can be easily identify.

For ex -
```
int x;
float y;
char z;
```

* **Data types** :- A data type specifies the type of data that a variable can store such as - integer, floating, characters etc. There are 4 types to data types in

C++ language.

	Types	data types
1.	Basic data types	int, char, float, double etc
2.	Derived data types	array, pointer etc
3.	Enumeration data type	enum
4.	User defined data type	structure

The basic data types are integer based and floating point based. C++ language supports both signed and unsigned literals. The memory size of basic data types may change according to 32 or 64 bit operating system.
Let us see the basic data types, its size is given according to 32 bit OS.

Data type	Size	Range
char	1 byte	-128 to 127
signed char	1 byte	128 to 127
unsigned char	1 byte	0 to 127
short	2 bytes	-32768 to 32767
signed short	2 bytes	-32768 to 32767
unsigned short	2 bytes	0 to 32767
int	2 bytes	-32768 to 32767
signed int	2 bytes	-32768 to 32767
unsigned int	2 bytes	0 to 32767
short int	2 bytes	-32768 to 32767
signed short int	2 bytes	-32768 to 32767

unsigned short int	2 bytes	0 to 32767
long int	4 bytes	
signed long int	4 bytes	
unsigned long int	4 bytes	
float	4 bytes	
double	8 bytes	
long double	10 bytes	

* Semi colon and blocks in C++ :- The semi colon is a statement terminater that is each individual statement must be ended with a semi colon. It indicates the end of one logical entity.
For ex- following are there different statements -

```
x = y;
y = y + 1;
```

A block is a set of logical connected statement that are surrounding by openning and closing braces.
For ex-

```
{
cout << "Hello World";
return 0;
}
```

* **Comments in C++ :-** Program comments are explainatory statement that you can include in the C++ codes that you write and have any one reading its source code. All programming languages allow for some form of comment. C++ supports single-line and multi-line comments. All characters available inside any comments are ignore by C++ compiler.

For ex-

```
{
/* cout << "Hello World"; */
   Cout << "Hello";
   return 0;
}
```

When the above code is compiled it will ignore cout << "Hello World"; and final executable will produce the following result - Hello

* **Variable Definition :-** A variable definition mean that the programmer write some instruction to tell the compiler to create the storage in a memory location.

Syntax -

```
data_type variable_name, variable_name;
```

Here data-type means the valid C++ data type which include int, float, char,

double, boolean and variable list is the list of variable names to be declared which is seperated by comma.
For ex -

```
char letter;  → Variable definition
float area;
```

* **Variable initialization in C++:-** Variables are declared in the above example but none of them has been assign any value. Variable can be initialized and the initial value can be assign along with their declaration.
Syntax -

```
data_type variable name = value;
```

For example -

```
char letter = 'A';
float area = 26.5;
```

* **Scope of variables:-** All the variables have their area of functioning and out of that boundry they do not hold their value, this boundry is called scope of variables. For most of the cases its between the curly braces in which variables is declared that a variable exist not outside it. We can

divide variables in two main types-

1- Global variables

2- Local variables

Program 1- A program to understand the use of cout object.

Ex-

```
#include <iostream.h>
#include <conio.h>
void main() {
clrscr();
int length;
length = 10;
cout << "The length is";
cout << length;
getch();
}
```

→ Insertion operator

Output :- The length is 10.

Program 2:-

```
#include <iostream.h>
#include <conio.h>
int a = 10; → global variable
int main() {
int a = 15; → local variable
cout << "local a": << a << "global a:" << a;
}
```

Output - local a: 15
global a: 10

★ Cin (Console Input) :- It is similar to scanf() in C programming. It is used to read input value of variables from

the keyboard. It is an object of istream class.

Syntax –

cin >> variable_name;
(>> → Extraction)

Program3- A program to view the use of cout & cin objects.

```
#include <iostream.h>
#include <conio.h>
void main(){
    int a;
    cout<<"Enter any number:";
    cin >> a;
    cout<<" You are the best student:"<<a;
    getch();
    return 0;
}
```

Output :- Enter any number : 6
You are the best student : 6

* Cout :- It is similar to printf() in C programming. It is used to print / display value of variables. It is an object of ostream class.

Program4- A program to view the use of cout object-

```
#include <iostream.h>
#include <conio.h>
void main(){
    cout<<"Welcome to C++ programming\n";
    cout<<"Welcome\n BCA\n students";
    getch();
}
```

```
return 0;
}
```

Output :- Welcome to C++ programming
Welcome
BCA
students.

