**1. Задание.**

Описать класс «студенческая группа». Предусмотреть возможность работы с переменным числом студентов, поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям. Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

**2. Описание основных структур данных и алгоритмов, используемых в работе**

Определяется класс Student, моделирующий отдельного студента и StudentGroup, моделирующий группу студентов.

Метод printStudentData() выводит данные о студенте.

Метод addStudent() запрашивает данные у пользователя и в соответствии с ними добавляет студента в группу.

Метод findStudent() находит запись о студенте по введенным данным либы выводит сообщение, что такой записи нет.

Метод deleteStudent() удаляет запись о студенте.

Метод sortStudents() сортирует записи о студентах в соответствии с введенными пользователем данными.

В int main() создано меню, позволяющее пользователю добавлять и удалять записи, а также сортировать их по разным полям.

**3. Описание интерфейса пользователя программы**

После запуска программы, выводится следующая подсказка:

*Операции:*

*1 - Добавить запись.*

*2 - Удалить запись.*

*3 - Найти запись.*

*4 - Отсортировать таблицу.*

*0 - Завершить работу.*

котораю предлагает пользователю выбрать операцию или завершить работу.

В соответствии с выбранной операцией пользователю также выводятся подсказки с необходимыми действиями для ее выполнения. Например:

*"Введите значения поля: "*

**4. Контрольный пример и результаты тестирования**

Операции:

1 - Добавить запись.

2 - Удалить запись.

3 - Найти запись.

4 - Отсортировать таблицу.

0 - Завершить работу.

1

Имя: Иван

Фамилия: Иванов

Дата рождения: 29.01.2000

Номер телефона: +79785847656

Число студентов в списке: 1.

Операции:

1 - Добавить запись.

2 - Удалить запись.

3 - Найти запись.

4 - Отсортировать таблицу.

0 - Завершить работу.

1

Имя: Виктор

Фамилия: Петров

Дата рождения: 12.12.1992

Номер телефона: +79788546514

Число студентов в списке: 2.

Операции:

1 - Добавить запись.

2 - Удалить запись.

3 - Найти запись.

4 - Отсортировать таблицу.

0 - Завершить работу.

4

Отсортировать по:

1 - имени;

2 - фамилии;

3 - дате рождения;

4 - номеру телефона;

0 - отмена.

1

1) Виктор Петров 12.12.1992 +79788546514

2) Иван Иванов 29.01.2000 +79785847656

Операции:

1 - Добавить запись.

2 - Удалить запись.

3 - Найти запись.

4 - Отсортировать таблицу.

0 - Завершить работу.

4

Отсортировать по:

1 - имени;

2 - фамилии;

3 - дате рождения;

4 - номеру телефона;

0 - отмена.

2

1) Иван Иваноа 29.01.2000 +79785847656

2) Виктор Петров 12.12.1992 +79788546514

Операции:

1 - Добавить запись.

2 - Удалить запись.

3 - Найти запись.

4 - Отсортировать таблицу.

0 - Завершить работу.

0

**5. Листинг программы**

#include <iostream>

#include <algorithm>

#include <string>

#include <vector>

using namespace std;

class Student

{

private:

string name = "Иван";

string last\_name = "Иванов";

string birth\_date = "01.01.2000";

string phone\_number = "+71111111111";

void printStudentData();

friend class StudentGroup;

friend bool nameSort(Student a, Student b);

friend bool lastNameSort(Student a, Student b);

friend bool birthDateSort(Student a, Student b);

friend bool phoneNumberSort(Student a, Student b);

};

class StudentGroup

{

private:

int n = 0;

vector <Student> arr;

public:

void addStudent(string name, string last\_name, string birth\_date, string phone\_number);

void sortStudents(int field);

int findStudent(int field, string key);

bool deleteStudent(int field, string key);

};

bool nameSort(Student a, Student b)

{

return a.name < b.name;

}

bool lastNameSort(Student a, Student b)

{

return a.last\_name < b.last\_name;

}

bool birthDateSort(Student a, Student b)

{

return a.birth\_date < b.birth\_date;

}

bool phoneNumberSort(Student a, Student b)

{

return a.phone\_number < b.phone\_number;

}

void Student::printStudentData()

{

cout << name << " ";

cout << last\_name << " ";

cout << birth\_date << " ";

cout << phone\_number << " ";

return;

}

void StudentGroup::addStudent(string name, string last\_name, string birth\_date, string phone\_number)

