C++ Programming

Trainer : Akshita Chanchlani

Email: akshita.chanchlani@sunbeaminfo.com

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)

History of C++

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

  - 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)

- 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.

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
    - 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
    1. **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
- Persistence

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.

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

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 (::)
- 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

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;

Class

- Building block that binds together data & code.
- Program is divided into different classes
- Class is collection of data member and member function.
- Class represents set/group of such objects which is having common structure and common behavior.
- Class is logical entity.
- Class has
  - Variables (data members)
  - Functions (member functions or methods)
- By default class members are private(not accessible outside class scope)
- Classes are stand-alone components & can be distributed in form of libraries
- Class is blue-print of an object

Data Members and Member Functions

Data Members

- Data members of the class are generally made as private to provide the data security.
- The private members cannot be accessed outside the class.
- So these members are always accessed by the member functions.

Member Functions

- Member functions are generally declared as public members of class.
- Constructor : Initialize Object
- Destructor : De-initialize Object
- Mutators : Modifies state of the object
- Inspectors : Don’t Modify state of object
- Facilitator : Provide facility like IO

Object

- Object is an instance of class.
- Entity that has physical existence, can store data, send and receive message to communicate with other objects.
- An entity, which get space inside memory is called object.
- Object is used to access data members and member function of the class
- Process of creating object from a class is called instantiation
- **Object has**
  - Data members (***state*** of object)
    - Value stored inside object is called state of the object.
    - Value of data member represent state of the object.
  - Member function (***behavior*** of object)
    - Set of operation that we perform on object is called behaviour of an object.
    - Member function of class represent behaviour of the object.
    - is how object acts & reacts, when its state is changed & operations are done
    - Operations performed are also known as messages
  - Unique address(***identity*** of object)

this pointer

- To process state of the object we should call member function on object. Hence we must define member function inside class.
- If we call member function on object then compiler implicitly pass address of that object as a argument to the function implicitly.
- To store address of object compiler implicitely declare one pointer as a parameter inside member function. Such parameter is called this pointer.
- this is a keyword. "this" pointer is a constant pointer.
- this is used to store address of current object or calling object.
- The invoking object is passed as implicit argument to the function.
- *this* pointer points to current object i.e. object invoking the member function.
- Thus every member function receives *this* pointer.
- Following functions do not get this pointer:
  1. Global Function
  2. Static Member function
  3. Friend Function.

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

Inline Function

- C++ provides a keyword *inline* that makes the function as inline function.
- Inline functions get replaced by compiler at its call statement. It ensures faster execution of function just like macros.
- Advantage of inline functions over macros: inline functions are type-safe.
- Inline is a request made to compiler.
- If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

When to use Inline function?

- We can use Inline function as per our needs.
- We can use the inline function when performance is needed.
- We can use the inline function over macros.
- We prefer to use the inline keyword outside the class with the function definition to hide implementation details of the function.

Default Arguments

- In C++, functions may have arguments with the default values. Passing these arguments while calling a function is optional.
- A default argument is a default value provided for a function parameter/argument.
- If the user does not supply an explicit argument for a parameter with a default argument, the default value will be used.
- If such argument is not passed, then its default value is considered. Otherwise arguments are treated as normal arguments.
- Default arguments should be given in right to left order.
  - int sum (int a, int b, int c=0, int d=0) {

```
        return a + b + c + d;
}
```

  - The above function may be called as
    - Res=sum(10,20);
    - Res=sum(10,20,40);
    - Res=sum(10,30,40,50);

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;;

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

Thank You