CLASSES AND OBJECTS

CLASS

Class is a group of objects that share common properties and relationships.
In C++, a class is a new data type that contains member variables and member functions that operates on the variables. A class is defined with the keyword class.

It allows the data to be hidden, if necessary from external use. When we defining a class, we are creating a new abstract data type that can be treated like any other built in data type.

Generally a class specification has two parts:-

a) Class declaration

b) Class function definition

Syntax

```
class class_name
{
Access specifier :Variable declarations;
Access specifier : function declarations;
};
```

Access Control

Access specifier or access modifiers are the labels that specify type of access given to members of a class.

These are used for data hiding. These are also called as visibility modes. There are three types of access specifiers

1. private
2. public
3. protected

Example

```
class item
{
int member;
float cost;
public:
void getldata (int a ,float b);
void putdata (void);
}
```

Object

Instance of a class is called object.

Syntax: class_name object_name;

Ex: student s;

Accessing members: dot operator is used to access members of class Object.

name.function-name(actual arguments);

Ex: s.get_data();

s.put_data();

DEFINING MEMBER FUNCTION

Member can be defined in two places

- Outside the class definition
- Inside the class function

INLINE FUNCTIONS:

Definition:

An inline function is a function that is expanded in line when it is invoked. Inline expansion makes a program run faster because the overhead of a function call and return is eliminated.

Necessity of Inline Function:

One of the objectives of using functions in a program is to save some memory space, which becomes appreciable when a function is likely to be called many times.

Every time a function is called, it takes a lot of extra time in executing a series of instructions for tasks such as jumping to the function, saving registers, pushing arguments into the stack, and returning to the calling function.

When a function is small, a substantial percentage of execution time may be spent in such overheads. One solution to this problem is to use macro definitions, known as macros. Preprocessor macros are popular in C. The major drawback with macros is that they are not really functions and therefore, the usual error checking does not occur during compilation.

To eliminate the cost of calls to small functions, C++ proposes a new feature called inline function.

Properties of inline function:

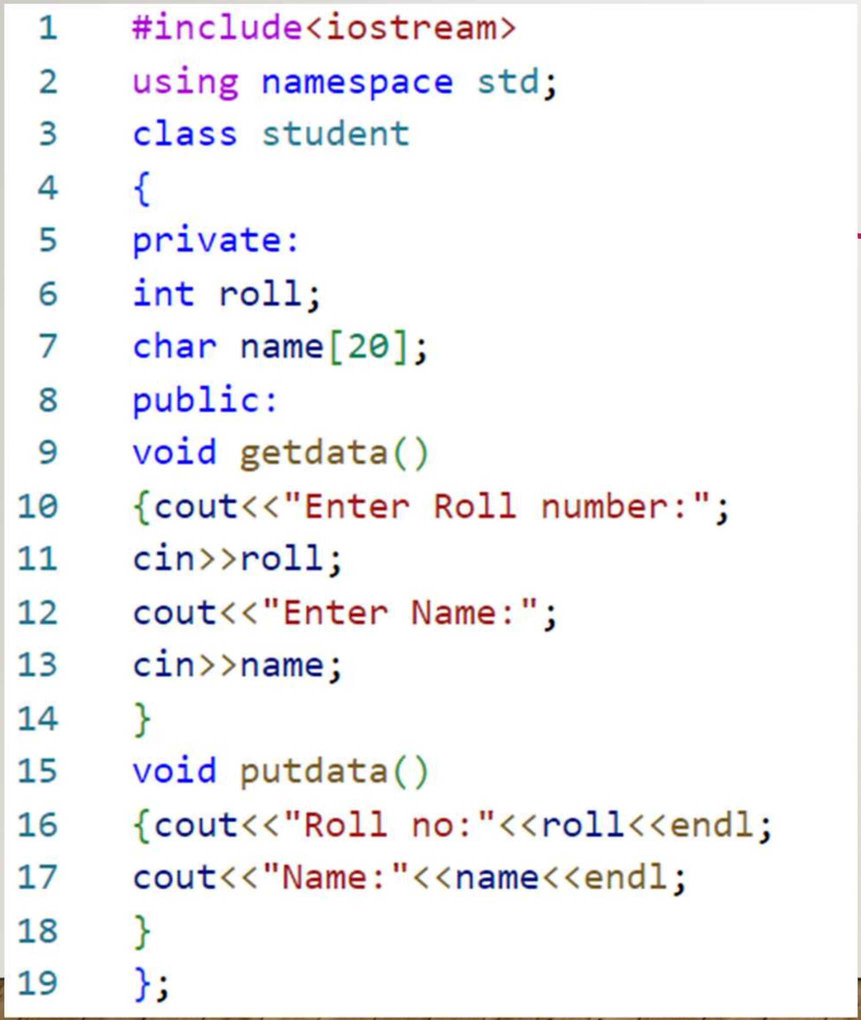
1.Inline function sends request but not a command to compiler

