**Modern Education Society’s**

**College of Engineering, Pune**

|  |  |
| --- | --- |
| **NAME OF STUDENT:** Prathamesh Kalyan Sable | **CLASS:** SE Comp 1 |
| **SEMESTER/YEAR:** Sem-3 / 2022-23 | **ROLL NO:** 015 |
| **DATE OF PERFORMANCE:**  / /2022 | **DATE OF SUBMISSION:** / /2022 |
| **EXAMINED BY:** Prof. R. H. Shende | **EXPERIMENT NO: A-2** |

###### TITLE : STUDENT DATABASE

**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 viz constructor, default constructor, Copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

### **OBJECTIVES:**

1. To use the object-oriented paradigm in program design.
2. To lay a foundation for advanced programming.
3. Provide programming insight using OOP constructs.

### **OUTCOMES:**

1. Analyze the strengths of object oriented programming.
2. Design and apply OOP principles for effective programming.

**PRE-REQUISITES:**

* 1. Knowledge of object oriented programming concepts.

**APPARATUS:**

Working Computer system with g++ installed

**QUESTIONS:**

1. Can we have more than one constructors in a class? If yes, explain the need for such a situation?
2. How is dynamic initialization of objects achieved?

**SOURCE CODE:**

#include <iostream>

#include <iomanip>

#include <fstream>

#define max 100  // defining size of max array

using namespace std;

class student;

class information {

    string name, dob, bloodgrp, mob\_num, address, dl\_num;

   public:

    information()  // default constructor

    {

        name = "ABC";

        dob = "dd/mm/yyyy";

        bloodgrp = "X+";

        mob\_num = "1234567890";

        address = "Pune, Maharashtra, India";

        dl\_num = "MH01/0123/0123456";

    }

    information(const information \*obj)  // copy constructor

    {

        this->name = obj->name;

        this->dob = obj->dob;

        this->bloodgrp = obj->bloodgrp;

        this->mob\_num = obj->mob\_num;

        this->address = obj->address;

        this->dl\_num = obj->dl\_num;

    }

    information(string name, string dob, string bloodgrp, string mob\_num, string address, string dl\_num)  // constructor

    {

        this->name = name;

        this->dob = dob;

        this->bloodgrp = bloodgrp;

        this->mob\_num = mob\_num;

        this->address = address;

        this->dl\_num = dl\_num;

    }

    void edit(int index, string list[]) {

        cout << "Enter new " << list[index] << ":";

        cin.ignore();

        switch (index) {

            case 1:

                getline(cin, name);

                break;

            case 5:

                getline(cin, dob);

                break;

            case 6:

                getline(cin, bloodgrp);

                break;

            case 7:

                getline(cin, mob\_num);

                break;

            case 8:

                getline(cin, dl\_num);

                break;

            case 9:

                getline(cin, address);

                break;

            default:

                break;

        }

    }

    friend class student;

    friend void display\_all(student \*s\_list[], information \*s\_info[]);

    friend void save\_data(student \*s\_list[], information \*s\_info[]);

};

class student {

    int roll\_no;

    string class\_name, remark;

    char div;

    static int student\_count;

   public:

    student()  // default constructor

    {

        roll\_no = 0;

        class\_name = "SE Comp";

        div = 'A';

        remark = "";

        // 'student\_count++' not required;

    }

    student(int roll\_no, string class\_name, char div, string remark = "")  // constructor

    {

        this->class\_name = class\_name;

        this->div = div;

        this->remark = remark;

        this->roll\_no = roll\_no;

        student\_count++;

    }

    student(const student \*stu)  // copy constructor

    {

        this->class\_name = stu->class\_name;

        this->div = stu->div;

        this->remark = stu->remark;

        this->roll\_no = stu->roll\_no;

        student\_count++;

    }

    ~student()  // destructor

    {

        student\_count--;

    }

    // friend functions

    friend int search\_stu(int roll, student \*s\_list[]);

    friend void display\_all(student \*s\_list[], information \*s\_info[]);

    friend void save\_data(student \*s\_list[], information \*s\_info[]);

    inline static int get\_student\_count() {

        return student\_count;

