

Constant vs Non Constant Functions and Static Data Members & Functions

Ali Haider

syedalihaider.ciit@gmail.com

Department of Computer Science IUB

Overview

- Constant Member Function
- Non Constant Function
- Data members and Member Functions Memory Representation
- Static Data Member
- Static Member Functions

Constant Member Function

- The const member functions are the functions which are declared as constant in the program.
- The object called by these functions cannot be modified.
- It is recommended to use const keyword so that accidental changes to object are avoided.
- A const member function can be called by any type of object.
- Non-const functions can be called by non-const objects only.
- The syntax of const member function in C++ language,
- **Return Type** function_name() **const { function body}**

Example

```
#include<iostream>
using namespace std;
class Demo {
    int val;
public:
    Demo(int x = 0) {
        cout << "Constructor is called";
        val = x;
    }
    int getValue() const {
        cout << "get value function called ";
        return val+=10;
    }
};
int main() {
    const Demo d(28);
    Demo d1(8);
    cout << "The value using object d : " << d.getValue();
    cout << "\nThe value using object d1 : " << d1.getValue();
    return 0;
}
```

- By Compiling this code which Error You had Seen??

Non Constant Function

- A const member function can be called by any type of object.
- Non-const functions can be called by non-const objects only.
- In this example the non constant function
- Can not be accessed by constant object
