C++ Programming

Trainer : Rajiv K

Email: rajiv.kamune@sunbeaminfo.com

Agenda

- High level language and it’s type
- Limitation of C language.
- OOP language and It’s History
- Major Pillar and Minor Pillar of OOP
- Data types in C++
- Structure in C++
- Difference in structure in C and C++
- Difference between POP and OOP.

What is High level language ?

- Programmers can **easily understand or interpret or compile** the high level language in comparison of machine.

- Examples of high level languages are **C, C++, Java, Python**, etc.

- Types of High Level Language
  - Procedure oriented Language
  - Object Oriented Language

Procedure Oriented Language

- A procedural language is a sort of computer programming language that has a **set of functions, instructions, and statements**.
- **"FORTRAN" is considered as first** high level POP language.
- All POP languages follows **"TOP Down"** approach.
- Function:- **a block of code which only runs when it is called.**
- Syntax:-

```
return_type function_name(//parameter list)
{
        //Body of a function
}
```

- Main function:- The main function **serves as the starting point for program execution(Entry Point Function/Call back function)**.

C Language Limitation

- C is said to be **process oriented**, **structured programming** language.
- When program becomes complex, **understating and maintaining** such programs is very **difficult**.
- Language **don’t** provide **security** for data.
- Using functions we can achieve code reusability, **but reusability is limited**.
- The programs are **not extendible.**

History of OOP and C++

- C++ is a product of the **1980s** and the work done by **Bjarne Stroustrup.**
- C++ is derived from C and Simula
- Its initial name was "**C With Classes**" which is developed in **"AT&T Bell Lab" in 1979.**
- In **1983 ANSI** renamed "C With Classes" to **C++.**
- It is developed on **Unix Operating System**.
- C++ is object oriented programming language

OOPS(Object Oriented Programming Language)

- It is a **programming methodology** to **organize** complex program in to simple program in terms of **classes and object** such methodology is called oops.

- It is a programming methodology to organized complex program into simple program by using concept of **abstraction** , **encapsulation** , **polymorphism** and **inheritance**.

- Languages which **support abstraction , encapsulation polymorphism and inheritance** are called OOP language.

Major pillars of oops

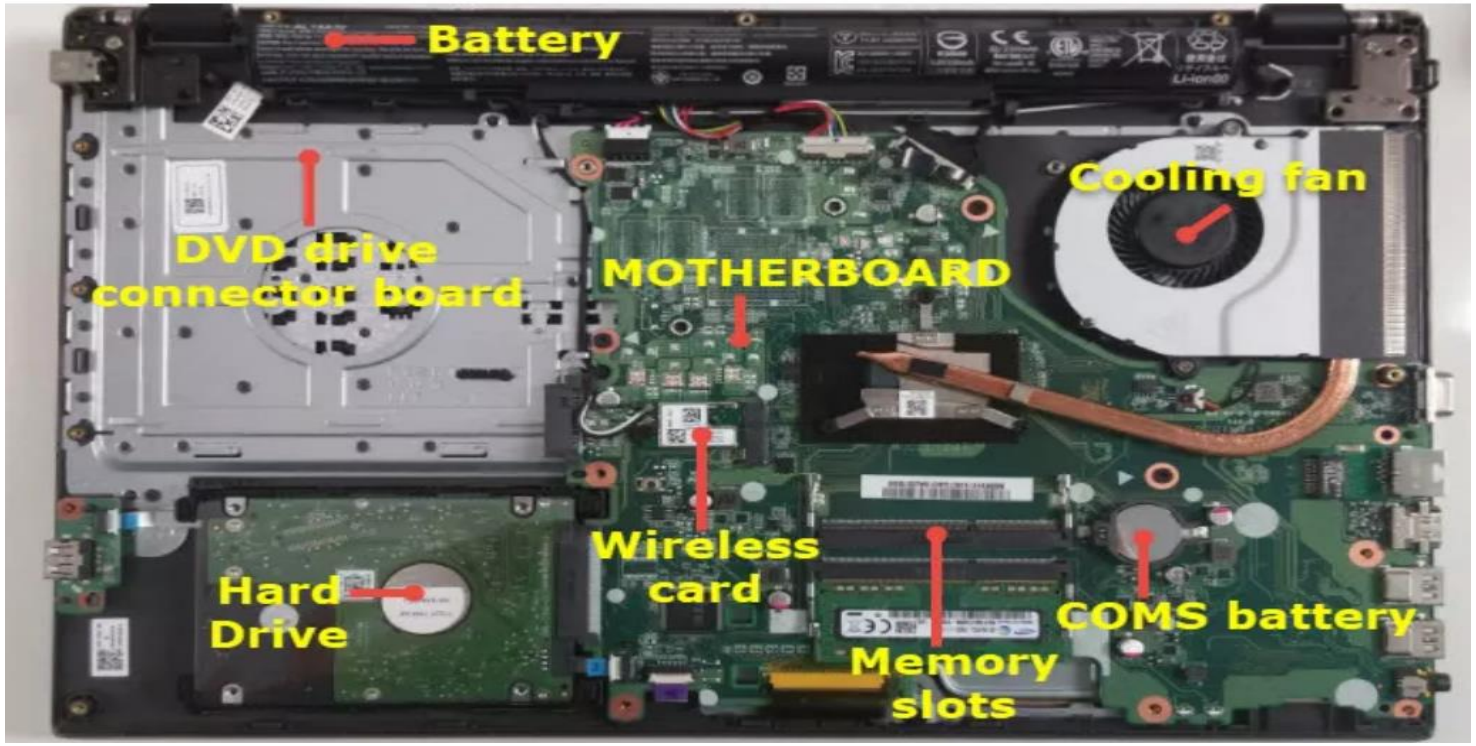
1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy

Abstraction

- Getting only **essential things** and hiding unnecessary details is called as abstraction.
- Abstraction focuses **on outer behavior.**
- In console application when we give **call to function** in to the main function , it represents the abstraction

Encapsulation

- **Binding of data and code** together is called as encapsulation.
- **Implementation of abstraction is called encapsulation.**
- Encapsulation always describe inner behavior of object.



