|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ**\_\_\_\_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** **\_\_\_\_*ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***\_\_\_\_\_\_

**О Т Ч Е Т**

**УЧЕБНАЯ ПРАКТИКА**

**«Ознакомительная практика»**

Студент гр. ИУК4-11Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Суриков Н.С.\_\_)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_Пчелинцева Н.И.\_)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка практики \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_Пчелинцева Н.И.\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_Амеличева К.А.\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_Гагарин Ю.Е.\_\_\_)

(подпись) (Ф.И.О.)

Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана*** *(национальный исследовательский университет)»* ***(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУК4\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Гагарин Ю.Е.)

« 04 » сентября 2023 г.

**ЗАДАНИЕ**

**на УЧЕБНУЮ, ОЗНАКОМИТЕЛЬНУЮ ПРАКТИКУ**

За время прохождения практики студенту необходимо:

1. Согласовать предметную область и тему проекта, закрепить сроки и требования по различным этапам реализации проекта, оформить требования в виде технического задания, утвердить техническое задание, оценить качество составленных документов.
2. Спроектировать структуру разрабатываемого приложения, продумать интерфейс взаимодействия пользователя с системой, оформить результаты работы в виде блок-схем, осуществить выбор библиотек и других технологий разработки.
3. Разработать и реализовать алгоритмы функционирования приложения, структуры, систем передачи информации, технологий обработки информации и интерфейса взаимодействия пользователя с системой, редактировать техническую документацию в соответствии с требованиями ГОСТ.
4. Разработать программное приложение для работы с файлами данных, содержащих информацию о студентах учебной группы, реализующее упорядочивание списка студентов по году рождения.
5. Подготовить отчет и защитить результаты практики.

Дата выдачи задания « 04 » сентября 2023 г.

|  |  |  |
| --- | --- | --- |
| Руководитель практики |  | Пчелинцева Н.И. |
|  |  |  |
| Задание получил студент группы ИУК4-11Б |  | Суриков Н.С. |

Оглавление

[ВВЕДЕНИЕ 4](#__RefHeading___Toc45272351)

[1. ПОДГОТОВИТЕЛЬНЫЙ ЭТАП 5](#__RefHeading___Toc45272352)

[1.1. Анализ задачи и её постановка 5](#__RefHeading___Toc45272353)

[1.2. Анализ входных и выходных данных 6](#__RefHeading___Toc14615_3241078693)

[1.3 Выбор метода решения задачи 6](#__RefHeading___Toc14623_3241078693)

[1.4 Технические требования 6](#__RefHeading___Toc14625_3241078693)

[2. ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП 8](#__RefHeading___Toc14609_3241078693)

[2.1 Принцип работы приложения 8](#__RefHeading___Toc14627_3241078693)

[2.2 Разработка алгоритма и структур данных 8](#__RefHeading___Toc14621_3241078693)

[2.3 Программная реализация алгоритма 9](#__RefHeading___Toc14629_3241078693)

[2.4 Тестирование и отладка 12](#__RefHeading___Toc14631_3241078693)

[ЗАКЛЮЧЕНИЕ 15](#__RefHeading___Toc14611_3241078693)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#__RefHeading___Toc14613_3241078693)

[ПРИЛОЖЕНИЕ 18](#__RefHeading___Toc45272357)

[Листинг программы 18](#__RefHeading___Toc17383_3241078693)

# ВВЕДЕНИЕ

Целью учебной практики является знакомство с основами будущей профессии, получение сведений о специфике избранной специальности, овладение первичными профессиональными умениями и навыками:

* по постановке задачи на разработку требований к подсистемам и контроль их качества,
* по применению методов оценивания временной и емкостной сложности программного обеспечения,
* разработке требований к системе в целом,

а также подготовка обучающегося к осознанному и углубленному практическому изучению учебных дисциплин.

Для достижения поставленной цели решаются следующие задачи:

* осуществить выбор библиотек, среды разработки, системы контроля версий, обосновать соответствующий выбор,
* подготовить доклад с использованием средств визуализации (презентации, графики, блок-схемы, UML-диаграммы и пр.), в котором будет представлен отчёт студента о проделанной работе, общая информация о разработанном проекте, указаны его преимущества, недостатки и перспективы дальнейшего развития.

# 1. ПОДГОТОВИТЕЛЬНЫЙ ЭТАП

1.1. Анализ задачи и её постановка

Задачей учебной практики является создание консольного приложения, реализующего обработку базы данных студентов учебной группы.