2.Compiler may serve or ignore the request

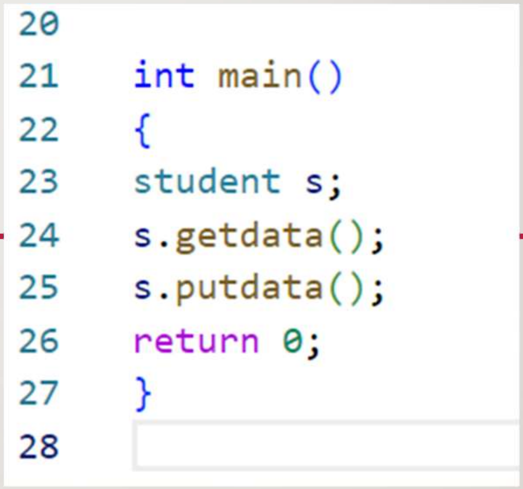
3. If function has too many lines of code or if it has complicated logic then it is executed as normal function.

Situations where inline does not work:

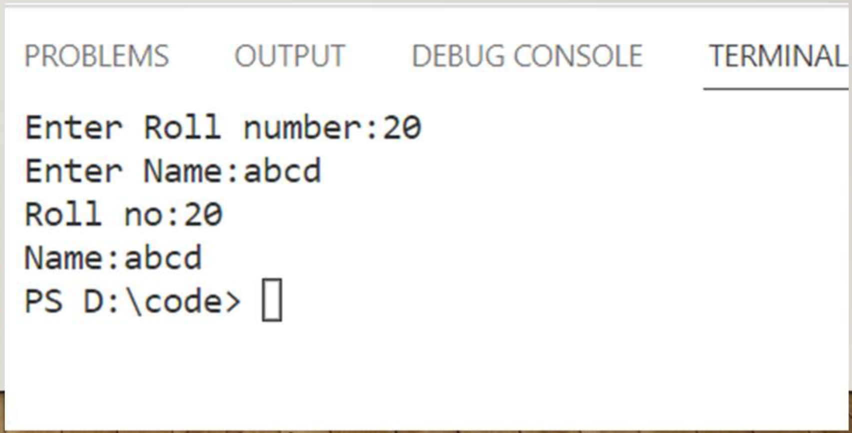
1. A function that is returning value , if it contains switch ,loop or both then it is treated as normal function.
2. If a function is not returning any value and it contains a return statement then it is treated as normal function.
3. If function contains static variables then it is executed as normal function.
4. If the inline function is declared as recursive function then it is executed as normal function.



```cpp
#include<iostream>
using namespace std;
class student
{
private:
int roll;
char name[20];
public:
void getdata()
{cout<<"Enter Roll number:";
cin>>roll;
cout<<"Enter Name:";
cin>>name;
}
void putdata()
{cout<<"Roll no:"<<roll<<endl;
cout<<"Name:"<<name<<endl;
}
};
```



```cpp

int main()
{
student s;
s.getdata();
s.putdata();
return 0;
}

```



```
Enter Roll number:20
Enter Name:abcd
Roll no:20
Name:abcd
PS D:\code> 
```

