01. Concept of OOPS

What is C++?

C++ is a high level general purpose Object Oriented Programming Language developed by “Bjarne Stroustraps” at Bell Lab in 1980.

Syntax

```
#include <iostream>
using namespace std;
int main(){
    // Code
    return 0;
}
```

Steps

1. Documentation
2. Preprocessor Directive
3. Global Declaration
4. Function Declaration

Difference - Procedural vs OOPS

PROCEDURAL	OOPS
deals with Algorithm	deals with Data
needs less memory	more memory compare to POP
concept of Top Down approach	concept of Bottom Up
Programs are divided into Functions	Programs are divided into Objects
Overloading is not Possible	Overloading is possible
Less Security	Highly Secure
Data handling is not possible	Data handling is possible
Does not have Access Specifier	Have Access Specifier

Various Library files in C++

What is Library file?

- A library file in programming is a collection of pre-compiled code that can be used by programs to perform specific tasks without the need for rewriting the code. Libraries provide reusable functions, classes, and routines that help in efficient program development by offering pre-written and tested code.

Examples

1. stdio.h

   >> printf, scanf, getc, puts, gets

2. conio.h

   >> clrscr, getch

3. stdlib.h

   >> ataf, atoll, rand, exit, abort

4. ctype.h

   >> isalpha, isdigit, islower, isupper

5. math.h

   >> sqrt, pow, floor, sign, cos

6. string.h

   >> strlen, strcmp, strrev, strlwr, strupr

7. process.h

   >> getpid, beginthread, endthread

What is Function?

- Function is a block of code.
- One or more statement in it.
- Performing a functionality

Example

```
int add(int a, int b){
    return a+b;
}
```

What is Object?

- In object-oriented programming, an object is an instance of a class that encapsulates data (attributes) and behavior (methods). Objects represent real-world entities or concepts, with

attributes representing their state and methods defining their interactions.

Features of C++

1. Simple

- It is a simple language in the sense that program can be broken down into logical unit and parts
- It has a rich library

2. Machine Independent / Portable

- C++ program can be executed in many machine with little bit or no change
- It is not platform independent

3. Midlevel

- We can do both system programming and build large scale user application so it is called midlevel

4. Structured Programming Language

- C++ is a Structured Programming Language that means we can break the program into parts using functions
- It is easy to understand and modify

5. Rich Library

- C++ provides a lot of functions and library that makes the development fast

6. Memory Management

- It supports the memory allocation.
- We can free the allocated memory at anytime by calling the "**free()**" function

7. Speed

- The compilation and execution time of C++ language is fast

8. Pointer

- C++ provide the feature of pointer.
- We can directly interact with the memory by using the pointer.

9. Extensible

- It is Extensible because it can easily adopt new features

10. OOP language

- C++ follow all the OOPs concepts like Class, Object, Polymorphism and Inheritance.

C++ OOPS basic Concept

1. Class

- Collection of objects
- Passive entity
- User defined data type which has data member & member function

2. Object

- Any entity that has state and behavior is known as object
- Object is a active entity
- An object is an instance of class
- When a class is define, no memory is allocated but when it is Instantiated

3. Inheritance

- inheritance is a mechanism where a new class (called the derived or child class) acquires the properties and behaviors (methods) of an existing class (called the base or parent class). This allows for code reusability and the creation of a hierarchical relationship between classes.

4. Polymorphism

- When one task is performed by different ways are known as polymorphism
- polymorphism is the ability of different objects to respond to the same function or method call in different ways. It allows methods to do different things based on the object it is acting upon, promoting flexibility and integration in code.
- **Example**:- A person at the same time can have different character like father, husband, employee so the same person possess different behavior in different situations.
- **There are two types of polymorphism**

1. Compile time
    a. Function overloading
    b. Operator overloading
2. Runtime
    a. Virtual Function
    b. Function Overriding

5. Abstraction

- abstraction is the concept of hiding the complex implementation details of a system and exposing only the essential features to the user. It allows for the simplification of complex systems by focusing on what an object does rather than how it does it.

6. Encapsulation

- encapsulation is the concept of bundling the data (attributes) and the methods (functions) that operate on the data into a single unit, or class. It restricts direct access to some of the object's components, which can help prevent the accidental modification of data, and provides a controlled interface for accessing and modifying that data.

