**Scope of Class**

A class can have either:

1. Local scope (local class)
2. Global scope (global class)

Local Class: A class declared inside a function is known as a local class in C++ as it is local to that function.

An example of a local class is given as follows.

#include<iostream.h>

void func() //some function

{

   class LocalClass //local class

{...

   };

}

int main()

{

   return 0;

}

Local classes must abide by the following rules:

1. A local class name can only be used in its function and not outside it i.e. the objects of such a class are local to the function scope.
2. The methods of a local class must be defined inside the local class only and not outside it.
3. A local class cannot have static data members but it can have static functions.
4. Local classes can access global types, variables and functions.
5. Local classes can access other local classes of same function.

A program that demonstrates a local class in C++ is given as follows.

#include<iostream.h>

void func()

{

   class LocalClass

{

      private:

      int num;

      public:

      void getdata(int n)

{

          num = n;

      }

      void putdata()

{

         cout<<"The number is "<<num;

      }

   };

   LocalClass obj;

   obj.getdata(7);

   obj.putdata();

}

int main()

{

   cout<<"Demonstration of a local class"<<endl;

   func();

   return 0;

}

O/P:

Demonstration of a local class

The number is 7

Global Class: A class defined outside all methods is a global class because its objects can be created from anywhere in the program and its members can be accessed from anywhere, i.e. either inside or outside the class.

A program that demonstrates a global class in C++ is given as follows.

#include<iostream.h>

class GlobalClass //local class

{...

}

GlobalClass g1; //local object

;

GlobalClass g2; //global object

}

int main()

{

GlobalClass g3; //local object

   return 0;

}

Example for static functions and static data members in Local classes:

#include <iostream>

using namespace std;

void fun()

{

class Test // local to fun

{

//static int a; //NOT ALLOWED!

public:

static void method()

{

cout << "Local Class method() called";

}

};

Test::method();

}

int main()

{

fun();

return 0;

}

O/P:

Local Class method() called

Lab assignment:

1. Explore each of the statements stated below and give an appropriate example for the each with regard to Local classes:
2. A local class name can only be used in its function and not outside it i.e. the objects of such a class are local to the function scope.
3. The methods of a local class must be defined inside the local class only and not outside it.
4. A local class cannot have static data members but it can have static functions.
5. Local classes can access global types, variables and functions.
6. Local classes can access other local classes of same function.
7. Explore and write an appropriate example for the fact that “Global class members can be accessed from anywhere”.

Ques: Can a local class have a global object?

**Scope Resolution Operator**

Scope Resolution Operator: is basically an operator used in order to resolute the scope. Thus, Scope resolution operator is used to get the hidden names due to variable scopes so that you can still use them.

In C++, the scope resolution operator can be used as both unary ( : ) and binary ( :: )

Scope resolution operator in C++ can be used for:

* Accessing a global variable when there is a local variable with same name
* Defining a function outside a class
* Accessing static member function and static variables
* In case of multiple Inheritance
* Namespace

Accessing a global variable when there is a local variable with same name:

|  |  |
| --- | --- |
|  | #include<iostream.h>    int num = 30;  // Initializing a global variable num  int main()  {  int num = 10; // Initializing the local variable num  cout << "Value of global num is " << ::num;  cout << "nValue of local num is " << num;  return 0;  } |

Defining a function outside a class:

|  |  |
| --- | --- |
|  | #include<iostream.h>  using namespace std;  class Bike  {  public:  // Just the Function Declaration  void Speed();  };  // Defining the Speed function outside Bike class using ::  void Bike::Speed()  {  cout << "Speed of Bike is 90 KMPH";  }  int main()  {  Bike bike;  bike.Speed();  return 0;  } |

Accessing static member function and static variables:

#include <iostream>

**using** **namespace** std;

**class** Demo

{

**private**:

//static data members

**static** **int** X;

**static** **int** Y;

**public**:

//static member function

**static** **void** Print()

{

cout <<"Value of X: " << X << endl;

cout <<"Value of Y: " << Y << endl;