Scope Resolution operator

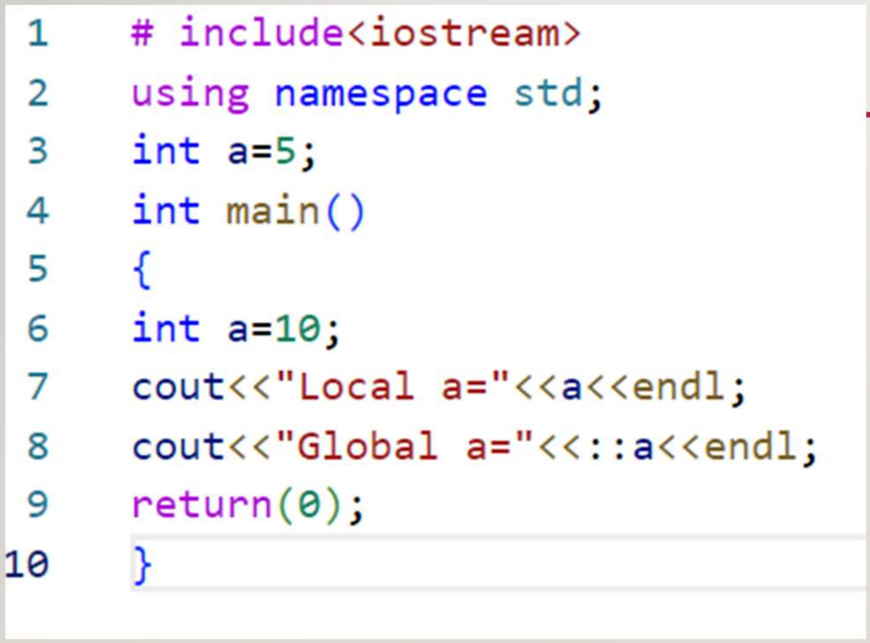
Scope

Visibility or availability of a variable in a program is called as scope. There are two types of scope.

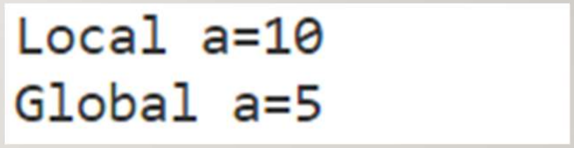
i)Local scope

ii)Global scope

Scope resolution operator "::" is used to access global variables if same variables are declared as local and global.



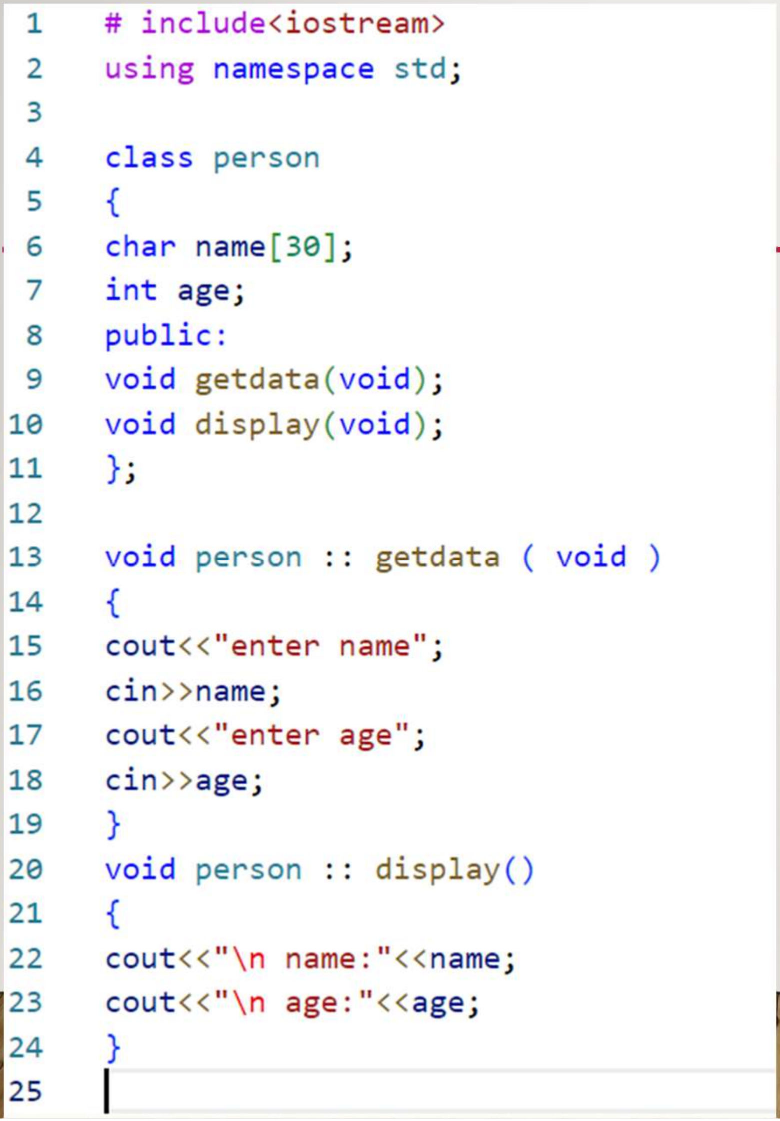
```cpp
# include<iostream>
using namespace std;
int a=5;
int main()
{
int a=10;
cout<<"Local a="<<a<<endl;
cout<<"Global a="<<::a<<endl;
return(0);
}
```



