Name:- Sameer Manoj Bramhecha
Roll No:- 2115
Division:- SE-01
Date of Submission:- 11/09/2021

Date
Page

Assignment No: 2

Title:→ Demonstrate basic concepts of object oriented programming for a class to store details of student.

Objectives:→ 1.) To understand use of different types of constructor and destructor.
2.) To understand this pointer, dynamic memory allocation operators like new and delete.
3.) To understand use of static members and inline keyword.

Problem Statement:→ Develope an object oriented program in C++ to create a database of student information system containing the following information: Name, Roll number, class, division, Date of birth, Blood group, contant address, telephone number, driving license number etc. Construct the database with suitable member functions for initializing and destroying the data viz. Contructor, default constructor, copy constructor, destructor, static member functions, class, this pointer, inline code and dynamic memory allocation operators - new and delete.

Theory:→

Constructor:→ It is a special member function whose task is to initialize the objects of its class. This is the first method when an instance of a type is created. A constructor is invoked whenever an object of its associated class is created. If a class contains a constructor, then an object created by that class will be initialized

(2).

automatically. We pass data to the constructor by enclosing it in the parentheses following the class name when creating an object.

Characteristics of a constructor :→

1.] They should be declared in the public section.
2.] Constructor name and class name must be same.
3.] They are invoked automatically when the objects are created.
4.] They cannot be inherited
5.] They cannot be virtual.

Constructors are special functions which are having some different properties than that of the normal member functions of C++ class. They are special because they share exactly same name as that of class name and they get called automatically when the object of corresponding class is created in memory. In other words it means that constructor is invoked whenever the object of associated class is created. It's called constructor because it construct the values of data members of class.

A constructor is declared and defined as :→

```
class Sample
{
 int a,b;
 public:
  Sample()
};
Sample : Sample()
{
```

Date
Page

a=0; b=0;
}

When a class contains a constructor like the defined one above, it is guaranteed that an object created by class will initialized automatically.
For example:
Sample s;

It will not only create the object S of type Sample but also initializes its data members a & b to zero. A constructor with no parameter is called as the default constructor. If no constructor is defined then the compiler supplies a default constructor.

The constructors function has some special characteristics:
1.) The constructor should be declared in public section.
2.) They are automatically invoked when the objects are created.
3.) They do not have return type, not even void type is present. They do not have return values.
4) Similar to the other C++ function they can have also default arguement.
5) Constructors cannot be virtual.
6.) We cannot refer to their addresses.
7.) As object with constructor cannot be used as a member of union.
8.) They may implicit calls to the operators new and delete when memory allocation is required.
9.) They cannot be inherited though a derived class can call the base constructor.

Date
Page

Types of Constructors :→

Constructors are of 3 types :→

1.) Default constructor.

2.) Parameterized constructor.

3.) Copy constructor.

Default Constructor :→ It is the constructor which doesn't take any arguement. It has no parameter.

```
class Cube
{
int side;
public:
Cube()
{
side = 10;
}
};
int main();
{
Cube c;
cout << c.side;
}.
```

Parameterized Constructor :→ These are the constructors with parameter. Using this constructor you can provide different values to data members of different objects, by passing the appropriate values as arguement.

Date
Page

```
class Cube
{
int side;
public:
Cube (int x)
{
side = x;
}
};
int main ()
{
Cube c1(10);
Cube c2(20);
Cube c3(30);
cout << c1.side;
cout << c2.side;
Cout << c3.side;
}
```

Copy constructor:→ These are special type of constructors which takes an object as argument, and is used to copy values of data members of one object into other object.

```
class fun
{
float x,y;
public:
fun (float a, float b) // constructor
{
 x = a;
 y = b;
}
```

```
fun(fun &f) // copy constructor.
{
cout<<"\n copy constructor at work\n";
x=f.x;
y=f.y;
}
void display (void)
{
cout<< x << y << endl;
}
};
```

Destructor:→ It is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.
The syntax for destructor is same as that for the constructor; the class name is used for the name of destructor, with a tilde ~ sign as prefix to it.
Destructors will never have any arguements.

```
class A
{
public:
~A();
};
```

Allocation of memory:→ There are two ways that memory gets allocated for data storage:-

- Compile Time Allocation:- Memory for named variables is allocated by the compiler. Exact size and type of storage must be known at compile time. For standard array declarations, this is why size has to

Date
Page

be constant.

- Dynamic Memory Allocation:- Memory allocated "on the fly" during run time dynamically allocated space usually placed in a program segment known as the heap or the free store. Exact amount of space or number of items does not have to be known by the compiler in advance. For dynamic memory allocation, pointers are crucial.

Need of Dynamic Memory Allocation:→ All memory needs were determined before program execution by defining the variables needed. But there may be cases where the memory needs of a program can only be determined during runtime.
For example, when the memory needed depends on user input. On these cases, programs need to dynamically allocate memory, for which the C++ language integrates the operators new and delete.

Steps for Dynamic Memory Allocation:→
① Creating the dynamic space.
② Storing its address in a pointer (so that the space can be accessed)
③ Allocating space with new:
To allocate space dynamically, use the unary operator new, followed by the type being allocated.

```
new int;    //dynamically allocates an int.
new double; //dynamically allocates a double.
```

If creating an array dynamically, use the same form, but put brackets with a size after the type:

Date
Page

new int[40]; // dynamically allocates an array of 40 ints.

new double[size]; // dynamically allocates an array of
// size doubles.
// note that the size can be a variable

these statements above are not very useful by themselves, because the allocated spaces have no names! But, the new operator returns the starting address of the allocated space, and this address can be stored in a pointer:

```
int *p; // declare a pointer p
p = new int; // dynamically allocate an int and
             // load address into p
```

```
double *d; // declare a pointer d
d = new double; // dynamically allocate a double and
                // load address into d.
// we can also do these in single line statements.
```

```
int x = 40;
int *list = new int[x];
float *numbers = new float[x+10];
```

"this" pointer:- The 'this' pointer is passed as a hidden arguement to all non static member function calls and is available as a local variable within the body of all non-static functions.
'this' pointer is a constant pointer that holds the memory address of the current object. 'this' pointer

Date
Page

is not available in static member functions as static member functions can be called without any object (with class name).