{

Student new\_student;

new\_student.name = name;

new\_student.last\_name = last\_name;

new\_student.birth\_date = birth\_date;

new\_student.phone\_number = phone\_number;

n++;

arr.push\_back(new\_student);

cout << "Число студентов в списке: " << n << "." << endl;

}

int StudentGroup::findStudent(int field, string key)

{

if (field == 0) {

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

{

if (arr[i].name == key) {

arr[i].printStudentData();

return i;

}

}

cout << "Запись не найдена." << endl;

return -1;

}

if (field == 1) {

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

{

if (arr[i].last\_name == key) {

arr[i].printStudentData();

return i;

}

}

cout << "Запись не найдена." << endl;

return -1;

}

if (field == 2) {

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

{

if (arr[i].birth\_date == key) {

arr[i].printStudentData();

return i;

}

}

cout << "Запись не найдена." << endl;

return -1;

}

if (field == 3) {

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

{

if (arr[i].phone\_number == key) {

arr[i].printStudentData();

return i;

}

}

cout << "Запись не найдена." << endl;

return -1;

}

cout << "Запись не найдена." << endl;

return -1;

}

bool StudentGroup::deleteStudent(int field, string key)

{

int index = findStudent(field, key);

if (index == -1)

{

return false;

}

else

{

n--;

arr.erase(arr.begin() + index);

cout << "- запись удалена." << endl;

cout << "Число студентов в списке: " << n << "." << endl;

return true;

}

}

void StudentGroup::sortStudents(int field)

{

if (field == 0)

{

sort(arr.begin(), arr.end(), nameSort);

}

else if (field == 1)

{

sort(arr.begin(), arr.end(), lastNameSort);

}

else if (field == 2)

{

sort(arr.begin(), arr.end(), birthDateSort);

}

else if (field == 3)

{

sort(arr.begin(), arr.end(), phoneNumberSort);

}

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

{

cout << i + 1 << ") ";

arr[i].printStudentData();

cout << endl;

}

return;

}

int main()

{

setlocale(LC\_ALL, ".UTF-8");

StudentGroup my\_group;

while(true)

{

int operation;

cout << "Операции: " << endl;

cout << "1 - Добавить запись. " << endl;

cout << "2 - Удалить запись. " << endl;

cout << "3 - Найти запись. " << endl;

cout << "4 - Отсортировать таблицу. " << endl;

cout << "0 - Завершить работу. " << endl;

cin >> operation;

cout << endl;

if (operation == 0) return 0;

else if (operation == 1)

{

string name, last\_name, birth\_date, phone\_number;

cout << "Имя: ";

cin >> name;

cout << "Фамилия: ";

cin >> last\_name;

cout << "Дата рождения: ";

cin >> birth\_date;

cout << "Номер телефона: ";

cin >> phone\_number;

cout << endl;

my\_group.addStudent(name, last\_name, birth\_date, phone\_number);

cout << endl;

}

else if (operation == 2)

{

int field;

string key;

cout << "Удалить по: " << endl;

cout << "1 - имени; " << endl;

cout << "2 - фамилии; " << endl;

cout << "3 - дате рождения; " << endl;

cout << "4 - номеру телефона; " << endl;

cout << "0 - отмена. " << endl;

cin >> field;

if (field != 0)

{

cout << "Введите значения поля: ";

cin >> key;

cout << endl;

my\_group.deleteStudent(field - 1, key);

}

}

else if (operation == 3)

{

int field;

string key;

cout << "Найти по: " << endl;

cout << "1 - имени; " << endl;

cout << "2 - фамилии; " << endl;

cout << "3 - дате рождения; " << endl;

cout << "4 - номеру телефона; " << endl;

cout << "0 - отмена. " << endl;

cin >> field;

if (field != 0)

{

cout << "Введите значения поля: ";

cin >> key;

cout << endl;

my\_group.findStudent(field - 1, key);

}

}

else if (operation == 4)

{

int field;

cout << "Отсортировать по: " << endl;

cout << "1 - имени; " << endl;

cout << "2 - фамилии; " << endl;

cout << "3 - дате рождения; " << endl;

cout << "4 - номеру телефона; " << endl;

cout << "0 - отмена. " << endl;

cin >> field;

if (field != 0)

{

my\_group.sortStudents(field - 1);

}

}

else cout << "Неверный номер операции..." << endl;

}

}