Assignment No. 04

Roll No. SC55 - Shreyas Chavhan

===================================================

# Student Information System

## Problem Statement -

Develop 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, Contact address, telephone number, driving license no. etc Construct the database with suitable member functions for initializing and destroying the

data with constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline function.Code and dynamic memory allocation

operators-new and delete.

## Theory -

A special method of the class that will be automatically invoked when an instance of

the class is created is called as constructor. Following are the most useful features of constructor.

1) Constructor is used for Initializing the values to the data members of the Class.

2) Constructor is that whose name is same as name of class.

3) Constructor gets Automatically called when an object of class is created.

4) Constructors never have a Return Type even void.

5) Constructor is of Default, Parameterized and Copy Constructors.

The various types of Constructor are as follows:-

Constructors can be classified into 3 types

1. Default Constructor

2. Parameterized Constructor

3. Copy Constructor

1. Default Constructor:- Default Constructor is also called as Empty Constructor which has

no arguments and It is Automatically called when we creates the object of class but

Remember name of Constructor is same as name of class and Constructor never declared

with the help of Return Type.

2. Parameterized Constructor: - This is another type constructor which has some

Arguments and same name as class name but it uses some Arguments So For this We have

to create object of Class by passing some Arguments at the time of creating object with the

name of class. When we pass some Arguments to the Constructor then this will

automatically pass the Arguments to the Constructor and the values will retrieve by the

Respective Data Members of the Class.

3. Copy Constructor: - This is also another type of Constructor. In this Constructor we pass

the object of class into the Another Object of Same Class. As name Suggests you Copy,

means Copy the values of one Object into the another Object of Class .This is used for

Copying the values of class object into an another object of class So we call them as Copy

Constructor and For Copying the values We have to pass the name of object whose values

we wants to Copying and When we are using or passing an Object to a Constructor then

we must have to use the &amp; Ampersand or Address Operator.

Destructor: As we know that Constructor is that which is used for Assigning Some Values to data

Members and for Assigning Some Values this May also used Some Memory so that to free up the

Memory which is Allocated by Constructor, destructor is used which gets Automatically Called at

the End of Program and we doesn’t have to Explicitly Call a Destructor and Destructor Cant be

Parameterized or a Copy This can be only one Means Default Destructor which Have no

Arguments. For Declaring a destructor we have to use ~tiled Symbol in front of Destructor.

Static members

A class can contain static members, either data or functions.

A static member variable has following properties:

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.

It is the visible only within the class but its lifetime is the entire program.

Static data members of a class are also known as &quot;class variables&quot;, because there is only one

unique value for all the objects of that same class. Their content is not different from one object

static members have the same properties as global variables but they enjoy class scope. For that

reason, and to avoid them to be declared several times, we can only include the prototype (its

declaration) in the class declaration but not its definition (its initialization). In order to initialize a

static data-member we must include a formal definition outside the class, in the global scope of

this class to another. Because it is a unique variable value for all the objects of the same class, it

can be referred to as a member of any object of that class or even directly by the class name (of

course this is only valid for static members.

A static member function has following properties :

4. A static function can have access to only other static members (fun or var) declared in the

same class

5. A static function can be called using the class name instead of its object name

Class\_name :: function\_name;

Static member functions are considered to have class scope. In contrast to non static member

functions, these functions have no implicit this argument; therefore, they can use only static data

members, enumerators, or nested types directly. Static member functions can be accessed without

using an object of the corresponding class type.

The following restrictions apply to such static functions:

They cannot access non static class member data using the member-selection operators

(. or –&gt;).

They cannot be declared as virtual.

They cannot have the same name as a non static function that has the same argument

types.

Ex. Shall we give the example………………………

Friend functions:

In principle, private and protected members of a class cannot be accessed from outside the same

class in which they are declared. However, this rule does not affect friends. Friends are functions

or classes declared as such. If we want to declare an external function as friend of a class, thus

allowing this function to have access to the private and protected members of this class, we do it

by declaring a prototype of this external function within the class, and preceding it with the

keyword friend.

Properties of friend function:

It is not in the scope of the class to which it has been declared as friend.

Since it is not in the scope of the class , it cannot be called using the object of that class

It can be invoked like a normal function w/o the help of any object.

It can be declared in private or in the public part of the class.

Unlike member functions, it cannot access the member names directly and has to use an

object name and dot operator with each member name.

Friend classes

Just as we have the possibility to define a friend function, we can also define a class as friend of

another one, granting that second class access to the protected and private members of the first

one.

Pointers:

A pointer is a derived data type that refers to another data variable by storing the variables

memory address rather than data.

Declaration of pointer variable is in the following form:

Data\_type \* ptr\_var;

Eg. int \* ptr;

Here ptr is a pointer variable and points to an integer data type.

We can initialize pointer variable as follows

int p, \* ptr ; // declaration

ptr = &amp; a ; // initialization

Pointers to objects:

Consider the following example

item P ; // where item is class &amp; P is object

Similarly, we can define a pointer item\_ptr of type item as follows

item \*it\_ptr ;

Object pointers are useful in creating objects at runtime. We can also

access public members of the class using pointers.

Ex. item X;

item \*ptr = &amp;X;

the pointer „ptr „is initialized with address of X.

we can access the member functions and data using pointers as

follows

ptr getdata();

ptr show();

this pointer:

C++ uses a unique keyword called this to represent an object that invokes a member function. this

is a pointer that points to the object for which this function was called. This unique pointer is

automatically passed to a member function when it is called.

