**Assignment No:** 1

**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 viz. static member functions, friend class/friend functions, this pointer, inline codes and dynamic memory allocation operators-new and delete. Implement all the keywords as mentioned in the problem statement.

**Aim of Assignment:** To implement various C++ concepts such as friend functions, this pointer, etc. in order to design a student information system.

**Description:** Two classes named personaldetails and student are used in which personaldetails is used to store the personal information whereas the data like roll number, class and divison is stored in student class. Student class is derived from the personaldetails class depicting the inheritance concept. Four functions called accept(), acceptstud(), display() , searchh() and modify() is used to get, set,search and modify the data respectively. Here, accept() function is used to accept the personal details and acceptstud() is used to accept student information. Display() is used to display the record. Searchh() is the friend function used to search the entry of the user inputed roll number. Modify() is also the friend function used to to modify the record of user inputed roll number. main() has array of objects of student class of size 10.Also it contains the do-while loop due which multiple instances can be stored at one run. Switch is used to let user choose whether to accept or display the data.

**OOP Concept used:**

1. Static member :
2. **Static variables in a Function**: When a variable is declared as static, space for it gets allocated for the lifetime of theprogram. Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call.
3. **Static variables in a class**: As the variables declared as static are initialized only once as they are allocated space in separate static storage so, the static variables in a class are shared by the objects**.** There can not be multiple copies of same static variables for different objects. Also because of this reason static variables can not be initialized using constructors
4. Friend Function: Like friend class, a friend function can be given special grant to access private and protected members. A friend function can be:  
   a) A method of another class  
   b) A global function
5. This pointer: In C++, this pointer is used to represent the address of an object inside a member function. For example, consider an object obj calling one of its member function say method() as obj.method(). Then, this pointer will hold the address of object obj inside the member function method(). The this pointer acts as an [implicit argument](http://stackoverflow.com/questions/3057859/whats-the-difference-between-explicit-and-implicit-assignment-in-c) to all the member functions.
6. Inline code: C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.
7. New and Delete operator: The new operator denotes a request for memory allocation on the Heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable. Since it is programmer’s responsibility to deallocate dynamically allocated memory, programmers are provided delete operator by C++ language.
8. Inheritance: The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important feature of Object Oriented Programming.  
   **Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.  
   **Super Class:**The class whose properties are inherited by sub class is called Base Class or Super class.
9. Access Modifiers: Access modifiers are used to implement an important feature of Object-Oriented Programming known as [Data Hiding](https://practice.geeksforgeeks.org/problems/what-is-data-hiding). Consider a real-life example:  
   The Indian secret informatic system having 10 senior members have some top secret regarding national security. So we can think that 10 people as class data members or member functions who can directly access secret information from each other but anyone can’t access this information other than these 10 members i.e. outside people can’t access information directly without having any privileges. This is what data hiding is.  
   Access Modifiers or Access Specifiers in a [class](https://www.geeksforgeeks.org/c-classes-and-objects/) are used to set the accessibility of the class members. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions.

There are 3 types of access modifiers available in C++:

Public

Private

Protected

**Conclusion:** The various OOP concepts are implemented successfully.

**Program:**

#include<iostream>

#include<limits>

#include<string>

#include<sstream>

using namespace std;

class DateV

{

    private:

            int date,month,year;

            int days;

    public:

            DateV(int dd,int mm,int yy)

            {

                date=dd;

                month=mm;

                year=yy;

            }

//function for validation of date

            bool isValidDate()

            {

                if(year >= 1950 && year<=2001)

                {

                    switch(month)

                    {

                        case 1:case 3: case 5: case 7: case 8: case 10: case 12: days=31;

                        break;

                        case 4:case 6: case 9:case 11: days=30;

                        break;

                        case 2: if(year%4 == 0)

                                    days=29;

                                else

                                    days=28;

                        break;

                        default: return false;

                    }

                    if(date>=1 && date<=days)

                        return true;

                    else

                        return false;

                }

                else

                {

                    return false;

                }

            }

};

class Employee

{

  private:

  int empid;

char empname[30];

double empsal;

char empaddr[100];

int empbDay,empbMonth,empbYear;

string empDOB;

string empbloodGroup;

string empphone;

bool flag;

public:

            void setEmpId()

            {

                cin>>empid;

                while(1)

                {

                    if(cin.fail())

                    {

                        cin.clear();

                        cin.ignore(numeric\_limits<streamsize>::max(),'\n');

                        cout<<"\nEmployee ID should be number Only!";

                        cout<<"\nEmployee ID: ";

                        cin>>empid;

                    }

                    else

                    {

                        break;

                    }

                }

            }

            void setEmpName()