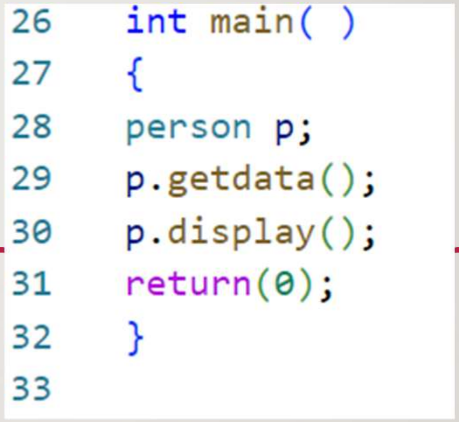
```
Local a=10
Global a=5
```

Class Scope

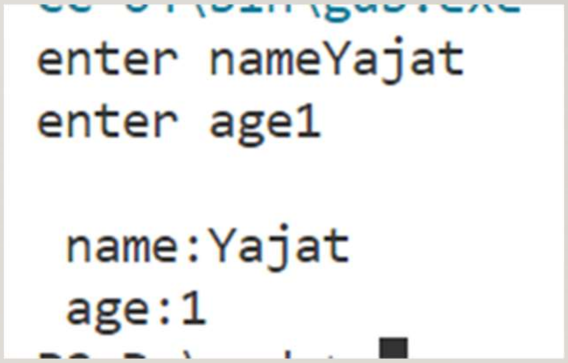
Scope resolution operator(::) is used to define a function outside a class.



```cpp
# include<iostream>
using namespace std;

class person
{
char name[30];
int age;
public:
void getdata(void);
void display(void);
};

void person :: getdata ( void )
{
cout<<"enter name";
cin>>name;
cout<<"enter age";
cin>>age;
}
void person :: display()
{
cout<<"\n name:"<<name;
cout<<"\n age:"<<age;
}

```



```cpp
int main( )
{
person p;
p.getdata();
p.display();
return(0);
}

```



```
enter nameYajat
enter age1

 name:Yajat
 age:1
```

Private member functions

Although it is a normal practice to place all the data items in a private section and all the functions in public, some situations may require contain functions to be hidden from the outside calls.

Tasks such as deleting an account in a customer file or providing increment to and employee are events of serious consequences and therefore the functions handling such tasks should have restricted access. We can place these functions in the private section.

A private member function can only be called by another function that is a member of its class. Even an object can not invoke a private function using the dot operator.

NESTING OF MEMBER FUNCTION

A member function can be called by using its name inside another member function of the same class. This is known as nesting of member functions.

Memory Allocation for Objects

Memory for objects is allocated when they are declared but not when class is defined. All objects in a given class uses same member functions.

The member functions are created and placed in memory only once when they are defined in class definition.