Приложение должно обеспечивать следующее:

* 1. Ввод информации с клавиатуры
  2. Ввод информации из ранее созданного текстового файла
  3. Ввод информации из ранее созданного бинарного файла
  4. Вывод данных в виде таблицы на экран
  5. Вывод данных в файл
  6. Распечатку упорядоченного по году рождения списка студентов учебной группы
  7. Перевод БД в текстовый файл
  8. Перевод БД из текстового файла в бинарный
  9. Добавление записи в БД
  10. Изменение записи в БД
  11. Удаление записи в БД
  12. Сортировка БД
  13. Выход из приложения

Данные задачи были систематизированы для создания меню состоящего из 9 пунктов:

* 1. Добавление студента
  2. Изменение студента
  3. Удаление студента
  4. Вывод списка студентов
  5. Вывод сортированного списка студентов
  6. Импорт списка студентов
  7. Экспорт списка студентов
  8. Трансформирование текстового файла списка студентов в бинарный
  9. Выход

1.2. Анализ входных и выходных данных

Входные данные представляют собой информацию (символьную или числовую), вводимую пользователем с клавиатуры или получаемую из файла. После ее обработки приложением она представляется в виде базы данных студентов учебной группы, которая может быть изменена, выведена на экран в виде таблицы или экспортирована. Каждый студент в БД имеет фамилию, имя, отчество, год рождения, курс обучения и оценки по 3 предметам.

1.3 Выбор метода решения задачи

В качестве средства для разработки приложения был выбран высокоуровневый язык программирования C++. Его главное преимущество - высокое быстродействие по сравнению с другими ЯП. При создании приложения были задействованы некоторые заголовочные файлы из Стандартной Библиотеки C++: iostream (для ввода данных с клавиатуры и вывода их в консоль), fstream (для импорта/экспорта данных, содержащихся в файлах), iomanip (для организации вывода данных в табличном виде).

Средой программирования был выбран Visual Studio Code. К его достоинствам относится высокая модульность, представленная различными расширениями, и широкий выбор настроек. Всё это позволяет гибко настроить рабочее пространство. В качестве компилятора был использован clang++ 17 версии и последний на данный момент стандарт C++23.

Так как данные в БД на каждого студента идентичные, то их хранение было организовано в виде массива структур.

1.4 Технические требования

Приложение разрабатывалось на одном из дистрибутивов Linux и может быть легко запущено на любой Unix подобной OS. Запуск на Windows так же возможен, но не был протестирован. Минимальные системные требования соответствуют системным требованиям для запуска соответствующей ОС.

# 2. ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП

2.1 Принцип работы приложения

Приложение реализует интерфейс в виде консольного меню для взаимодействия с базой данных студентов учебной группы. Оно позволяет добавлять, изменять, удалять запись о студенте, выводить предварительно отсортированный в заданном порядке список студентов на экран, импортировать и экспортировать базу данных для хранения в текстовый или бинарный файл.

2.2 Разработка алгоритма и структур данных

Для удобства хранения и обработки базы данных она была реализована с помощью массива структур, каждая из которых представляет собой 3 символьных массива (для фамилии, имени и отчества), 2 целочисленных поля (для года рождения и курса обучения) и целочисленный массив размером в 3 ячейки (для оценок по предметам). Блок-схема структуры представлена на рисунке 1. За счет использования такого подхода достигается четкая структуризация данных, а также повышается читаемость кода по сравнению, например, с более тривиальной реализацией БД в виде двумерного массива.