Following is an example situation where 'this' pointer is used :-

When local variable's name is same as member's name

```
#include <iostream>
using namespace std;
/* local variable is same as a member's name */
class Test
{
private:
int x;
public:
void setX (int x)
{
// The 'this' pointer is used to retrieve the object's x
// hidden by the local.
variable 'x'
this -> x = x;
}
void print()
{
cout << "x = " << x << endl;
}
};
int main()
{
```

```
Test obj;
int x = 20;
obj.SetX(x);
obj.print();
return 0;
}
```

Algorithm:→

1) Define a class Student with data members, char *name, *rollno, *division, *dob, *bg, *address, *teleno, *dln. , ~~static int ch~~ choice. int choice.

2) In public, define constructor, parameterized constructor, copy constructor and destructor. and declare functions like display() and addata().

3) Define the function addata() and accept name, rollno, division, date of birth, blood group, address, telephoneno and driving license number in the function.

4) Define the function display() and display the accepted information ~~for~~ to the user.

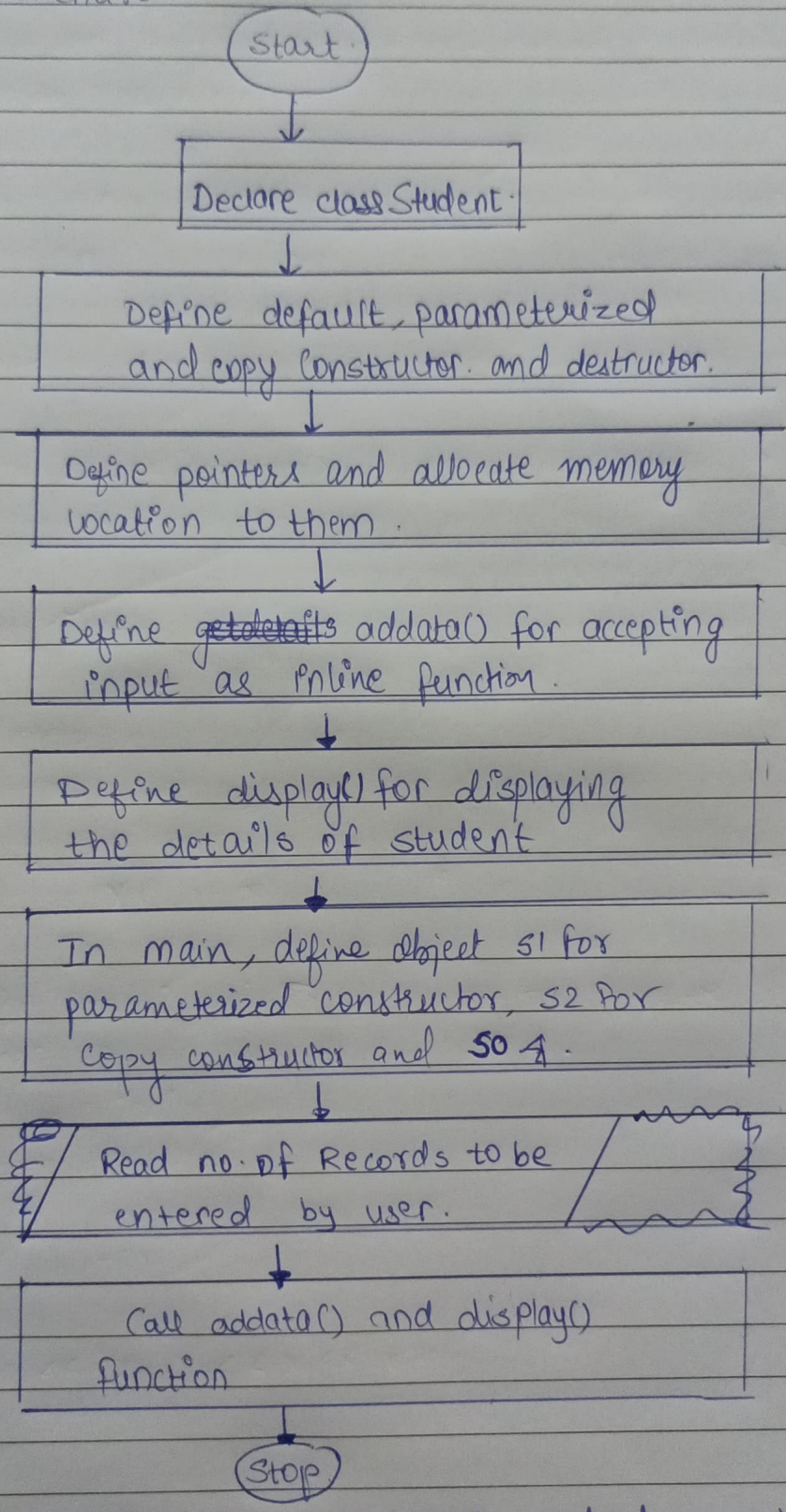
5) In main(), accept number of records to be entered in the database.

6) Depending Create an object of the class Student.

7) Using the object created, call the functions addata() and display().

Date
Page

Flowchart :→



Conclusion:→ Hence, we have studied concept of constructors (default, parameterized and copy), destructors, friend function, this pointer, inline code and dynamic memory allocation operators - new and delete.