Modularity

- Dividing programs into **small modules** for the purpose of **simplicity** is called modularity.
- **Loosely coupled** modules.
- Advantages are **Ease of Use, reusability and Ease of Maintenance.**
- In programming modularity is achieve using library files.

Hierarchy

- Hierarchy is ranking or ordering of abstractions.
- Main purpose of hierarchy is to achieve re-usability.

- **Types of Hierarchy**
  - Has-a
  - Is-a
  - Use-a
  - Creates-a

- **Example:**

Car → Has-a → Engine

Hierarchy

Person

General Term

Is-a

Police Man

SpecificTerm

Minor pillars of oops

1. **Polymorphism (Typing)**
2. **Concurrency**
3. **Persistence**

1. Polymorphism (Typing):

- One interface having multiple forms is called as polymorphism
- It is having 2 types
  - Compile Time Polymorphism(Function Overloading)
  - Run time Polymorphism(Function Overriding)

2. Concurrency:

- Process of executing multiple tasks simultaneously.
- It’s job is to use resources effectively.

Minor pillars of oops

- **Persistence:**
  - To maintain state of object on secondary storage.
  - By using file handling and database programming we achieve persistence.

Data Types In C++

- It describes three things
  1. **Memory:** How much memory is required to store the data(Size).
  2. **Nature:** Which kind/type of data memory is going to contain.
  3. **Operation:** Which operations can be performed on the data**.**

Different data types:

Fundamental Data Types	Derived Data Types	User Defined Data Types
void(Not Mentioned)	Array	enum
bool(1 byte)	Function	union
char(1 byte)	Pointer	structure
wchar_t(2 bytes)	Reference	class
int(4 bytes)		
float(4 bytes)		
double(8 bytes)		

Data Types in C++

- **bool:**
  - A boolean data type is declared with the bool keyword and can only take the values true or false.
  - When the value is returned, true = 1 and false = 0.
- **Wchar_t:**
  - Wide char is **similar to char** data type, except that wide char **take up twice the space** and can take on much **larger values** as a result.
  - char can take 256 values which corresponds to entries in the ASCII table.
  - Wide char can take on **65536 values** which corresponds to **UNICODE** values.
  - Mostly the wchar_t datatype is used when international languages like Japanese are used.
  - This data type occupies 2 or 4 bytes depending on the compiler being used.
- **Reference:**
  - When a variable is declared as a reference, it becomes an alternative name for an existing variable.
  - A variable can be declared as a reference by putting '&' in the declaration.

Data Types in C++

- **Structure:-**
  - A structure is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.
  - **Data Member**: These members are normal C++ variables. We can create a structure with variables of different data types in C++.
  - **Member Functions**: These members are normal C++ functions. Along with variables, we can also include functions inside a structure declaration.

Access Specifier

- By default all members in structure are accessible everywhere in the program by dot(.) or arrow(→) operators.
- But such access can be restricted by applying access specifiers
  - **private**: Accessible only within the struct
  - **public**: Accessible within & outside struct

Example:

```
struct time {
    private:
            int hr, min, sec;
    public:
            void display()
            {
                    printf("%d:%d:%d",this→hr, this→min, this→sec);
            }
    };
```

Structure in C & C++

struct in c	struct in c ++
we can include only variables into the structure.	we can include the variables as well as the functions in structure.
We need to pass a structure variable by value or by address to the functions.	We don't pass the structure variable to the functions to accept it / display it.The functions inside the struct are called with the variable and DOT operator.
By default all the variables of structure are accessible outside the structure. (using structure variable name)	By default all the members are accessible outside the structure, but we can restrict their access by applying the keywords private /public/ protected.
struct Time t1;	struct Time t1;
AcceptTime(struct Time &t1);	t1.AcceptTime(); //function call

Structure in C & C++

```
struct time {
   int hr, min, sec;
};
void display( struct time *p) {
   printf("%d:%d:%d", p→hr,
   p→min, p→sec);
}
struct time t;
display(&t);
```

```
struct time {
   int hr, min, sec;
void display(){
   printf("%d:%d:%d",
this→hr, this→min, this→sec);
}
};
time t;
t.display();
```

OOP and POP

OOP (Object Oriented Programming)	**POP** (Procedural Oriented Programming)
Emphasis on data of the program	Emphasis on steps or algorithm
OOP follows bottom up approach.	OOP follows top down approach.
A program is divided to objects and their interactions. Programs are divide into small data units i.e. classes	A program is divided into funtions and they interacts. Programs are divided into small code units i.e. functions
Objects communicates with each other by passing messeges.	Functions communicate with each other by passing parameters.
Inheritance is supported.	Inheritance is not supported.
Access control is supported via access modifiers. (private/ public/ protected)	No access modifiers are supported.
Encapsulation is used to hide data.	No data hiding present. Data is globally accessible.
C++, Java	C , Pascal
It overloads functions, constructors, and operators.	Neither it overload functions nor operators
Classes or function can become a friend of another class with the keyword "friend". Note: "**friend**" keyword is used only in c++	No concept of friend function.
Concept of virtual function appear during inheritance.	No concept of virtual classes .

Thank You