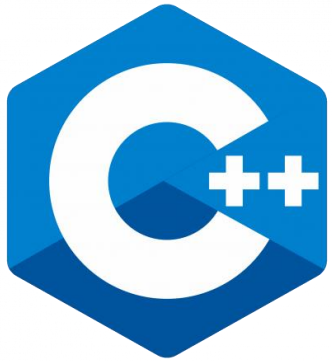

C++ Programming

Trainer : Pradnyaa S Dindorkar

Email: pradnya@sunbeaminfo.com

we did -

1. access specifiers
2. this pointer
3. Class and object (data member, member functions)
4. Live examples of class and object
5. Types of Member Functions within class
6. Constructor

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

Types of Constructor

- Parameterless constructor
  - also called zero argument constructor or user defined default constructor
  - If we create object without passing argument then parameterless constructor gets called
  - Constructor do not take any parameter
- Parameterized constructor
  - If constructor take parameter then it is called parameterized constructor
  - If we create object, by passing argument then paramterized constructor gets called
- Default constructor
  - If we do not define constructor inside class then compiler generates default constructor for the class.
  - Compiler generated default constructor is parameterless.

Constructor's member initializer list

- If we want to initialize data members according to users requirement then we should use constructor body.

```
class Test
{
private:
        int num1;
        int num2;
        int num3;
public:
        Test( void )
        {
        this->num1 = 10;
        this->num2 = 20;
        this->num3 = num2;
        }
};
```

•If we want to initialize data member according to order of data member declaration then we can use constructors member initializer list.

```
class Test
{
private:
        int num1;
        int num2;
        int num3;
public:
        Test( void ) : num1( 10 ), num2( 20 ), num3( num2)
        { }
};
```

Except array we can initialize any member inside constructors member initializer list.

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 or 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 : do not change the state of the object**
- **facilitator**

Scope

- It decides area / region / boundary in which we can access the element.
- **Types of scope in C++:**
  1. **Block scope**
  2. **Function scope**
  3. **Prototype scope**
  4. **Class scope**
  5. **Namespace scope**
  6. **File scope**
  7. **Program scope**

Example Scope

```
int num6;          //Program Scope
```

```
static int num5;  //File Scope
```

```
int main( void )
{
        int num1 = 10; //Function Scope
        while( true )
        {
                int temp = 0; //Block Scope
        } ;
        return 0;
}
```

```
namespace ntest
{
        int num4;   //Namespace scope
        class Test
        {
                int num3;   //Class Scope
        };
}
```

```
void sum( int num1, int num2 ); //Prototype scope
```

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.
- 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.

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;

Complex class :- Ex = 5+j7

Data member = real , imaginary

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

Thank You