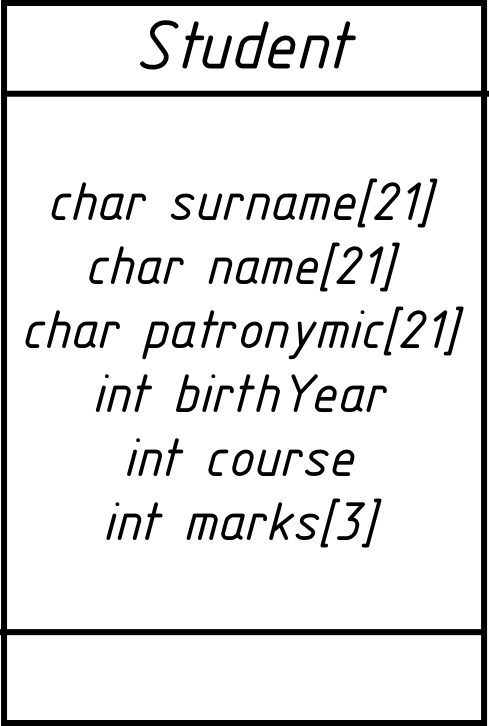


Рис. 1. Блок-схема структуры Student

За каждый пункт меню отвечает отдельная функция, причем каждая из них выполняет только одну конкретную задачу, заданную непосредственно пунктом меню. К примеру, функция для пункта 5 (вывод сортированного списка студентов) выполняет исключительно те действия, которые непосредственно связаны с выводом: запрос способа сортировки списка и сам вывод. За сортировку отвечает уже другая функция. Таким образом достигается высокая степень модульности приложения, значительно облегчается процесс его отладки и внедрения новых функциональностей при необходимости в дальнейшем.

2.3 Программная реализация алгоритма

Основной код приложения находится в файле main.cpp. В файле заголовочном файле main.h находятся прототипы функций и объявление структуры. Функция main, находящаяся в главном файле, реализовывает отображение меню и взаимодействие пользователя с ним. Как уже было отмечено, для каждого пункта меню была написана отдельная функция:

* Добавление студента — addStudent
* Изменение студента — changeStudent
* Удаление студента — deleteStudent
* Вывод списка студентов — displayStudentList
* Вывод сортированного списка студентов — displaySortedStudentList
* Импорт списка студентов — importStudentList
* Экспорт списка студентов —exportStudentList
* Трансформирование текстового файла списка студентов в бинарный — transformStudentList

Также были реализовано множество небольщих вспомогательных функций для сортировки списка студентов по разным полям, для получения и обработки пользовательского ввода, для получения пути файла и тд.

Алгоритмы работы этих функций следующие:

1. addStudent. Функция запрашивает количество студентов, которое требуется добавить, затем создает нужное количество структур Student, после чего заполняет их значениями, введенными пользователем. Затем они добавляются в конец массива структур, представляющего БД, а также переменная, хранящая количество записей в БД, увеличивается на количество добавленных студентов.
2. changeStudent. Функция выводит список студентов, запрашивает номер студента для изменения и номер поля, который требуется изменить. Затем структура с нужным номером перезаписывается в соответствии с введенными пользователем данными.
3. deleteStudent. Функция выводит список студентов, запрашивает у пользователя номер студента, которого требуется удалить, а затем удаляет соответствующую структуру Student из массива и обновляет количество записей в БД.
4. transformStudentList. Функция создает новый массив структур, представляющий БД, в который записываются данные из исходного текстового файла, который указывает пользователь. Чтение файла происходит посредством вызова функции importFromFile. Затем запрашивается конечный бинарный файл, после чего созданный массив записывается в этот файл. Запись происходит с помощью вызова функции exportToFile.
5. importFromFile. Функция запрашивает путь к файлу, из которого требуется импортировать данные. Затем она открывает этот файл и считывает из него данные, заполняя ими массив структур Student. После завершения чтения файла количество записей в БД обновляется.
6. exportToFile. Функция запрашивает путь к файлу, в который требуется экспортировать данные. Затем она открывает этот файл и записывает в него данные из массива структур Student.
7. displayStudentList. Функция предлагает выбрать пользователю куда вывести все записи из массива структур Student а затем выводит их.
8. printStudentList. Функция выводит в консоль все записи из массива структур Student.
9. displaySortedStudentList. Функция сначала сортирует массив структур Student, а затем выводит его на экран.
10. sortStudentList. Функция сортирует массив структур Student в зависимости от выбранного пользователем режима сортировки.
11. specifyFilePath. Функция запрашивает у пользователя путь к файлу и сохраняет его.
12. getInput. Функция запрашивает у пользователя ввод и возвращает его. Имеет две перегрузки для числового и текстового ввода.
13. swapStudents. Функция меняет местами две структуры Student в массиве.
14. sortByName, sortByPatronymic, sortByBirthYear, sortByCourse, sortByMark. Эти функции сортируют массив структур Student по соответствующим полям.
15. importStudentList. Функция предлагает пользователю выбрать из какого типа файлов производить считывание, запрашивает путь к файлу и считывает данные.
16. exportStudentList. Функция предлагает пользователю выбрать в какого типа файл производить запись, запрашивает путь к файлу и записывает в него данные.
17. isStudents. Функция проверяет, есть ли студенты в списке.
18. clearInput. Функция очищает ввод пользователя.
19. clearDisplay. Функция очищает консоль.
20. printMenu. Функция отображает меню программы.