}

};

//static data members initializations

**int** Demo :: X =10;

**int** Demo :: Y =20;

**int** main()

{

Demo OB;

//accessing class name with object name

cout<<"Printing through object name:"<<endl;

OB.Print();

//accessing class name with class name

cout<<"Printing through class name:"<<endl;

Demo::Print();

**return** 0;

}

O/P:

Printing through object name:

Value of X: 10

Value of Y: 20

Printing through class name:

Value of X: 10

Value of Y: 20

**NOTE: According to the rule of static in C++, only static member function can access static data members. Non-static data member can never be accessed through static member functions.**

In case of multiple Inheritance

If you have two parent classes with same variable names and you are inheriting both of them in the child class, then you can use scope resolution operator with the class name to access the individual variables.

**Example**

|  |  |
| --- | --- |
|  | #include<iostream.h>    class Parent1  {  protected:  int num;  public:  Parent1()  { num = 100; }  };  class Parent2  {  protected:  int num;  public:  Parent2()  { num = 200; }  };  class Child : public Parent1, public Parent2  {  public:  void function()  {  cout << "Parent1's num is " << Parent1::num;  cout << "nParent2's num is " << Parent2::num;  }  };  int main()  {  Child obj;  obj.function();  return 0;  } |

**Namespace**

Suppose we have two namespaces & both contains class with same name. So to avoid any conflict we can use namespace name with the scope resolution operator. In the below example we are using ***std::cout***.

**Example**

|  |  |
| --- | --- |
|  | #include<iostream>  int main(){  std::cout << "Hello" << std::endl;  } |

Member Function of a Class

* Member functions are the functions which are the members of a particular class and it works on the data members of the class.
* The definition of member functions can be inside or outside the definition of class.

class className

{

returnType MemberFunction(arguments)

{

//function body

}

. . . .

};

* If the member function is defined inside the class then it can be defined directly (i.e. without declaring the function)

class Cube

{

public:

int side;

int getVolume()

{

return side\*side\*side; //returns volume of cube

}

};

int main()

{

Cube C1;

C1.side = 4; // setting side value

cout<< "Volume of cube C1 = "<< C1.getVolume();

}

* If the member function is defined outside the class, then we have to use the scope resolution

: : operator along with class name while defining the function and it obviously then becomes necessary to declare the function inside the class.

class Cube

{

public:

int side;

int getVolume();

};

// member function defined outside class definition

int Cube :: getVolume()

{

return side\*side\*side;

}

int main()

{

Cube C1;

C1.side = 4; // setting side value

cout<< "Volume of cube C1 = "<< C1.getVolume();

}

Assignment: Explore each of the following member function by yourself:

1. Static functions
2. Const functions
3. Inline functions
4. Friend functions

Access Specifiers

Access Specifier in C++ defines how the members of the class can be accessed. C++ has 3 types of access specifiers:

1. Private Access Specifier
2. Public Access Specifier
3. Protected Access Specifier

A class can have three different sections: public, protected, or private and each section can have its own data members and member functions.

### Syntax of Declaring Access Specifiers in C++

class className

{

private:

// private data members and private member functions

public:

// public data members and private member functions

protected:

// protected data members and private member functions

};

**Note: If no access specifier is specified, then by default, all members and function of a class are private.**

Example:

class A

{

private:

int a;

public:

int b;

void show()

{

a=10;

b=20;

//All the data members of the class can be accessed here

cout<<"\nAccessing variable within the class"<<endl;

cout<<"Value of a: "<<a<<endl;

cout<<"Value of b: "<<b<<endl;

}

};

int main()

{

A obj; // create object

obj.show();

cout<<"\nAccessing variable outside the class"<<endl;

//'a' cannot be accessed as it is private

//cout<<"value of a: "<<obj.a<<endl;

//'b' is public as can be accessed from any where

cout<<"value of b: "<<obj.b<<endl;

return 0;

}