★ Abstraction :- Abstract classes are the way to achieve abstraction in C++. Abstraction in C++ is the process to hide the internal details, showing the functionality only. Abstraction can be achieve by two ways -

1. Abstract class
2. Interface

Abstract class and interface both can have abstract method which are necessary for abstraction. Data abstraction refers to providing only essential information to the outside world and hiding their background details to represent the needed information in program without presenting the details.

Data abstraction is a programming (and design) technique that relies on the seperation of interface and implementation. One real life example of a T.V. which you can turn on and off, change the channel, adjust the volume and add

external components such as speakers, DVD player but you do not know its internal details.

* **Encapsulation** :- Encapsulation is the process of enclosing within class and object the attributes and the methods. In other words, encapsulation allow wrapping upto data and functions into a single unit.

Advantages of Encapsulation-

1. Data hiding - Abstraction of external access a feature of a class.

2. Information hiding - The implementation of data are not known the users.

3. Implementation independency - A change in the implementation is done easily without affecting the interface.

* **Polymorphism in C++** :- The word polymorphism means having many forms. We can say polymorphism as the ability of a message to be display in more

than one form.
A real life example of polymorphism - a person at a same time can have different characteristics. Like a man at a same time is a father, a husband, an employee so a same person posses have different behaviour in different situation, this is called polymorphism. Polymorphism mainly divided in two types -

1- Compile time polymorphism (Compile time binding, early binding, static binding)
2- Run time polymorphism (Run time binding, late binding, dynamic binding)

1- Compile time polymorphism - This type of polymorphism is achieve by function overloading or operator overloading.

Function overloading - Where there are multiple function with same name but different parameters, then this function are said to be overloaded. Function can be overloaded by change in number of argument and change in type of argument.

2- Run time polymorphism - This type of polymorphism is achieve by function overriding. A function overriding on the

other hand access when a derived class has a definition for one of the member function of the base class that base function is said to be overriding.

* Inheritance :- The capability of a class to derive properties and characteristics from other class is called inheritance. Inheritance is one of the most important feature of object oriented programming.

Super class (Base class or parent class) - The class whose properties are inherited in sub class is called base or super class.

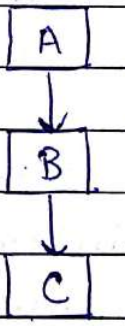
Sub class (Child class or Derived class) - The class that inherit properties from other class is called sub-class or derived class or child class.

Types of Inheritance – There are five Types of inheritance.

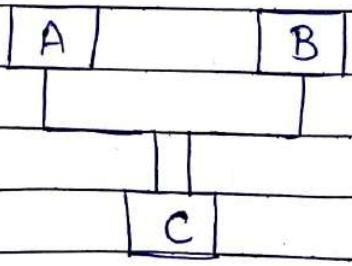
1. Single inheritance
2. Multiple inheritance
3. Multi-level inheritance
4. Hierarchical inheritance
5. Hybrid inheritance

1- Single Inheritance –

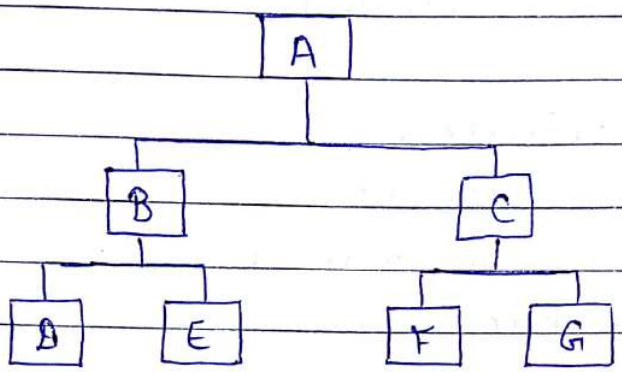
2- Multi-level Inheritance –



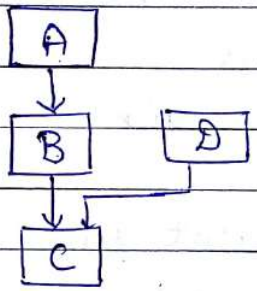
3- Multiple Inheritance –



4- Hierar



5- Hybrid Inheritance –



* Object – It is a class type variable.

* Class – A class is a group of common behaviour and functions. A class is a user-defined data-type like any other built-in data-type.

Example–

class → class student
{
- - - - - -
- - - - -
}

student stu1, stu2, stu3;
(stu1, stu2, stu3 → Objects)

* Example of Encapsulation, Inheritance —

```
class student
{
private:
    int id;
    char name;
public:
    void getdata();
    void display()
{
cout << id << \n << name. << end;
} };
int main()
{
student stu1, stu2, stu3;
-- --;
- - - -;
```

Access control specifies	Accessible to own class members	Object of class
Private	Yes	NO
Protected	Yes	NO
Public	Yes	Yes