2.4 Тестирование и отладка

Тестирование производилось на заранее заготовленных данных как комплексно для всей программы целиком так и для отдельных её компонентов в разных сценариях использования. В процессе отладки были выявлены некорректные ответы программы при определённых значениях и это было учтено при дальнейшем рефакторинге кода. В конечном итоге удалось достичь максимальной отказоустойчивости программы.

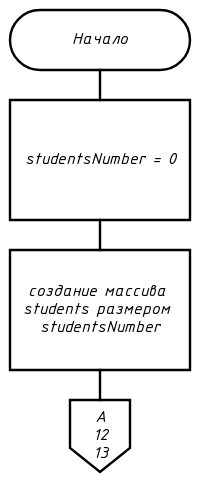
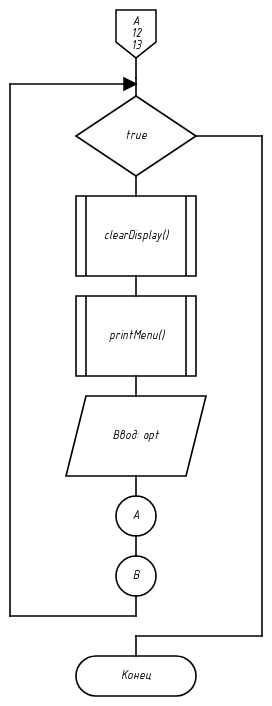
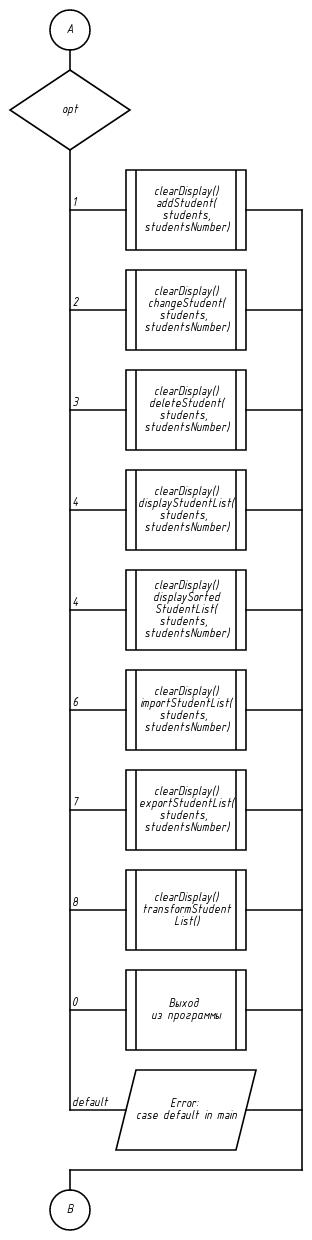
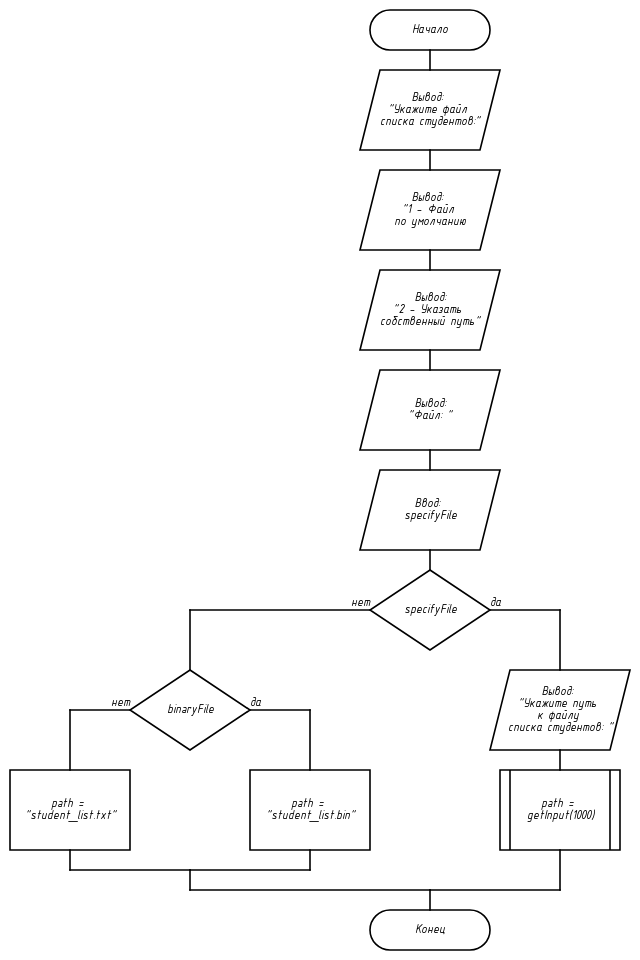


