C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Oops (Object Oriented Programming Concepts) with C++

- Introduction to Object Oriented Programming
- C++ Introduction
- Namespaces
- Functions
- Class and Object
- Exception Handling
- Dynamic Memory Allocation
- Inheritance
- Virtual Function

Limitations of C Programming

- 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.

Classification of high level Languages

- Procedure Oriented Programming language(POP)
  - ALGOL, FORTRAN, PASCAL, BASIC, C etc.
  - "FORTRAN" is considered as first high level POP language.
  - All POP languages follows "TOP Down" approach

- Object oriented programming languages(OOP)
  - Simula, Smalltalk, C++, Java, C#, Python, Go
  - "Simula" is considered as first high level OOP language.
  - more than 2000 lang. are OO.
  - All OOP languages follows "Bottom UP" approach

Few Real Time Applications of C++

- Games
- GUI Based Application (Adobe)
- Database Software (MySQL Server)
- OS (Apple OS)
- Browser(Mozilla)
- Google Applications(Google File System and Chrome browser)
- Banking Applications
- Compilers
- Embedded Systems(smart watches, MP3 players, GPS systems)

Characteristics of Language

1. It has own syntax
2. It has its own rule(semantics)
3. It contain tokens:
   - Identifier
   - Keyword
   - Constant/literal
   - Operator
   - Seperator / punctuators
4. It contains built in features.
5. We use language to develop application(CUI, GUI, Library)

History of C++

- Inventor of C++ is Bjarne Stroustrup.
- C++ is derived from C and simula.
- Its initial name was "C With Classes".
- At is developed in "AT&T Bell Lab" in 1979.
- It is developed on Unix Operating System.
- In 1983 ANSI renamed "C With Classes" to C++.
- C++ is objet oriented programming language

OOPS(Object Oriented Programming Language)

- OOPS is not a syntax.
- It is a process / programming methodology which is used to solve real world problems.
- 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

- **Abstraction**
  - getting only essential things and hiding unnecessary details is called as abstraction.
  - Abstraction always describe outer behavior of object.
  - 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
  - Function call is abstraction
  - Function definition is encapsulation.
  - Information hiding
    - Data : unprocessed raw material is called as data.
    - Process data is called as information.
    - Hiding information from user is called information hiding.
    - In c++ we used access Specifier to provide information hiding.
- **Modularity**
  - Dividing programs into small modules for the purpose of simplicity is called modularity.
- **Hierarchy (Inheritance [is-a] , Composition [has-a] , Aggregation[has-a], Dependancy)**
  - Hierarchy is ranking or ordering of abstractions.
  - Main purpose of hierarchy is to achieve re-usability.

Minor pillars of oops

- **Polymorphism (Typing)**
  - One interface having multiple forms is called as polymorphism.
  - Polymorphism have two types
    1. **Compile time polymorphism**

       when the call to the function resolved at compile time it is called as compile time polymorphism. And it is achieved by using function overloading and operator overloading
    2. **Runtime polymorphism.**

       when the call to the function resolved at run time it is called as run time polymorphism. And it is achieved by using function overriding.
  - Compile time / Static polymorphism / Static binding / Early binding / Weak typing / False Polymorphism
  - Run time / Dynamic polymorphism / Dynamic binding / Late binding / Strong typing / True polymorphism
- Concurrency
  - The concurrency problem arises when multiple threads simultaneously access same object.
  - You need to take care of object synchronization when concurrency is introduced in the system.
- Persistence
  - It is property by which object maintains its state across time and space.
  - It talks about concept of serialization and also about transferring object across network.

Main Function

- main should be entry point function of C/C++
- Calling/invoking main function is responsibility of operating system.
- Hence it is also called as Callback function

Data Types in C++

- It describes 3 things about variable / object

1. Memory : How much memory is required to store the data.
2. Nature : Which type of data memory can store
3. Operation : Which operations are allowed to perform on data stored inside memory.

- Fundamental Data Types
  - (void, int,char,float,double)
- Derived Data Types
  - (Array, Function, Pointer, Union ,Structure)

Two more additional data types that c++ supports are

1. ***bool* :-** it can take *true* or *false* value. It takes one byte in memory.
2. **wchar_t :-** it can store 16 bit character. It takes 2 bytes in memory.