Important notes on this pointer:

this pointer stores the address of the class instance, to enable pointer access of the

members to the member functions of the class.

this pointer is not counted for calculating the size of the object.

this pointers are not accessible for static member functions.

this pointers are not modifiable.

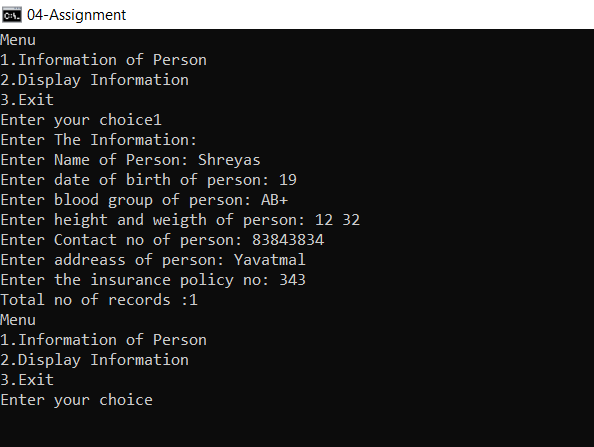
## Code -

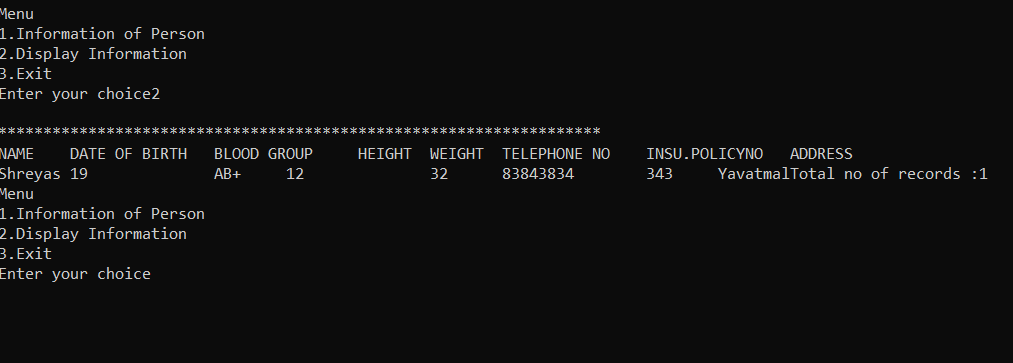
|  |
| --- |
| #include <iostream>  #include <string>  #include <cstring>  using namespace std;  class person  {  private:  char name[40], date\_of\_birth[15], blood\_group[15];  int h, w;  public:  static int count;  friend class personal;  person()  {  char \*name = new char[40];  char \*date\_of\_birth = new char[80];  char \*blood\_group = new char[15];  h = w = 0;  }  static void recordcount()  {  cout << "Total no of records :" << count << endl;  }  };  class personal  {  private:  char add[70], telephone[15], policy\_no[10];  public:  personal()  {  strcpy(add, "");  strcpy(telephone, "");  strcpy(policy\_no, "");  }  void getdata(person \*obj);  void displaydata(person \*obj);  friend class person;  };  int person::count = 0;  void personal::getdata(person \*obj)  {  cout << "Enter Name of Person: ";  cin >> obj -> name;  cout << "Enter date of birth of person: ";  cin >> obj -> date\_of\_birth;  cout << "Enter blood group of person: ";  cin >> obj -> blood\_group;  cout << "Enter height and weigth of person: ";  cin >> obj -> h >> obj -> w;  cout << "Enter Contact no of person: ";  cin >> this -> telephone;  cout << "Enter addreass of person: ";  cin >> this -> add;  cout << "Enter the insurance policy no: ";  cin >> this -> policy\_no;  obj->count++;  }  void personal :: displaydata(person \*obj)  {  cout << obj -> name << "\t"  << obj -> date\_of\_birth << "\t\t"  << obj -> blood\_group << "\t"  << obj -> h << "\t\t"  << obj -> w << "\t"  << this -> telephone << "\t"  << this -> policy\_no << "\t"  << this -> add;  }  int main()  {  personal \*p1[30];  person \*p2[30];  int n = 0, ch, i;  do  {  cout << "Menu" << endl;  cout << "1.Information of Person" << endl;  cout << "2.Display Information" << endl;  cout << "3.Exit" << endl;  cout << "Enter your choice";  cin >> ch;  switch (ch)  {  case 1:  cout << "Enter The Information: " << endl;  p1[n] = new personal;  p2[n] = new person;  p1[n]->getdata(p2[n]);  n++;  person::recordcount();  break;  case 2:  cout << " ";  cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "NAME"  << "\t"  << "DATE OF BIRTH"  << "\t"  << "BLOOD GROUP"  << "\t"  << "HEIGHT"  << "\t"  << "WEIGHT"  << "\t"  << "TELEPHONE NO"  << "\t"  << "INSU.POLICYNO"  << "\t"  << "ADDRESS ";  cout << endl;  for (i = 0; i < n; i++)  {  p1[i]->displaydata(p2[i]);  }  person::recordcount();  break;  }  } while (ch != 4);  return 0;  } |

Roll No. SC55 - Shreyas Chavhan

===================================================

## Output -





===================================================

Roll No. SC55 - Shreyas Chavhan