Рис. 2. Блок-схема функции main(часть 1)

Рис. 3. Блок-схема функции main(часть 2)

Рис. 4. Блок-схема функции main(часть 3)

Рис. 5. Блок-схема функции specifyFilePath

# ЗАКЛЮЧЕНИЕ

В ходе выполнения учебной практики было разработано приложение, реализующее интерфейс взаимодействия с базой данных студентов учебной группы. В результате были выполнены следующие задачи:

* создание консольного меню для работы с данными, хранящимися в БД: создание, изменение и удаление записей о студентах
* организация ввода данных с клавиатуры и их обработки для дальнейшей работы
* организация импорта\экспорта БД в текстовый и бинарный файл, а также трансформирование текстового файла БД в бинарный
* организация вывода отсортированного списка студентов из БД в консоль или текстовый файл

В ходе разработки приложения мною были изучены структуры, принцип работы с файловой системой, принцип организации хранения и обработки информации в БД, одномерные массивы (в т.ч. символьные).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово : Кемеровский государственный университет культуры и искусств, 2013. - 36 с. : ил., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=274218>

2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 c.— Режим доступа: <http://www.iprbookshop.ru/66817.html>

3. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс]: учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово: Кемеровский технологический институт пищевой промышленности, 2009. — 123 c. — Режим доступа: <http://www.iprbookshop.ru/14381.html>

4. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс]: учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола: Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 c. — Режим доступа: <http://www.iprbookshop.ru/22586.html>

5. Мокий, М.С. Методология научных исследований[Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.

6. Рогов, В.А. Методика и практика технических экспериментов[Текст]: учеб.пособие / В.А. Рогов, А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с.

7. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: <http://biblioclub.ru/index.php?page=book&id=89693>.

8. Моделирование систем[Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.

9. Порсев, Е. Г. Организация и планирование экспериментов [Электронный ресурс]: учебное пособие / Е. Г. Порсев.— Новосибирск : Новосибирский государственный технический университет, 2010. — 155 c. — Режим доступа: <http://www.iprbookshop.ru/45415.html>

10. Matthew Scarpino (2019). Algorithmic Trading with Interactive Brokers (Python and C++).

11. Курс лекций доцента кафедры ФН1-КФ Пчелинцевой Н.И. 19

12. Программирование на языке высокого уровня С/С++ [Электронный ресурс]: конспект лекций / – Электрон. текстовые данные. – М.: Московский государственный строительный университет, Ай Пи Эр Медиа, ЭБС 2016. – 140 c. – Режим доступа: http://www.iprbookshop.ru/48037 АСВ,

13. Новиков, Б. А. Основы технологий баз данных / Б. А. Новиков ; под редакцией Е. В. Рогова. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978- 5-94074-820-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/123699>

14. Базы данных : учебное пособие / . — Саратов : Научная книга, 2012. — 158 c. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/6261.html

15. Ревунков, Г. И. Базы и банки данных : учебное пособие / Г. И. Ревунков. — Москва : МГТУ им. Н.Э. Баумана, 2011. — 68 с. — Текст : электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/52425

# ПРИЛОЖЕНИЕ

Листинг программы

1 #include <iostream>

2 #include <fstream>

3

4 #include <iomanip> // для setw()

5 #include <cstring>

6 #include <limits> // для numeric\_limits

7

8 #include <stdio.h> // для getchar()

9 #include <format> // для format()

10

11 using namespace std;

12

13 struct Student

14 {

15 char surname[21];

16 char name[21];

17 char patronymic[21];

18 int birthYear;

19 int course;

20 int marks[3];

21 };