- It will show error
- passing 'const Test' as 'this' argument of 'int
- Test::getValue()' discards qualifiers

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

class Test {
    int value;
public:
    Test(int v = 0) {value = v;}
    int getValue() {return value;}
};

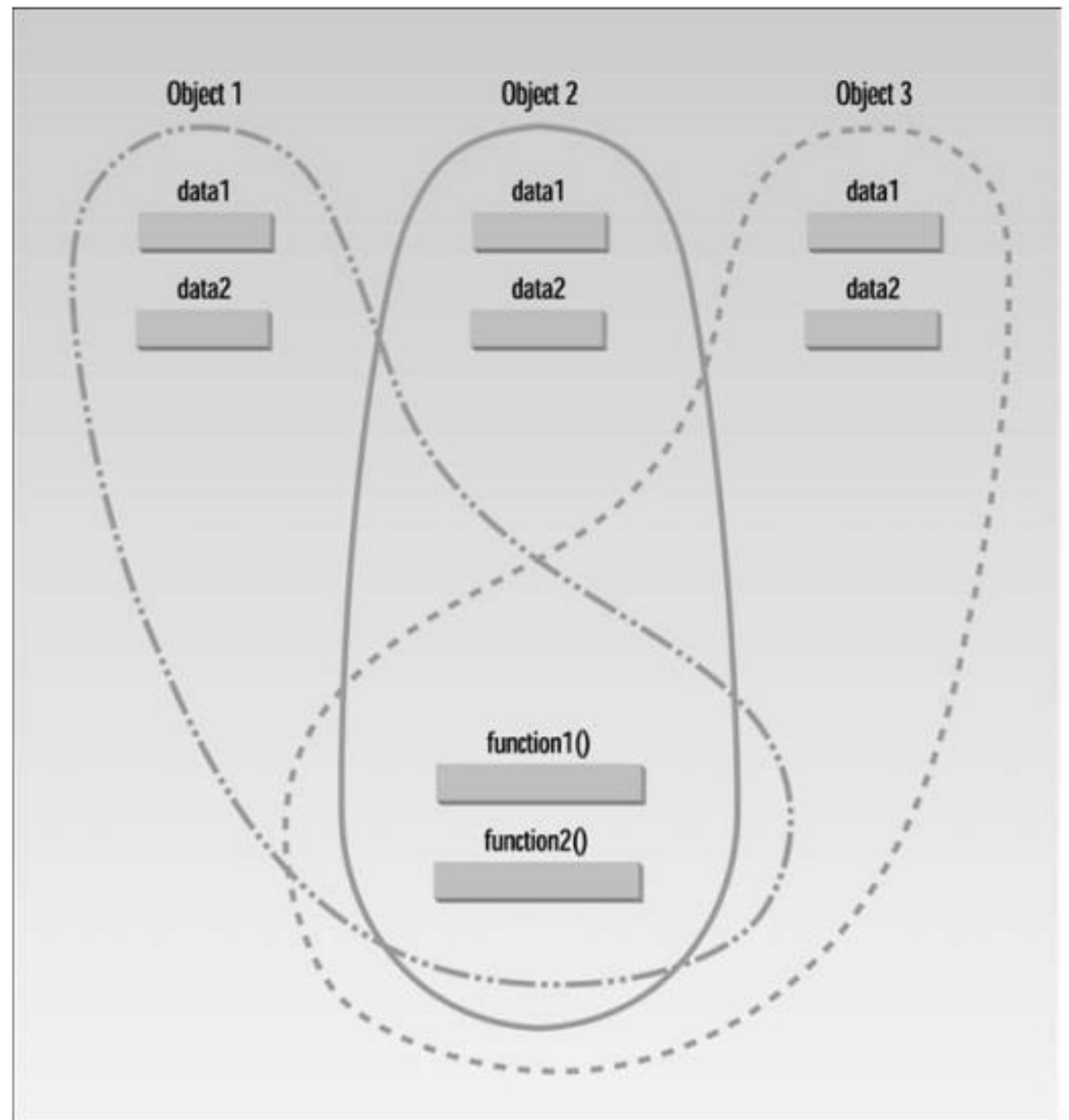
int main() {
    const Test t;
    cout << t.getValue();
    return 0;
}
```

Data members and Member Functions

Memory Representation

- Each newly created object have copies of that class's data and member functions.
- The member functions are created and placed in memory only once when they are defined in the class definition.
- This makes sense; there's really no point in duplicating all the member functions in a class every time you create another object of that class,
- since the functions for each object are identical.
- The data items, however,
- will hold different values, so there must be a separate instance of each data item for each object.
- Data is therefore placed in memory when each object is defined, so there is a separate set of data for each object.

Example



Static Data Member

- We can define class members static using **static** keyword.
- When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.
- A static member is shared by all objects of the class.
- All static data is initialized to zero when the first object is created, if no other initialization is present.
- We can't put it in the class definition but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the scope resolution operator `::` to identify which class it belongs to.