STATIC CLASS MEMBERS

a) Static Data Members

b) Static Member Functions

Static Data Members

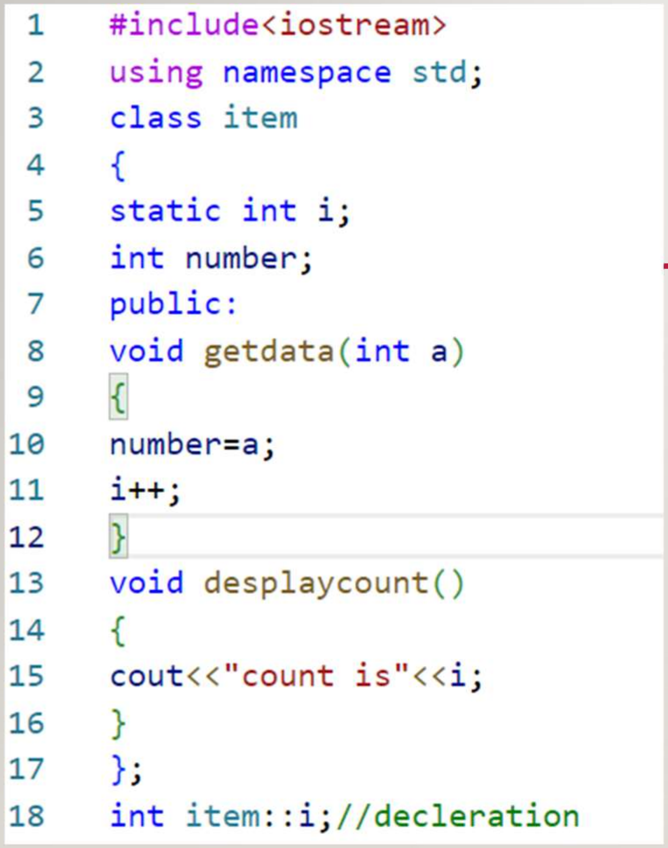
A data member of a class can be qualified as static.

A static member variable has certain special characteristics:

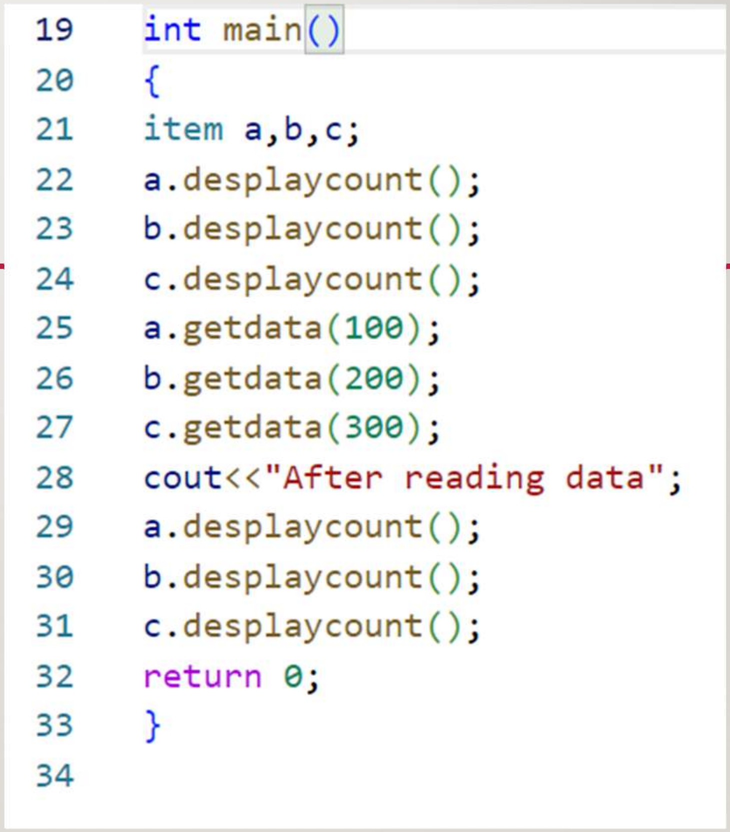
- It is initialized to zero when the first object of its class is created. No other initialization is permitted.
- Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
- It is visible only within the class, but its lifetime is the entire program.
- Static data member is defined by keyword "static".