22

23 // Указатель для передачи шаблонов значений в функцию getInput

24 int \*allowedValues\_i{nullptr};

25

26 // Функция для импорта данных студентов из файла

27 int importFromFile(Student \*&students, int &studentsNumber, const char \*path, int binaryFile);

28

29 // Функция для экспорта данных студентов в файл

30 void exportToFile(Student \*&students, int &studentsNumber, const char \*path, int binaryFile);

31

32 // Функции для отображения списка студентов

33 void displayStudentList(Student \*students, int studentsNumber); // Отображает список студентов в консоли

34 void printStudentList(Student \*students, int studentsNumber); // Печатает список студентов

35 void displaySortedStudentList(Student \*students, int studentsNumber); // Отображает отсортированный список студентов

36

37 // Функция для сортировки списка студентов

38 void sortStudentList(Student \*&students, int studentsNumber, int sortMode, bool descendingSort);

39

40 // Функция для указания пути к файлу

41 void specifyFilePath(char \*path, int binaryFile);

42

43 // Функция для получения ввода от пользователя

44 int getInput(int \*templ = nullptr, int templNum = 0); // Получает целочисленный ввод от пользователя

45 char \*getInput(int strLen); // Получает строковый ввод от пользователя

46

47 // Функция для обмена местами двух студентов

48 void swapStudents(Student &a, Student &b);

49

50 // Функции для сортировки студентов по различным критериям

51 void sortByName(Student \*&students, int j, bool ascendingSort);

52 void sortByPatronymic(Student \*&students, int j, bool ascendingSort);

53 void sortByBirthYear(Student \*&students, int j, bool ascendingSort);

54 void sortByCourse(Student \*&students, int j, bool ascendingSort);

55 void sortByMark(Student \*&students, int j, bool ascendingSort, int markNumber);

56

57 // Функции для добавления, изменения и удаления студента

58 void addStudent(Student \*&students, int &studentsNumber);

59 void changeStudent(Student \*&students, int studentsNumber);

60 void deleteStudent(Student \*&students, int &studentsNumber);

61

62 // Функции для импорта и экспорта списка студентов

63 void importStudentList(Student \*&students, int &studentsNumber);

64 void exportStudentList(Student \*students, int studentsNumber);

65

66 // Функция для проверки, есть ли студенты в списке

67 bool isStudents(int studentsNumber);

68

69 // Функция для преобразования списка студентов

70 void transformStudentList();

71

72 // Функции для очистки ввода и вывода

73 void clearInput(); // Очищает ввод пользователя

74 void clearDisplay(); // Очищает консоль

75

76 // Функция для отображения меню

77 void printMenu();

78

79 int main()

80 {

81 setlocale(LC\_ALL, "Russian");

82

83 int studentsNumber{0};

84 Student \*students = new Student[studentsNumber];

85 while (true)

86 {

87 int opt;

88 clearDisplay();

89 printMenu();

90

91 allowedValues\_i = new int[10]{1, 2, 3, 4, 5, 6, 7, 8, 0};

92 opt = getInput(allowedValues\_i, 10);

93 delete[] allowedValues\_i;

94 allowedValues\_i = nullptr;

95

96 switch (opt)

97 {

98 case 1:

99 clearDisplay();

100 addStudent(students, studentsNumber);

101 break;

102 case 2:

103 clearDisplay();

104 changeStudent(students, studentsNumber);

105 break;

106 case 3:

107 clearDisplay();

108 deleteStudent(students, studentsNumber);

109 break;

110 case 4:

111 clearDisplay();

112 displayStudentList(students, studentsNumber);

113 break;

114 case 5:

115 clearDisplay();

116 displaySortedStudentList(students, studentsNumber);

117 break;

118 case 6:

119 clearDisplay();

120 importStudentList(students, studentsNumber);

121 break;

122 case 7:

123 clearDisplay();

124 exportStudentList(students, studentsNumber);

125 break;

126 case 8:

127 clearDisplay();

128 transformStudentList();

129 break;

130 case 0:

131 exit(0);

132 default:

133 cerr << "\nError: case default in main\n ";

134 break;

135 }

136 }

137 }

138

139 void clearDisplay()

140 {

141 cout << "\x1B[2J\x1B[H";

142 }

143

144 void clearInput()

145 {

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

147 }

148

149 void addStudent(Student \*&students, int &studentsNumber)

150 {

151 cout << "Укажите количество студентов, которое требуется добавить (0 - отмена операции): ";