    }

    void display\_details(const information \*info) {

        cout << "Student Name: " << info->name << endl;

        cout << "Roll No.: " << roll\_no << endl;

        cout << "Class: " << class\_name << endl;

        cout << "Division :" << div << endl;

        cout << "Date of Birth: " << info->dob << endl;

        cout << "Blood Group: " << info->bloodgrp << endl;

        cout << "Mobile Number: " << info->mob\_num << endl;

        cout << "DL Number: " << info->dl\_num << endl;

        cout << "Adderss of Student: " << info->address << endl;

        cout << "Remarks(if any): " << remark << endl;

    }

    void edit(information \*info) {

        int choice;

        string value;

        string choices[11] = {"", "Student Name", "Roll No.", "Class", "Division", "Date of Birth", "Blood Group", "Mobile Number", "DL Number", "Adderss of Student", "Remarks"};

        cout << "What you Want to Edit?" << endl;

        for (int j = 1; j <= 10; j++)

            cout << j << ". " << choices[j] << endl;

        cout << "11. Cancel" << endl;

        cout << "Enter Your Choice:";

        cin >> choice;

        if (choice <= 0 or choice > 11) {

            cout << "Invalid Choice, Canceling Edit." << endl;

            return;

        }

        if (choice == 11) {

            cout << "Canceling Edit." << endl;

            return;

        }

        if (choice == 3 or choice == 10) {

            cout << "Enter new " << choices[choice] << ":";

            cin.ignore();

            getline(cin, value);

            if (choice == 3) {

                class\_name = value;

            } else {

                remark = value;

            }

        } else if (choice == 4) {

            cout << "Enter new division:";

            cin >> div;

        } else if (choice == 2) {

            cout << "Enter new Roll no:";

            cin >> roll\_no;

        } else {

            info->edit(choice, choices);

        }

    }

};

int student::student\_count = 0;

// print row function

template <class i, class c>

void row\_print(i sr, string name, i roll, string cls, c div, string dob, string bgrp, string mobile, string dlno, string add, string remark, char sep = '|') {

    cout << sep << setw(5) << right << sr << sep;

    cout << setw(15) << left << name.substr(0, 15) << sep;

    cout << setw(7) << right << roll << sep;

    cout << setw(9) << left << cls.substr(0, 9) << sep;

    cout << setw(3) << left << div << sep;

    cout << setw(10) << left << dob.substr(0, 10) << sep;

    cout << setw(9) << left << bgrp.substr(0, 9) << sep;

    cout << setw(13) << left << mobile.substr(0, 13) << sep;

    cout << setw(17) << left << dlno.substr(0, 17) << sep;

    cout << setw(20) << left << add.substr(0, 20) << sep;

    cout << setw(15) << left << remark.substr(0, 15) << sep << endl;

}

// display all data

void display\_all(student \*s\_list[], information \*s\_info[]) {

    row\_print<string, string>("-----", "---------------", "-------", "---------", "---", "----------", "---------", "-------------", "-----------------", "--------------------", "---------------", '+');

    row\_print<string, string>("Sr No", "Student Name", "Roll No", "Class", "Div", "Birth Date", "Blood Grp", "Mobile Number", "DL Number", "Adderss of Student", "Remarks(if any)");

    row\_print<string, string>("-----", "---------------", "-------", "---------", "---", "----------", "---------", "-------------", "-----------------", "--------------------", "---------------", '+');

    for (int i = 0; i < student::get\_student\_count(); i++) {

        row\_print<int, char>(i + 1, s\_info[i]->name, s\_list[i]->roll\_no, s\_list[i]->class\_name, s\_list[i]->div, s\_info[i]->dob, s\_info[i]->bloodgrp, s\_info[i]->mob\_num, s\_info[i]->dl\_num, s\_info[i]->address, s\_list[i]->remark);

    }

    row\_print<string, string>("-----", "---------------", "-------", "---------", "---", "----------", "---------", "-------------", "-----------------", "--------------------", "---------------", '+');

