**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное

учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направление информатика и вычислительная техника

Отчет

по лабораторной работе №5

по дисциплине

**«**Программирование**»**

**Обработка файлов и структур**

Вариант - 13

Выполнил:

Студент группы 8В31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Моцыборова А.А.

Проверил:

Программист \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Воеводин К.Д.

Цель работы

Освоить загрузку и сохранение структур данных в файлах, реализовать диалоговый пользовательский интерфейс.

Задание

1. Составить алгоритм решения задачи в соответствии с заданным вариантом.

2. На основе созданного алгоритма разработать приложение.

3. Произвести тестирование разработанного приложения.

Требования к программе:

Произвести чтение массива структур данных. Структура должна иметь не менее 5 полей не менее 2 различных типов.

Должно быть реализовано меню действий пользователя, включающее как минимум следующий набор пунктов: загрузка перечня структур, вывод на экран всех структур, поиск по значению одного из полей, добавление новой структуры, удаление структуры по номеру, сохранение перечня структур.

Должна быть обеспечена корректная работа с динамической памятью.

Чтение данных организовать из файла, диалог с пользователем – в консоли, сохранение изменённых данных – в файл.

Предусмотреть форматированный вывод данных в удобном для пользователя виде.

Блочность приложения должна быть реализована с использованием функций.

Вариант структуры: Студент (ФИО, курс, факультет, средний балл, возраст).Ход работы

На рисунках 1 – 7 демонстрируется отладка программы

Рисунок 1 – Ввод исходной матрицы

Рисунок 2 – Определение индексов строк, начинающихся с нуля

Рисунок 3 – Определение отрицательных элементов и сохранение их в массиве B

Рисунок 4 – Печать преобразованной матрицы Z

Рисунок 5 – Преобразованная матрица полностью выведенаРисунок 6 – Выведен массив отрицательных элементов

Вывод

Освоили загрузку и сохранение структур данных в файлах, реализовали диалоговый пользовательский интерфейс.

Приложение 1 – Код программы

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

struct Student {

std::string fullName;

int course;

std::string faculty;

double gradePointAverage;

int age;

};

void loadStudents(std::vector<Student>& students) {

std::ifstream file("D:\\tpu\\programming\\lab5\\students.txt.txt", std::ios::in);

if (file.is\_open()) {

Student student;

while (file >> student.fullName >> student.course >> student.faculty >> student.gradePointAverage >> student.age) {

students.push\_back(student);

}

file.close();

std::cout << "The students are uploaded from a file." << std::endl;

} else {

std::cout << "File opening error." << std::endl;

}

}

void displayStudents(const std::vector<Student>& students) {

if (students.empty()) {

std::cout << "No students found." << std::endl;

return;

}

std::cout << "List of students:" << std::endl;

for (const auto& student : students) {

std::cout << "Full name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", Age: " << student.age << std::endl;

}

}

void searchStudents(const std::vector<Student>& students) {

std::string searchField;

std::cout << "Enter the search field (fullName, course, faculty, gradePointAverage, age): ";

std::cin >> searchField;

bool found = false;

for (const auto& student : students) {

if (searchField == "fullName" && student.fullName == searchField) {

std::cout << "Fuul name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", age: " << student.age << std::endl;

found = true;

} else if (searchField == "course" && std::to\_string(student.course) == searchField) {

std::cout << "Full name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", age: " << student.age << std::endl;

found = true;

} else if (searchField == "faculty" && student.faculty == searchField) {

std::cout << "Full name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", age: " << student.age << std::endl;

found = true;

} else if (searchField == "gradePointAverage" && std::to\_string(student.gradePointAverage) == searchField) {

std::cout << "Full name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", age: " << student.age << std::endl;

found = true;

} else if (searchField == "age" && std::to\_string(student.age) == searchField) {

std::cout << "Full name: " << student.fullName << ", Course: " << student.course

<< ", Faculty: " << student.faculty << ", Grade Point Average: " << student.gradePointAverage

<< ", age: " << student.age << std::endl;

found = true;

}

}

if (!found) {

std::cout << "Students with the specified field value were not found." << std::endl;

}

}

void addStudent(std::vector<Student>& students) {

Student newStudent;

std::cout << "Enter the information about the student:" << std::endl;

std::cout << "Full name: ";

std::getline(std::cin >> std::ws, newStudent.fullName);

std::cout << "Course: ";

std::cin >> newStudent.course;

std::cout << "Faculty: ";

std::getline(std::cin >> std::ws, newStudent.faculty);

std::cout << "Grade Point Average: ";

std::cin >> newStudent.gradePointAverage;

std::cout << "age: ";

std::cin >> newStudent.age;

students.push\_back(newStudent);

std::cout << "A new student has been added." << std::endl;

}

void deleteStudent(std::vector<Student>& students) {

int index;

std::cout << "Enter the student's index to delete: ";

std::cin >> index;

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

std::cout << "Invalid index." << std::endl;

return;

}

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

std::cout << "The student has been removed." << std::endl;

}

void saveStudents(const std::vector<Student>& students) {

std::ofstream file("students.txt");

if (file.is\_open()) {

for (const auto& student : students) {

file << student.fullName << " " << student.course << " " << student.faculty << " "

<< student.gradePointAverage << " " << student.age << std::endl;

}

file.close();

std::cout << "The students are saved to a file." << std::endl;

} else {

std::cout << "Error opening the file." << std::endl;

}

}

int main() {

std::vector<Student> students;

int choice;

while (true) {

std::cout << "Student Management Menu:" << std::endl;

std::cout << "1. Upload students" << std::endl;

std::cout << "2. Display students" << std::endl;

std::cout << "3. Student Search" << std::endl;

std::cout << "4. Add a student" << std::endl;

std::cout << "5. Delete a student" << std::endl;

std::cout << "6. Save students" << std::endl;

std::cout << "7. Exit" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

loadStudents(students);

break;

case 2:

displayStudents(students);

break;

case 3:

searchStudents(students);

break;

case 4:

addStudent(students);

break;

case 5:

deleteStudent(students);

break;

case 6:

saveStudents(students);

break;

case 7:

std::cout << "Exit the program..." << std::endl;

return 0;

default:

std::cout << "Wrong choice. Try again." << std::endl;

break;

}

}

}Ссылка на GitHub: ф<https://github.com/G1djet/Prog-lab5.git>