152 int studNum;

153 while (true)

154 {

155 studNum = getInput();

156 if (studNum < 0)

157 cout << "Некорректный ввод. Пожалуйста, введите правильное значение : ";

158 else if (studNum == 0)

159 {

160 cout << "\nДобавление студентов прервано.\n"

161 << endl

162 << "Для возврата в меню нажмите enter... ";

163 getchar();

164 return;

165 }

166 else

167 break;

168 }

169

170 Student \*temp = new Student[studentsNumber + studNum];

171 for (int i = 0; i < studentsNumber; i++)

172 temp[i] = students[i];

173

174 for (int i = 0; i < studNum; i++)

175 {

176 Student stud;

177 cout << "Введите фамилию студента №" << i + 1 << ": ";

178 strcpy(stud.surname, getInput(21));

179

180 cout << "Введите имя студента №" << i + 1 << ": ";

181 strcpy(stud.name, getInput(21));

182

183 cout << "Введите отчество студента №" << i + 1 << ": ";

184 strcpy(stud.patronymic, getInput(21));

185

186 cout << "Введите год рождения студента №" << i + 1 << " (1900 - 2023): ";

187 allowedValues\_i = new int[2023 - 1900 + 1];

188 for (int i = 0; i < 2023 - 1900 + 1; i++)

189 allowedValues\_i[i] = i + 1900;

190 stud.birthYear = getInput(allowedValues\_i, 2023 - 1900 + 1);

191 delete[] allowedValues\_i;

192

193 cout << "Введите курс обучения студента №" << i + 1 << " (1 - 6):";

194 allowedValues\_i = new int[6]{1, 2, 3, 4, 5, 6};

195 stud.course = getInput(allowedValues\_i, 6);

196 delete[] allowedValues\_i;

197

198 cout << "Введите оценки студента №" << i + 1 << " по 3 предметам (0 - 5) :" << endl;

199

200 cout << "Оценка 1: ";

201 allowedValues\_i = new int[6]{0, 1, 2, 3, 4, 5};

202 stud.marks[0] = getInput(allowedValues\_i, 6);

203

204 cout << "Оценка 2: ";

205 stud.marks[1] = getInput(allowedValues\_i, 6);

206

207 cout << "Оценка 3: ";

208 stud.marks[2] = getInput(allowedValues\_i, 6);

209

210 delete[] allowedValues\_i;

211 allowedValues\_i = nullptr;

212 temp[studentsNumber + i] = stud;

213 }

214 delete[] students;

215 students = temp;

216 studentsNumber += studNum;

217

218 cout << "\nДобавление студентов выполнено успешно!\n"

219 << endl;

220 cout << "Для возврата в меню нажмите enter... ";

221 getchar();

222 }

223

224 void changeStudent(Student \*&students, int studentsNumber)

225 {

226 if (!isStudents(studentsNumber))

227 return;

228

229 printStudentList(students, studentsNumber);

230

231 cout << "\nУкажите номер студента, которого требуется изменить (0 - отмема операции): ";

232

233 allowedValues\_i = new int[studentsNumber + 1];

234 for (int i = 0; i <= studentsNumber; i++)

235 allowedValues\_i[i] = i;

236 int changeStud = getInput(allowedValues\_i, studentsNumber + 1);

237 delete[] allowedValues\_i;

238 allowedValues\_i = nullptr;

239

240 if (changeStud == 0)

241 {

242 cout << "\nИзменение студента прервано.\n\nДля возврата в меню нажмите enter... ";

243 getchar();

244 return;

245 }

246

247 cout << "\nУкажите поле, которое требуется изменить:\n "

248 << "1 - Фамилия\n "

249 << "2 - Имя\n "

250 << "3 - Отчество\n "

251 << "4 - Год рождения\n "

252 << "5 - Курс обучения\n "

253 << "6 - Оценка 1\n "

254 << "7 - Оценка 2\n "

255 << "8 - Оценка 3\n ";

256 cout << "Данные для изменения: ";

257

258 allowedValues\_i = new int[8]{1, 2, 3, 4, 5, 6, 7, 8};

259 int changeMode = getInput(allowedValues\_i, 8);

260 delete[] allowedValues\_i;

261 allowedValues\_i = nullptr;

262

263 switch (changeMode)

264 {

265 case 1:

266 {

267 cout << "Введите новую фамилию студента: ";