}

void add\_student(int index, student \*s\_list[], information \*s\_info[]) {

    string name, dob, bloodgrp, mob\_num, address, dl\_num;

    int roll\_no;

    string class\_name, remark;

    char div;

    bool while\_control = true;

    char choice;

    while (while\_control) {

        cin.clear();

        try {

            // student info data inputs

            cin.ignore();

            cout << "Enter Student Details below(\* Indicates Required)" << endl;

            cout << "\*Name:";

            getline(cin, name);

            cout << "\*DOB(DD/MM/YYYY):";

            getline(cin, dob);

            cout << "Blood Group:";

            getline(cin, bloodgrp);

            cout << "Mobile Number:";

            getline(cin, mob\_num);

            cout << "\*Address:";

            getline(cin, address);

            cout << "DL Number:";

            getline(cin, dl\_num);

            // student list data inputs

            cout << "Class:";

            getline(cin, class\_name);

            cout << "\*Division:";

            cin >> div;

            cout << "\*Roll Number:";

            cin >> roll\_no;

            cout << "Remarks:";

            cin.ignore();

            getline(cin, remark);

            if (cin.fail()) {

                throw(0);

            } else if (name.size() == 0) {

                throw(1);

            } else if (dob.size() != 10) {

                throw(2);

            } else if (address.size() == 0) {

                throw(3);

            } else {

                while\_control = false;

            }

        } catch (int err\_code) {

            switch (err\_code) {

                case 0:

                    cout << "Failed to Add record, Please Try Again" << endl;

                    break;

                case 1:

                    cout << "Name can not be empty" << endl;

                    break;

                case 2:

                    cout << "Enter a valid DOB in DD/MM/YYYY format" << endl;

                    break;

                case 3:

                    cout << "Address can not be empty" << endl;

                    break;

                default:

                    cout << "Unexcepted error occured" << endl;

                    break;

            }

            cout << "Want to Retry(y/n):";

            cin >> choice;

            if (choice == 'y' || choice == 'Y') {

                while\_control = true;

            } else {

                while\_control = false;

                return;

            }

        }

    }

    s\_info[index] = new information(name, dob, bloodgrp, mob\_num, address, dl\_num);

    s\_list[index] = new student(roll\_no, class\_name, div, remark);

}

int search\_stu(int roll, student \*s\_list[]) {

    for (int i = 0; i < student::get\_student\_count(); i++) {

        if (s\_list[i]->roll\_no == roll) {

            return i;

        }

    }

    return -1;

}

void save\_data(student \*s\_list[], information \*s\_info[]) {

    ofstream data\_file("StudentData.txt");

    char sep = '$';

    for (int i = 0; i < student::get\_student\_count(); i++) {

        // add record to file

        data\_file << s\_info[i]->name << sep << s\_list[i]->roll\_no << sep << s\_list[i]->class\_name << sep << s\_list[i]->div << sep << s\_info[i]->dob << sep << s\_info[i]->bloodgrp << sep << s\_info[i]->mob\_num << sep << s\_info[i]->dl\_num << sep << s\_info[i]->address << sep << s\_list[i]->remark << endl;

    }

    data\_file.close();

}

void load\_data(int \*current, student \*s\_list[], information \*s\_info[]) {

    // checking if file exists

    ifstream data\_file("StudentData.txt");

    if (data\_file) {

        cout << "Loading Previous Data, Please Wait..." << endl;

    } else {

        // file doesn't exist

        return;

    }

    char sep = '$';

    string line;

    string all\_data[10] = {""};

    int last, line\_index;

    int err\_cnt = 0;

    while (data\_file.eof() == 0) {

        getline(data\_file, line);

        if (line == "") {

            // skip empty line

            continue;

        }

        try {

            last = 0;

            line\_index = 0;

            for (int i = 0; i < line.size(); i++) {

                if (line[i] == sep) {

                    all\_data[line\_index] = line.substr(last, i - last);

                    last = i + 1;

                    line\_index++;

                }

            }

            all\_data[line\_index] = line.substr(last, line.size() - last);

            // passign values to function for object creation

            s\_info[\*current] = new information(all\_data[0], all\_data[4], all\_data[5], all\_data[6], all\_data[8], all\_data[7]);

            s\_list[\*current] = new student(stoi(all\_data[1]), all\_data[2], all\_data[3][0], all\_data[9]);

            (\*current)++;

        } catch (...) {

            err\_cnt++;

        }

    }

    data\_file.close();

    cout << "Imported " << \*current << " records;" << err\_cnt << " Import Error" << endl;

}

int main() {

    // list of pointers for data

    student \*stu\_list[max];

    information \*stu\_info[max];