```

1 #include <iostream>
2 using namespace std;
3 //////////////////////////////////////
4 class foo
5 {
6 private:
7     static int count;    //only one data item for all objects
8     //note: "declaration" only!
9 public:
10    foo()                //increments count when object created
11    { count++; }
12    int getcount()        //returns count
13    { return count; }
14 };
15 //-----
16 int foo::count = 0;      /*definition* of count
17 //////////////////////////////////////
18 int main()
19 {
20     foo f1, f2, f3;      //create three objects
21     cout << "count is " << f1.getcount() << endl; //each object
22     cout << "count is " << f2.getcount() << endl; //sees the
23     cout << "count is " << f3.getcount() << endl; //same value
24     return 0;
25 }

```

C:\Users\ShahTab\Documents\OOP\StaticDataMembers.exe

Output

```

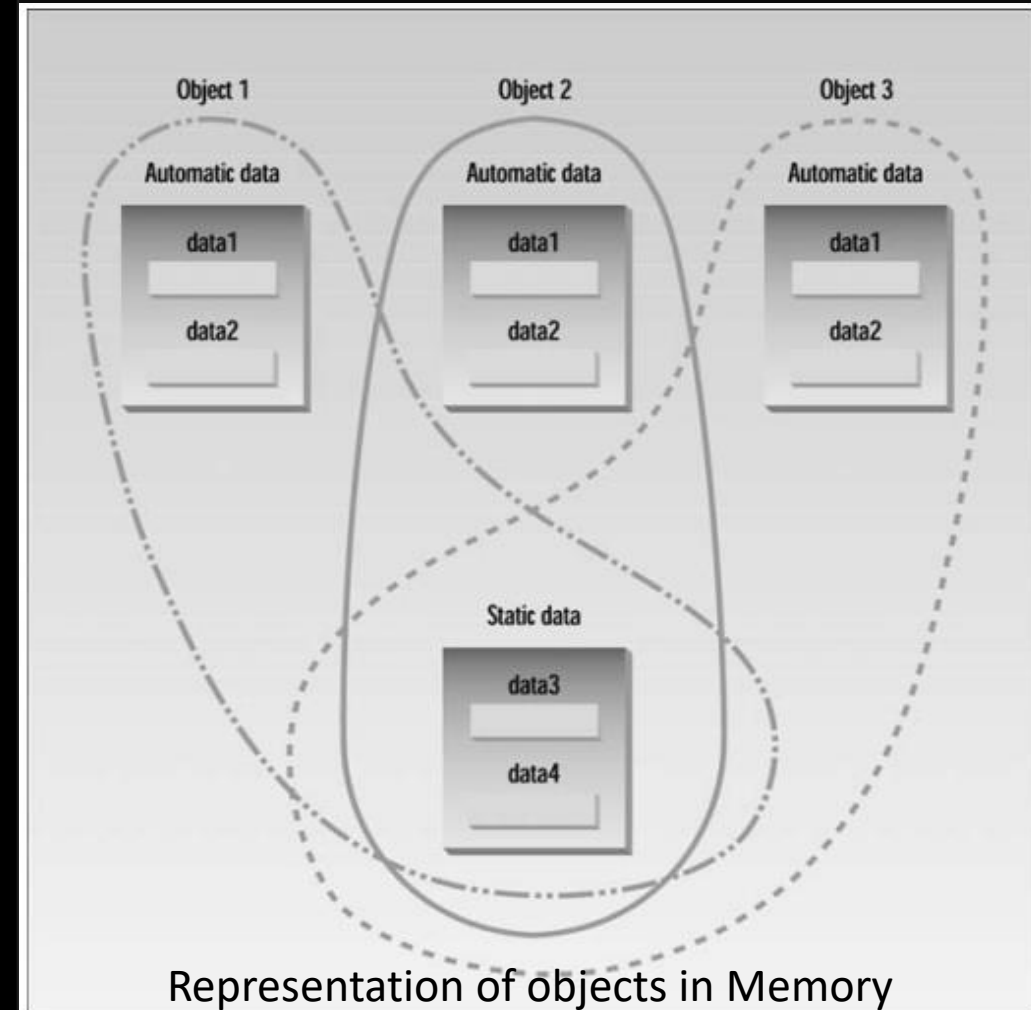
count is 3
count is 3
count is 3

```

```

-----
Process exited after 0.1298 seconds with return value 0
Press any key to continue . . .

```



Representation of objects in Memory

Static Member Functions

- By declaring a function member as static, you make it independent of any particular object of the class.
- A static member function can be called even if no objects of the class exist and the **static** functions are accessed using only the class name and the scope resolution operator ::.
- A static member function can only access static data member, other static member functions and any other functions from outside the class.
- Static member functions have a class scope
- You could use a static member function to determine whether some objects of the class have been created or not.

```

1  #include <iostream>
2  using namespace std;
3  class Box {
4      private:
5          double length;    // Length of a box
6          double breadth;    // Breadth of a box
7          double height;    // Height of a box
8          static int objectCount;
9      public:
10         // Constructor definition
11         Box(double l, double b, double h) {
12             cout << "Constructor called." << endl;
13             length = l;
14             breadth = b;
15             height = h;
16             // Increase every time object is created
17             objectCount++;
18         }
19         double Volume() {
20             return length * breadth * height;
21         }
22         static int getCount() {
23             return objectCount;
24         }
25     };
26     int Box::objectCount=0; //initialization of static data member
27     int main(void) {
28         // Print total number of objects before creating object.
29         cout << "Initial Stage Count: " << Box::getCount() << endl;
30         Box Box1(3.3, 1.2, 1.5);    // Declare box1
31         Box Box2(8.5, 6.0, 2.0);    // Declare box2
32         // Print total number of objects after creating object.
33         cout << "Final Stage Count: " << Box::getCount() << endl;
34         return 0;
35     }
36

```

C:\Users\ShahTab\Documents\OOP\StaticMemberFunction.exe

Initial Stage Count: 0

Constructor called.

Constructor called.

Final Stage Count: 2

Output

Process exited after 0.1591 seconds with return value 0

Press any key to continue . . .