Syntax:

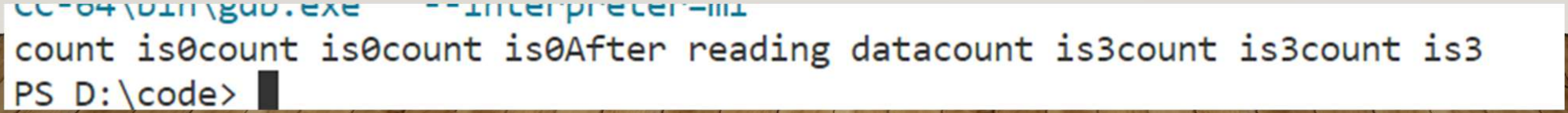
Data type class name::static_variable Name; Ex: int item::count;



```cpp
#include<iostream>
using namespace std;
class item
{
static int i;
int number;
public:
void getdata(int a)
{
number=a;
i++;
}
void desplaycount()
{
cout<<"count is"<<i;
}
};
int item::i;//decleration
```



```cpp
int main()
{
item a,b,c;
a.desplaycount();
b.desplaycount();
c.desplaycount();
a.getdata(100);
b.getdata(200);
c.getdata(300);
cout<<"After reading data";
a.desplaycount();
b.desplaycount();
c.desplaycount();
return 0;
}
```



```
cc-64\bin\gdb.exe   --interpreter=mi
count is0count is0count is0After reading datacount is3count is3count is3
PS D:\code> █
```

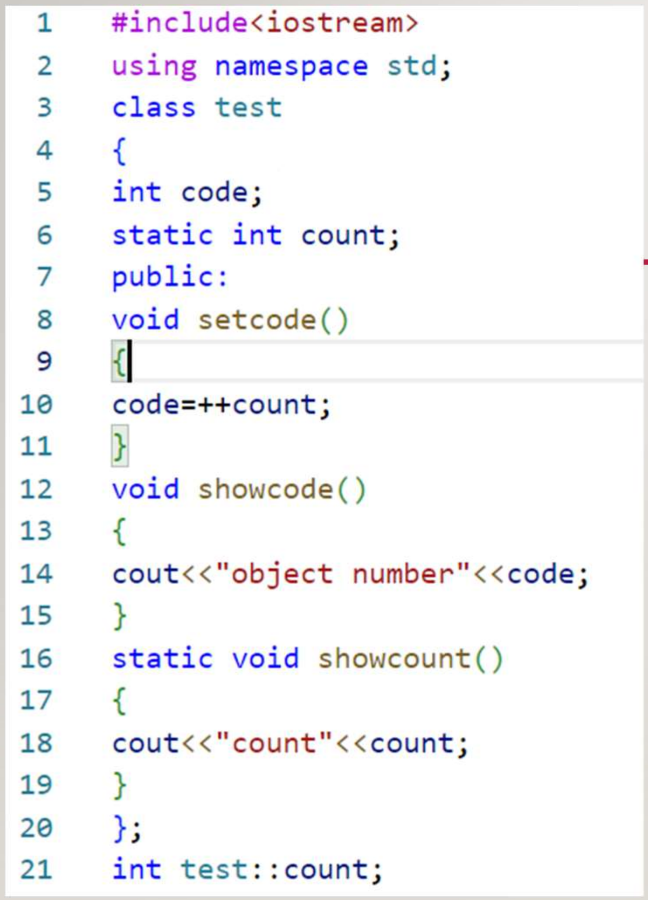
Static Member Functions

Like static member variable, we can also have static member functions.