    // sample rec

    student \*sample\_s = new student;

    information \*sample\_i = new information;

    bool while\_control = true;

    int srch, ind, choice, current = 0;

    // load past data

    load\_data(&current, stu\_list, stu\_info);

    while (while\_control) {

        cout << "-------- MENU --------" << endl;

        cout << "  1. Add New Student" << endl;

        cout << "  2. Display All Students" << endl;

        cout << "  3. Search and Display Student Record" << endl;

        cout << "  4. Delete Student Record" << endl;

        cout << "  5. Edit Student Record" << endl;

        cout << "  6. Show Sample Record" << endl;  // to execute default constructor

        cout << "  7. Duplicate Record" << endl;    // to execute copy constructor

        cout << "  8. Save & Exit" << endl;

        cout << "Enter your choice:";

        cin >> choice;

        cout << endl;

        switch (choice) {

            case 1:

                // condition to check limit

                if (current < 100) {

                    add\_student(current, stu\_list, stu\_info);

                    current++;

                    cout << "Added student record." << endl;

                } else {

                    cout << "Can't add student record, Because Reached the limit of records." << endl;

                }

                break;

            case 2:

                cout << "Found " << student::get\_student\_count() << " records." << endl;

                if (student::get\_student\_count() != 0) {

                    display\_all(stu\_list, stu\_info);

                }

                break;

            case 3:

                cout << "Enter Roll number to Search(Enter 0 display list):";

                cin >> srch;

                if (srch == 0) {

                    display\_all(stu\_list, stu\_info);

                    cout << "Enter Roll number to Search:";

                    cin >> srch;

                }

                ind = search\_stu(srch, stu\_list);

                if (ind == -1)

                    cout << "No Record Found" << endl;

                else {

                    cout << "\tRecord Found." << endl;

                    stu\_list[ind]->display\_details(stu\_info[ind]);

                }

                break;

            case 4:

                cout << "Enter Roll number to Delete(Enter 0 display list):";

                cin >> srch;

                if (srch == 0) {

                    display\_all(stu\_list, stu\_info);

                    cout << "Enter Roll number to Delete:";

                    cin >> srch;

                }

                ind = search\_stu(srch, stu\_list);

                if (ind == -1)

                    cout << "No Record Found to Delete" << endl;

                else {

                    delete stu\_list[ind];

                    delete stu\_info[ind];

                    stu\_list[ind] = stu\_list[current - 1];

                    stu\_info[ind] = stu\_info[current - 1];

                    current--;

                    cout << "Record Deleted Successfully." << endl;

                }

                break;

            case 5:

                cout << "Enter Roll number to Edit(Enter 0 display list):";

                cin >> srch;

                if (srch == 0) {

                    display\_all(stu\_list, stu\_info);

                    cout << "Enter Roll number to Edit:";

                    cin >> srch;

                }

                ind = search\_stu(srch, stu\_list);

                if (ind == -1)

                    cout << "No Record Found to Edit" << endl;

                else {

                    stu\_list[ind]->edit(stu\_info[ind]);

                    cout << "Record Edited Successfully." << endl;

                }

                break;

            case 6:  // sample

                cout << "Sample Record is" << endl;

                sample\_s->display\_details(sample\_i);

                break;

            case 7:  // duplicate

                cout << "Enter Roll number to Duplicate(Enter 0 display list):";

                cin >> srch;

                if (srch == 0) {

                    display\_all(stu\_list, stu\_info);

                    cout << "Enter Roll number to Duplicate:";

                    cin >> srch;

                }

                ind = search\_stu(srch, stu\_list);

                if (ind == -1)

                    cout << "No Record Found to Duplicate" << endl;

                else {

                    stu\_info[current] = new information(stu\_info[ind]);

                    stu\_list[current] = new student(stu\_list[ind]);

                    current++;

                    cout << "Record Duplicated Successfully, We recommend to change its details." << endl;

                }

                break;

            case 8:

                save\_data(stu\_list, stu\_info);

                cout << "Thank you for using Application:)" << endl;

                while\_control = false;

                break;

            default:

                cout << "Enter a Valid Choice." << endl;

                break;

        }

        cout << endl;

    }

    return 0;

}

**OUTPUT:**

