Part1: **Constant objects and Member Function**

A function is made a constant function by placing the keyword const after the function header before the function body. A member function that does not change the value of its object but acquire data from their object is an obvious candidate for constant function. For example.

return\_type func\_name(argument list) const //const function

{

//Function body;

}

When an object is declared as constant, we can't modify it. A constant object can only call its constant member functions because they are the only ones that guarantee not to modify its value.

General syntax:

const class\_name object\_name; //creation of constant object

Here is an example

#include <iostream>

using namespace std;

class mk

{

private:

int d;

public:

mk()

{ d = 0; cout << "object created"<<endl; }

~mk()

{ cout << "object time out" << endl; }

void setd(int a)

{ d = a; }

int getd()

{ return d; }

int calx(int a) const

{ a++; return a;}

};

int main()

{

mk m1, m2;

const mk m3;

m1.setd(100);

m2.setd(200);

cout << m3.calx(102) << endl;

cout << m1.calx(100)<<endl;

return 0;

}

Output:

|  |
| --- |
|  |

Try to add the following function to mk class:

int cald(int a) const

{ d++; return a;}

What is the output, does it work, why?

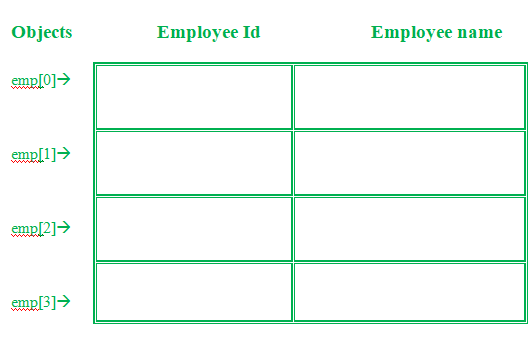
|  |
| --- |
|  |

Part2: **Array of class objects**

When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

The Array of Objects stores objects. An array of a class type is also known as an array of objects.

Example#1:   
Storing more than one Employee data. Let’s assume there is an array of objects for storing employee data emp[50].



Below is the C++ program for storing data of one Employee:

#include<iostream>

using namespace std;

class Employee

{

int id;

char name[30];

public:

// Declaration of function

void getdata();

// Declaration of function

void putdata();

};

// Defining the function outside

// the class

void Employee::getdata()

{

cout << "Enter Id : ";

cin >> id;

cout << "Enter Name : ";

cin >> name;

}

// Defining the function outside

// the class

void Employee::putdata()

{

cout << id << " ";

cout << name << " ";

cout << endl;

}

// client code

int main()

{

// This is an array of objects having

// maximum limit of 30 Employees

Employee emp[30];

int n, i;

cout << "Enter Number of Employees - ";

cin >> n;

// Accessing the function

for (i = 0; i < n; i++)

emp[i].getdata();

cout << "Employee Data - " << endl;

// Accessing the function

for (i = 0; i < n; i++)

emp[i].putdata();

}

* **Modify the above class so it asks users to get the salary for each employee.**

|  |
| --- |
|  |

Part3: **Static Data Members and member functions**

If a data item in class is declared as static, then only one copy of that item is created for the entire class, no matter how many objects there are. All the objects share a common item of information. The static data are to be defined separately and they are not allocated during object creation.

General syntax:

class class\_name

{

static data\_type variable\_name;

//declaration of static data member

};

data\_type class\_name::variable\_name;

//definition of static data member

//be assign to variable.

A static member function can have access to only other static members declared in the same class and it can be called using the class name

General Syntax:

static return\_type func\_name (argument list);

//declaration of static member function

//....

class\_name::fun\_name(argument passed)

//static member function call

Here is an example of static variables and member functions

#include <iostream>

using namespace std;

class mk

{

private:

int d;

static int f;

public:

mk()

{

d = f++;

}

void getdf()

{

cout << "d=" << d << endl;

}

static void getnob()

{

cout << "f=" << f << endl;

}

};

int mk::f = 0; // notice where the f initialized

int main()

{

mk o1[10];

for(int i=0;i<10;i++)

o1[i].getdf();

mk::getnob();

return 0;

}

Output

|  |
| --- |
| Summaries in few words what is the above code doing?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Update the getnob() function to print the d variable.**

|  |
| --- |
|  |

**What is the output, does it work, and why?**

|  |
| --- |
|  |

Exercises:

1. Create a class with a data member to hold a "serial number" for each object created from the class. That is, the first object created will be numbered 1, the second 2, and so on by using the basic concept of static data members. Use static member function if it is useful in the program. Otherwise, make a separate program that demonstrates the use of static member function.

* Use static variables and static member functions

1. Assume that one constructor initializes data member say num\_vehicle, hour, and rate. There should be a 10% discount if num\_vehicle exceeds 10. Display the total charge.
2. Use array of objects to creat a class for university courses. The class must ask users to save the id, name and credit hours for each course. In the client code (main function) create and array of objects to hold information for 100 courses.

That is the end of labsheet.. Good Luck