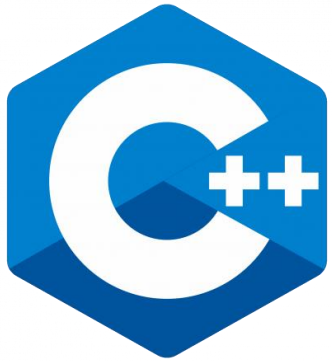

C++ Programming

Trainer : Pradnyaa S Dindorkar

Email: pradnya@sunbeaminfo.com

we did -

1. Struct in C and Cpp
2. Class and object (data member, member functions)
3. Live examples of class and object
4. this pointer
5. Types of Member Functions within class
6. Constructor
7. Destructor

Todays Topics

1. Destructor
2. Mutators / setter
3. Inspector / getter
4. facilitator
5. namespace
6. cin and cout
7. complex class
8. Modular Approach
9. Constant
10. references
11. Difference between Pointers and reference
12. Sum function and Copy Constructor

Destructor

- It is a member function of a class which is used to release the resources.
- It is considered as special function of the class
  - Its name is same as class name and always preceds with tilde operator(~)
  - It doesn't have return type and doesn't take parameter.
  - It is designed to call implicitly.
- Destructor calling sequence is exactly opposite of constructor calling sequence.
- Destructor is designed to call implicitly.
- If we do not define destructor inside class then compiler generates default destructor for the class.
- Default destructor do not release resources allocated by the programmer. If we want to release it then we should define destructor inside class.

Other Member functions of class

- Mutators / setter : modify state of object
- inspector/getter : read the data member but do not change the state of the object
- Facilitator : Provide extra facility to work with object

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

Namespace

- To prevent name conflicts/ collision / ambiguity in large projects
- to group/ organize functionally equivalent / related types together.
- 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.
- 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.

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 : Keyboard
  - cin is an external object of istream class.
  - cin is a member of std namespace and std namespace is declared in iostream header file.
  - cin uses Extraction operator(>>)
- The output:
  - cout << “Hello C++”;
- The input:
  - cin >> var;

Complex class :- Ex = 5+j7

Data member = real , imaginary

member functions = complex()
complex(int r,int i)
acceptComplexNumber()
printComplexNumber()
~complex()

Example Scope Resolution

```
class complex {
int real, imag;
public: complex();
void show();
};
```

complex.h

```
complex::complex() {
        real = imag = 0;
}
void complex::show() {
        cout<<real<<"+j"<<imag;
}
```

complex.cpp

```
main()
{
        complex obj;
        obj.show();
}
```

Program.cpp

Modular Approach

- "/usr/include" directory is called standard directory for header files.
- It contains all the standard header files of C/C++
- If we include header file in angular bracket (e.g #include<filename.h>) then preprocessor try to locate and load header file from standard directory only(/usr/include).
- If we include header file in double quotes (e.g #include“filename.h") then preprocessor try to locate and load header file first from current project directory if not found then it try to locate and load from standard directory.

Constant in C++

- We can declare a constant variable that cannot be modified in the app.
- If we do not want to modify value of the variable then const keyword is used.
- constant variable is also called as read only variable.
- The value of such variable should be known at compile time.
- In C++ , Initializing constant variable is mandatory
- const int i=3; //VALID
- Const int val; //Not ok in c++
- Generally const keyword is used with the argument of function to ensure that the variable cannot be modified within that function.

Constant data member

- Once initialized, if we do not want to modify state of the data member inside any member function of the class including constructor body then we should declare data member constant.
- If we declare data member constant then it is mandatory to initialize it using constructors member initializer list.

```
class Test
{
private:
        const int num1;
public:
        Test( void ) : num1( 10 ) //OK
        {
                //this->num1 = 10; //Not OK
        }
};
```

Const member function

- The member function can declared as const. In that case object invoking the function cannot be modified within that member function.
- We can not declare global function constant but we can declare member function constant.
- If we do not want to modify state of current object inside member function then we should declare member function as constant.
- void display() const; , void printData() const;
- Even though normal members cannot be modified in const function, but *mutable* data members are allowed to modify.
- In constant member function, if we want to modify state of non constant data member then we should use **mutable keyword.**
- We can not declare following function constant:
  1. Global Function
  2. Static Member Function
  3. Constructor
  4. Destructor

Const object

- If we don’t want to modify state of the object then instead of declaring data member constant, we should declare object constant.
- On non constant object, we can call constant as well as non constant member function.
- On Constant object, we can call only constant member function.

Thank You