Bool and wchar_t

- **e.g: bool val=true;**
- **wchar_t**: Wide Character. This should be avoided because its size is implementation defined and not reliable.
- 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. On the other hand, wide char can take on 65536 values which corresponds to UNICODE values which is a recent international standard which allows for the encoding of characters for virtually all languages and commonly used symbols.

- The type for character constants is char, the type for wide character is wchar_t.
- This data type occupies 2 or 4 bytes depending on the compiler being used.
- 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.
- L is the prefix for wide character literals and wide-character string literals which tells the compiler that that the char or string is of type wide-char.
- w is prefixed in operations like scanning (**wcin)** or printing (**wcout)** while operating wide-char type.

Structure

- Structure is a collection of similar or dissimilar data. It is used to bind logically related data into a single unit.
- This data can be modified b any function to which the structure is passed.
- Thus there is no security provided for the data within a structure.
- This concept is modified by C++ to bind data as well as functions.

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

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 .

Namespace

- To prevent name conflicts/ collision / ambiguity in large projects
- to group/orgaize functionally equivalent / related types toghther.
- If we want to access value of global variable then we should use scope resolution operator (::)
- We can not instantiate namespace.
- It is designed to avoid name ambiguity and grouping related types.
- If we want to define namespace then we should use **namespace** keyword.
- We can not define namespace inside function/class.
- If name of the namespaces are same then name of members must be different.
- We can not define main function inside namespace.
- Namespace can contain:
  1. Variable
  2. Function
  3. Types[structure/union/class]
  4. Enum
  5. Nested Namespace

Note :
•If we define member without namespace then it is considered as member of global namespace.
•If we want to access members of namespace frequently then we should use using directive.

Scope Resolution Operator (: :)

- :: operator is used to bind a member with some class or namespace.
- It can be used to define members outside class.
- Also used to resolve ambiguity.
- It can also be used to access global members.
  - Example :- ::a =10; access global var.
- Scope resolution Operator is used to :
  - to call global functions
  - to define member functions of class outside the class
  - to access members of namespaces

cin and cout

- C++ provides an easier way for input and output.
- Console Output : Monitor
  - iostream is the standard header file of C++ for using cin and cout.
  - cout is external object of ostream class.
  - cout is member of std namespace and std namespace is declared in iostream header file.
  - cout uses insertion operator(<<)
- Console Input : Keyborad
  - cin is an external object of istream class.
  - cin is a member of std namespace and std namespace is declared in header file.
  - cin uses Extraction operator(>>)
- The output:
  - cout << “Hello C++”;
- The input:
  - cin >> var;

Functions / User Defined Functions

- It is a set of instructions written to gather as a block to complete specific functionality.
- Function can be reused.
- It is a subprogram written to reduce complexity of source code
- Function may or may not return value.
- Function may or may not take argument
- Function can return only one value at time
- Function is building block of good top-down, structured code function as a "black box“
- **Writing function helps to**
  - improve readability of source code
  - helps to reuse code
  - reduces complexity
- **Types of Functions**
  - Library Functions
  - User Defined Functions

User Defined Functions

- **Function declaration / Prototype / Function Signature**

  <return type> <functionName> ([<arg type>...]);

- **Function Definition**

  <return type> < functionName > ([<arg type> <identifier>...])

  {

  //function body

  }

- **Function Call**

  <location> = < functionName >(<arg value/address>);

Function Overloading

- Functions with same name and different signature are called as overloaded functions.
- Return type is not considered for function overloading.
- Function call is resolved according to types of arguments passed.
- Function overloading is possible due to name mangling done by the C++ compiler (Name mangling process , mangled name)
- Differ in number of input arguments
- Differ in data type of input arguments
- Differ at least in the sequence of the input arguments

- Example :
  - int sum(int a, int b) { return a+b; }
  - float sum(float a, float b) { return a+b; }
  - int sum(int a, int b, int c) { return a+b+c;;

Thank You