7. Message Passing

- message passing is the process by which objects communicate with each other by sending and receiving messages. A message is a request for an object to execute one of its methods or to perform a certain action. This interaction allows objects to collaborate and coordinate their behavior to achieve a desired outcome.

8. Dynamic Binding

- dynamic binding (or late binding) is the process where the method to be executed in response to a function call is determined at runtime rather than compile time. This allows for flexibility in code, enabling objects to use overridden methods based on their actual types at runtime, which supports polymorphism.

DATA-TYPE IN CPP

1. In-Built Datatype

a. Integer

b. Character

c. Float

d. Double

e. Void

2. Derived Datatype

a. Function

b. Array

c. Pointer

d. References

3. User Defined Datatype

a. Class

b. Structure

c. Union

d. Enum

e. Typeof

f. Sizeof

Scope Resolution

- scope resolution is a mechanism used to specify and access the scope of variables, functions, and class members. It is achieved using the scope resolution operator (`::`). It helps distinguish between identifiers with the same name but in different scopes, such as accessing global variables, defining class member functions outside the class, and referencing members of namespaces.

Manipulators

- manipulators are special functions or objects that modify the behavior of input and output streams. They are used to control the formatting of data when it is written to or read from streams like `std::cout` and `std::cin`. Manipulators help adjust aspects such as field width, precision, alignment, and other formatting details.
- Manipulators are used to format the appearance of data in output streams or control how input data is read. They help in customizing the display and alignment of data.
- They are used with the stream insertion operator (`<<`) and the stream extraction operator (`>>`). For example, they can change the width of output fields, set the number of decimal places, or force alignment.
- These manipulators are defined in the `<iomanip>` header and can be combined in a single stream operation to achieve complex formatting. By using manipulators, developers can ensure that their program's output is consistently formatted, making it more readable and professionally presented.

Concept of String

- A string in C++ is a sequence of characters used to represent and manipulate text.
- C++ provides two main ways to handle strings
- **C-Style Strings**: These are arrays of characters terminated by a null character (`'\0'`). They require manual management of memory and string operations.
- `std::string` **Class**: Part of the C++ Standard Library, `std::string` is a class that provides a more flexible and convenient way to handle text. It supports dynamic sizing, automatic memory management, and a variety of functions for operations such as concatenation, substring extraction, and searching.
- Functions: `strlen()`, `strcmp()`, `strncmp()`, `strcpy()`, `strncpy()`, `strcat()`, `strncat()`, `strchr()`, `strrchr()`, `strstr()`, `strtok()`, `memcpy()`, `memmove()`, `memcmp()`, `memset()`, `memchr()`

Character Array

- A character array in C++ is an array of `char` elements used to store a sequence of characters, often representing a string. It must be null-terminated (`'\0'`) to indicate the end of the string.
- A character array initialized with `"Hello"` will contain the characters `H`, `e`, `l`, `l`, `o`, and `\0`.

Pointer to character array

- A pointer to a character array in C++ is a variable that holds the memory address of the first element of a character array. It is used to manipulate and access the elements of the character array through pointer arithmetic.
- A pointer to a character array is declared using the `char*` type. This pointer can be used to traverse and modify the character array.
- If you have a character array `char str[] = "Hello";`, a pointer to this array can be declared as `char* ptr = str;`. You can then access and modify the array elements using the pointer.

Memory Management Operators

- memory management operators are used to dynamically allocate and deallocate memory during the program's execution. The primary operators for this purpose are `new` and `delete` .
- **new**: Allocates memory on the heap for a single object.
- `pointer_variable = new data_type;`
- **delete:** Deallocates memory that was previously allocated for a single object using `new`.
- `delete pointer_variable;`

Type Cast

- Type casting in C++ is a way to convert a variable from one data type to another. This can be useful when you need to perform operations that require operands of the same type or when you need to interpret data in a different type. There are several ways to perform type casting in C++
- **Implicit Type Casting**:
  Converting
  `int` to `float` during an arithmetic operation.
- **Explicit Type Casting (C-Style Cast)**:

`(float) a` converts `a` to `float`.