МИНИСТЕРСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ СВЯЗИ»

ФАКУЛЬТЕТ ЭЛЕКТРОСВЯЗИ

КАФЕДРА ПОСТ

Основы и методологии программирования

Отчет по лабораторной работе №15

Выполнила студентка гр. АП491 Шкундич А.А.

Руководитель Рогалевич П.И.

Минск 2024

**«**Написание программ с использованием контейнеров STL**»**

# Цель: освоить технологию обобщенного программирования с использованием библиотеки стандартных шаблонов (STL) языка C++.

Код

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <algorithm>

#include <math.h>

using namespace std;

struct STUDENT {

string name;

string surname;

int group;

string phone;

int marks[5];

int medMark;

};

int calcMedMark(int marks[5]);

void printStudents(vector <STUDENT> students);

void searchStudents(string searchStr, vector <STUDENT> students);

void addStudent(vector <STUDENT>& students);

void deleteStudent(int number, vector <STUDENT>& students);

void sortStudents(vector <STUDENT>& students);

bool sortSurnameAZ(STUDENT &a, STUDENT &b);

bool sortSurnameZA(STUDENT& a, STUDENT& b);

bool sortGroup(STUDENT& a, STUDENT& b);

bool sortMarkMinMAx(STUDENT& a, STUDENT& b);

bool sortMarkMaxMin(STUDENT& a, STUDENT& b);

int main() {

vector<STUDENT> students;

STUDENT student1 = {

"Mary-Alice",

"Smith",

144,

"80335656444",

{10, 6, 4, 5, 8}

};

STUDENT student2 = {

"Sofia",

"Chappil",

144,

"80445556868",

{10, 10, 8, 5, 9}

};

STUDENT student3 = {

"John",

"Spencer",

144,

"80297558046",

{10, 10, 2, 5, 9}

};

student1.medMark = calcMedMark(student1.marks);

student2.medMark = calcMedMark(student2.marks);

student3.medMark = calcMedMark(student3.marks);

students.push\_back(student1);

students.push\_back(student2);

students.push\_back(student3);

int option, studentNumberToDelete;

string searchStr;

do {

cout << "\nChoose an option\n\t1. Print all students\n\t2. Search student by phone\n\t3. Add new student\n\t4. Delete a student\n\t5. Sort students list\n\tPrint other number to exit\n > ";

cin >> option;

switch (option)

{

case 1:

printStudents(students);

break;

case 2:

cout << "Phone for searching > ";

cin.ignore();

getline(cin, searchStr);

searchStudents(searchStr, students);

break;

case 3:

addStudent(students);

break;

case 4:

cout << "Student number > ";

cin >> studentNumberToDelete;

break;

case 5:

sortStudents(students);

break;

default:

cout << "Exit..." << endl;

break;

}

} while (option > 0 && option <= 5);

}

int calcMedMark(int marks[5]) {

double medMark = 0;

for (int i = 0; i < 5; i++) {

medMark += marks[i];

}

return round(medMark / 5);

}

void printStudents(vector <STUDENT> students) {

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

cout << students[i].name

<< " " << students[i].surname

<< ", group " << students[i].group

<< " , tel: " << students[i].phone

<< ", medium mark: " << students[i].medMark << endl;

}

}

void searchStudents(string searchStr, vector <STUDENT> students) {

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

if (students[i].phone.find(searchStr) != string::npos) {

cout << students[i].name << " " << students[i].surname << " , tel: " << students[i].phone << endl;

}

}

}

void addStudent(vector <STUDENT>& students) {

STUDENT newStudent;

cout << "Name: ";

cin.ignore();

getline(cin, newStudent.name);

cout << "Surname: ";

getline(cin, newStudent.surname);

cout << "Group: ";

cin >> newStudent.group;

cout << "Phone: ";

cin.ignore();

getline(cin, newStudent.phone);

cout << "Marks:\n";

for (int i = 0; i < 5; i++) {

int mark;

cout << "> ";

cin >> mark;

newStudent.marks[i] = mark;

}

newStudent.medMark = calcMedMark(newStudent.marks);

students.push\_back(newStudent);

}

void deleteStudent(int number, vector <STUDENT>& students) {

if (number >= students.size() || number < 0) {

cout << "There are no student with index " << number << endl;

return;

}

vector<STUDENT>::iterator iter = students.begin();

students.erase(iter + number);

}

bool sortSurnameAZ(STUDENT& a, STUDENT& b) {

return a.surname < b.surname;

}

bool sortSurnameZA(STUDENT& a, STUDENT& b) {

return a.surname > b.surname;

}

bool sortGroup(STUDENT& a, STUDENT& b) {

return a.group < b.group;

}

bool sortMarkMinMAx(STUDENT& a, STUDENT& b) {

return a.medMark < b.medMark;

}

bool sortMarkMaxMin(STUDENT& a, STUDENT& b) {

return a.medMark > b.medMark;

}

void sortStudents(vector <STUDENT>& students) {

int sortType;

cout << "Choose param for sorting\n\t1. Surname A-Z\n\t2. Surname Z-A\n\t3. Group\n\t4. Medium mark 1-10\n\t5. Medium mark 10-1\n > ";

cin >> sortType;

if (sortType <= 0 || sortType > 5) {

cout << "There is no such param for sorting. Please, try again" << endl;

}

vector<STUDENT>::iterator iter;

switch (sortType)

{

case 1:

sort(students.begin(), students.end(), sortSurnameAZ);

break;

case 2:

sort(students.begin(), students.end(), sortSurnameZA);

break;

case 3:

sort(students.begin(), students.end(), sortGroup);

break;

case 4:

sort(students.begin(), students.end(), sortMarkMinMAx);

break;

default:

sort(students.begin(), students.end(), sortMarkMaxMin);

break;

}

}