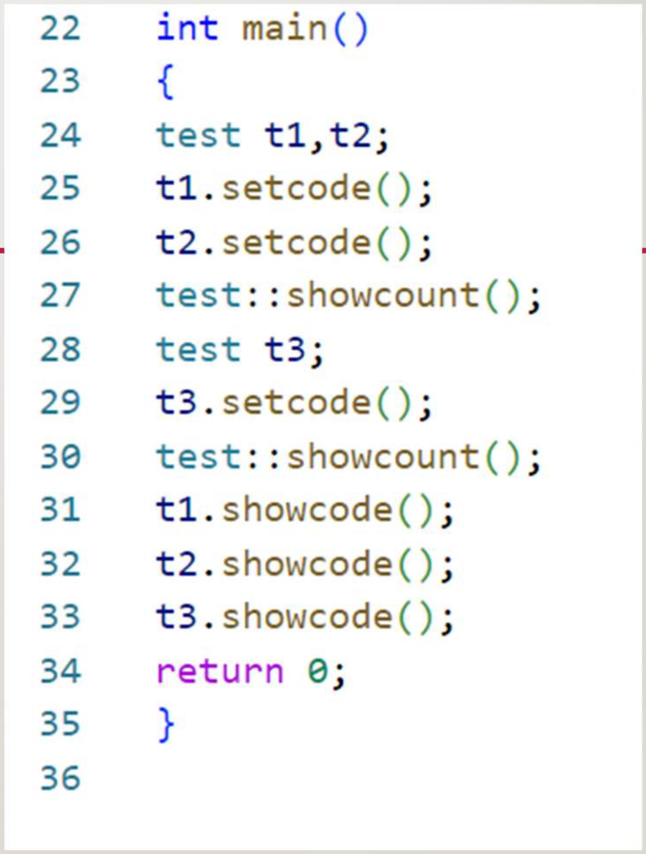
A member function that is declared static has the following properties:

- A static function can have access to only other static members (functions or variables) declared in the same class.
- A static member function is to be called using the class name (instead of its objects) as follows:

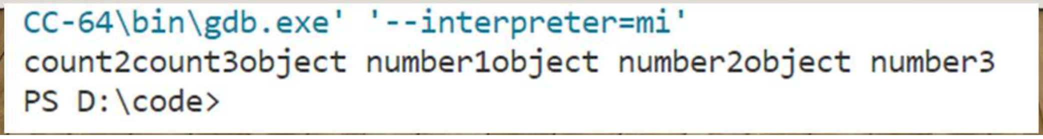
```
class-name :: function-name;
```



```cpp
#include<iostream>
using namespace std;
class test
{
int code;
static int count;
public:
void setcode()
{
code=++count;
}
void showcode()
{
cout<<"object number"<<code;
}
static void showcount()
{
cout<<"count"<<count;
}
};
int test::count;
```



```cpp
int main()
{
test t1,t2;
t1.setcode();
t2.setcode();
test::showcount();
test t3;
t3.setcode();
test::showcount();
t1.showcode();
t2.showcode();
t3.showcode();
return 0;
}
```



```
CC-64\bin\gdb.exe' '--interpreter=mi'
count2count3object number1object number2object number3
PS D:\code>
```

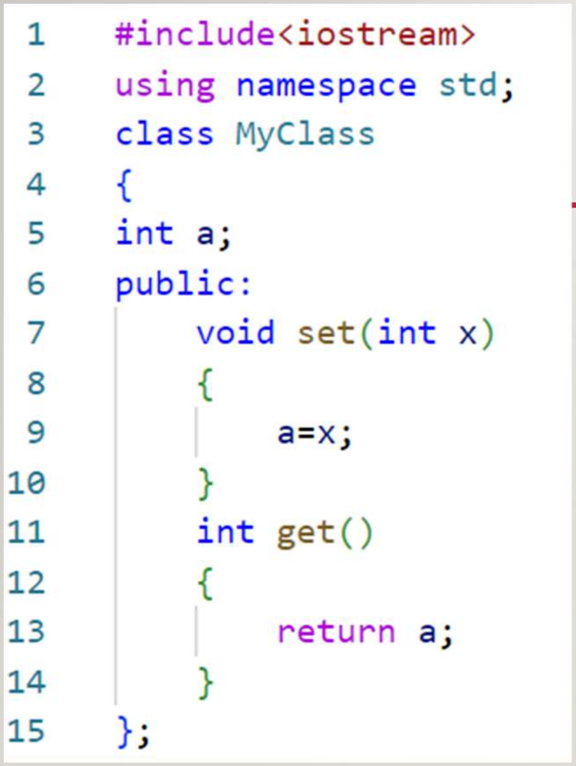
Arrays of Objects

Arrays of variables of type "class" is known as "Array of objects". An array of objects is stored inside the memory in the same way as in an ordinary array.