            {

                cin.ignore();

                cin.getline(empname,30,'\n');

            }

            void setEmpSalary()

            {

                cin>>empsal;

                while(1)

                {

                    if(cin.fail())

                    {

                        cin.clear();

                        cin.ignore(numeric\_limits<streamsize>::max(),'\n');

                        cout<<"\nEmployee Salary should be number Only!";

                        cout<<"\nEmployee Salary: ";

                        cin>>empsal;

                    }

                    else

                    {

                        break;

                    }

                }

            }

            void setEmpPhoneNo()

            {

                try

                {

                    int digitCount;

                    cin>>empphone;

                    digitCount=empphone.length();

                    bool first=empphone.at(0)=='9' ||empphone.at(0)=='8'||empphone.at(0)=='7'||empphone.at(0)=='6';

                    while(1)

                    {

                        if(digitCount!=10 || (!first))

                        {

                            throw "Invalid phone number try again!!!";

                        }

                        else

                        {

                            break;

                        }

                    }

                }catch(const char\* exp)

                {

                    cout<<"\nError!! "<<exp;

                    cout<<"\nEnter the phone No: ";

                    setEmpPhoneNo();

                }

            }

            void setEmpAddress()

            {

                cin.ignore();

                cin.getline(empaddr,50,'\n');

            }

            void setEmpDOB()

            {

                bool invalid;

                do

                {

                    cout<<"\nDate: ";

                    cin>>empbDay;

                    cout<<"\nMonth: ";

                    cin>>empbMonth;

                    cout<<"\nYear: ";

                    cin>>empbYear;

                    DateV d(empbDay,empbMonth,empbYear);

                    if(!d.isValidDate())

                    {

                        cout<<"Invalid Date Please try again!!\n";

";

                        invalid=true;

                    }

                }while(invalid);

                stringstream ss;

                ss<<empbDay<<"/"<<empbMonth<<"/"<<empbYear;

                ss>>empDOB;

            }

            void setEmpBloodGroup()

            {

             int bloodType;

             do

             {

                    cout<<"\nEnter your choice..!";

                    cout<<"\n1: A+\t2: A-\t3: B+\t4: B-\t5: O+\t6: O-\t7: AB+\t8: AB-\n";

                    cin>>bloodType,flag=1;

                    switch(bloodType)

                    {

                     case 1: empbloodGroup="A+";

                     break;

                     case 2: empbloodGroup="A-";

                     break;

                     case 3: empbloodGroup="B+";

                     break;

                     case 4: empbloodGroup="B-";

                     break;

                     case 5: empbloodGroup="O+";

                     break;

                     case 6: empbloodGroup="O-";

                     break;

                     case 7: empbloodGroup="AB+";

                     break;

                     case 8: empbloodGroup="AB-";

                     break;

                     default: cout<<"\nWrong choice, Please Enter Valid Blood Group!";

                     flag=0;

                    }

                }while(flag==0);

            }

            int getEmpId()

            {

                return empid;

            }

            char\* getEmpName()

            {

                return empname;

            }

            double getEmpSalary()

            {

                return empsal;

            }

            string getEmpPhoneNo()

            {

                return empphone;

            }

            char\* getEmpAddress()

            {

                return empaddr;

            }

            string getEmpDOB()

            {

                return empDOB;

            }

            string getEmpBloodGroup()

            {

                return empbloodGroup;

            }

void acceptData();

void displayData();

 };

void Employee::acceptData()

{

    cout<<"\nEmployee ID: ";

    setEmpId();

    cout<<"\nEmployee Name: ";

    setEmpName();

    cout<<"\nEmployee Salary: ";

    setEmpSalary();

    cout<<"\nEmployee Phone No: ";

    setEmpPhoneNo();

    cout<<"\nEmployee Address: ";

    setEmpAddress();

    cout<<"\nEmployee DOB: ";

    setEmpDOB();

    cout<<"\nEmployee Blood Group: ";

    setEmpBloodGroup();

}

void Employee::displayData()

{

    cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Employee Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

    cout<<"\nID\t\t |\t"<<getEmpId();

    cout<<"\nName\t\t |\t"<<getEmpName();

    cout<<"\nSalary\t\t |\t"<<getEmpSalary();

    cout<<"\nPhone No.\t |\t"<<getEmpPhoneNo();

    cout<<"\nAddress\t\t |\t"<<getEmpAddress();

    cout<<"\nDOB\t\t |\t"<<getEmpDOB();

    cout<<"\nBlood Group\t |\t"<<getEmpBloodGroup();

    cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

int main()

{

  cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Employee\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

  Employee e;

  e.acceptData();

  e.displayData();

  return 0;

}

  
  
  
  
  
  
  
  
  
  
  
  