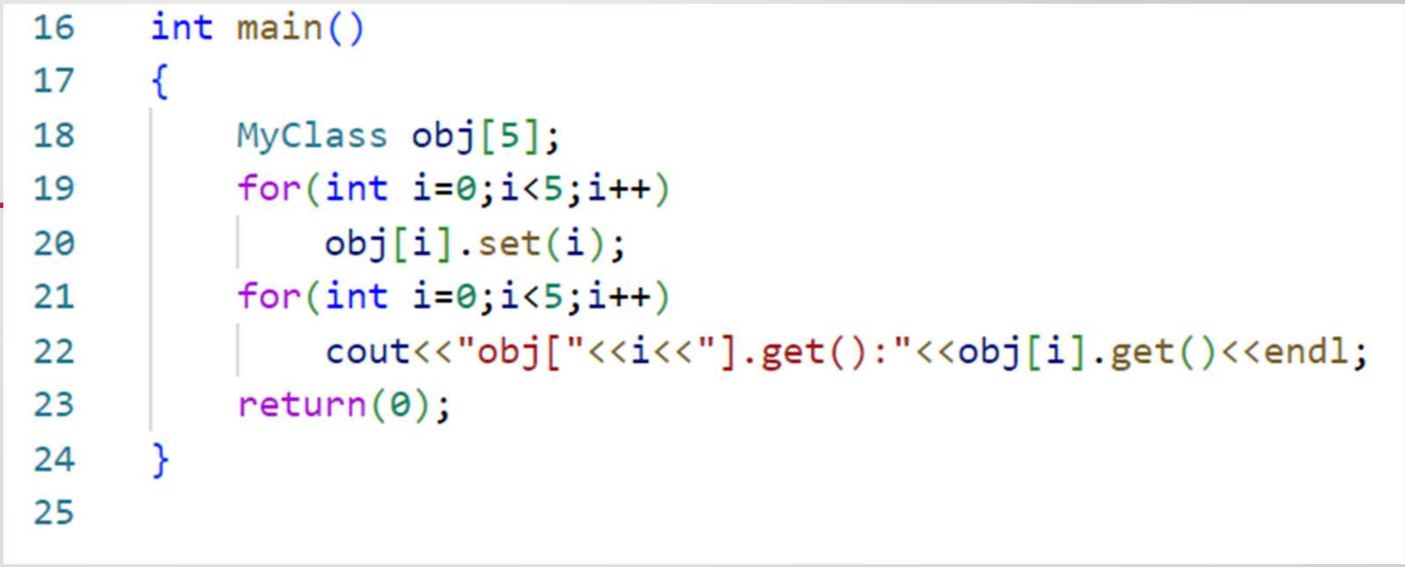
Syntax:

Class_name object_name[size]; //Where size is the size of array

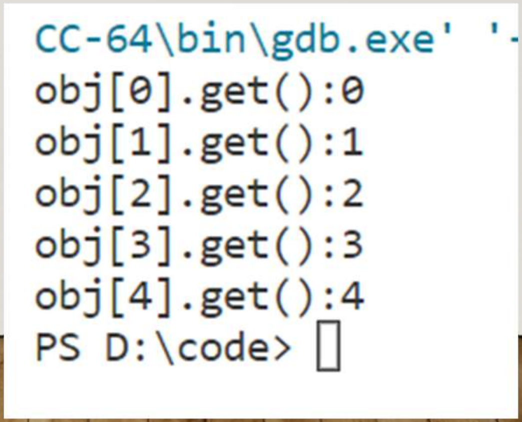
Ex: Myclass obj[10];



```cpp
#include<iostream>
using namespace std;
class MyClass
{
int a;
public:
    void set(int x)
    {
        a=x;
    }
    int get()
    {
        return a;
    }
};
```



```cpp
int main()
{
    MyClass obj[5];
    for(int i=0;i<5;i++)
        obj[i].set(i);
    for(int i=0;i<5;i++)
        cout<<"obj["<<i<<"].get():"<<obj[i].get()<<endl;
    return(0);
}

```



```
CC-64\bin\gdb.exe' '-
obj[0].get():0
obj[1].get():1
obj[2].get():2
obj[3].get():3
obj[4].get():4
PS D:\code> 
```