268 strcpy(students[changeStud - 1].surname, getInput(21));

269 break;

270 }

271 case 2:

272 {

273 cout << "Введите новое имя студента: ";

274 strcpy(students[changeStud - 1].name, getInput(21));

275 break;

276 }

277 case 3:

278 {

279 cout << "Введите новое отчество студента: ";

280 strcpy(students[changeStud - 1].patronymic, getInput(21));

281 break;

282 }

283 case 4:

284 {

285 cout << "Введите новый год рождения студента (1900 - 2022):";

286 allowedValues\_i = new int[2022 - 1900 + 1];

287 for (int i = 0; i < 2022 - 1900 + 1; i++)

288 allowedValues\_i[i] = i + 1900;

289 students[changeStud - 1].birthYear = getInput(allowedValues\_i, 2022 - 1900 + 1);

290 delete[] allowedValues\_i;

291 allowedValues\_i = nullptr;

292 break;

293 }

294 case 5:

295 {

296 cout << "Введите новый курс обучения студента (1 - 6): ";

297 allowedValues\_i = new int[6]{1, 2, 3, 4, 5, 6};

298 students[changeStud - 1].course = getInput(allowedValues\_i, 6);

299 delete[] allowedValues\_i;

300 allowedValues\_i = nullptr;

301 break;

302 }

303 case 6:

304 {

305 cout << "Введите новую оценку 1 студента (0 - 5): ";

306 allowedValues\_i = new int[6]{0, 1, 2, 3, 4, 5};

307 students[changeStud - 1].marks[0] = getInput(allowedValues\_i, 6);

308 delete[] allowedValues\_i;

309 allowedValues\_i = nullptr;

310 break;

311 }

312 case 7:

313 {

314 cout << "Введите новую оценку 2 студента (0 - 5): ";

315 allowedValues\_i = new int[6]{0, 1, 2, 3, 4, 5};

316 students[changeStud - 1].marks[1] = getInput(allowedValues\_i, 6);

317 delete[] allowedValues\_i;

318 allowedValues\_i = nullptr;

319 break;

320 }

321 case 8:

322 {

323 cout << "Введите новую оценку 3 студента (0 - 5): ";

324 allowedValues\_i = new int[6]{0, 1, 2, 3, 4, 5};

325 students[changeStud - 1].marks[2] = getInput(allowedValues\_i, 6);

326 delete[] allowedValues\_i;

327 allowedValues\_i = nullptr;

328 break;

329 }

330 }

331 cout << "\nИзменение студента выполнено успешно!\n\nДля возврата в меню нажмите enter... ";

332 getchar();

333 }

334

335 void deleteStudent(Student \*&students, int &studentsNumber)

336 {

337 if (!isStudents(studentsNumber))

338 return;

339

340 printStudentList(students, studentsNumber);

341

342 cout << "\nУкажите номер студента, которого требуется удалить (0 - отмена операции): ";

343

344 allowedValues\_i = new int[studentsNumber + 1];

345 for (int i = 0; i <= studentsNumber; i++)

346 allowedValues\_i[i] = i;

347 int deleteStud = getInput(allowedValues\_i, studentsNumber + 1);

348 delete[] allowedValues\_i;

349 allowedValues\_i = nullptr;

350

351 if (deleteStud == 0)

352 {

353 cout << "\nУдаление студента прервано.\n\nДля возврата в меню нажмите enter... ";

354 getchar();

355 return;

356 }

357

358 Student \*temp = new Student[studentsNumber - 1];

359 for (int i = 0, j = 0; i < studentsNumber; i++, j++)

360 {

361 if (i == deleteStud)

362 {

363 j--;

364 continue;

365 }

366 temp[j] = students[i];

367 }

368

369 delete[] students;

370 students = temp;

371 studentsNumber--;

372 cout << "\nУдаление студента выполнено успешно!\n"

373 << endl

374 << "\nДля возврата в меню нажмите enter... ";

375 getchar(); // ожидание нажатия клавиши

376 }

377

378 void displaySortedStudentList(Student \*students, int studentsNumber)

379 {

380 if (!isStudents(studentsNumber))

381 return;

382 cout << "\nВыберите тип сортировки:\n"

383 << "1 - По фамилии\n"

384 << "2 - По имени\n"

385 << "3 - По отчеству\n"

386 << "4 - По году рождения\n"

387 << "5 - По курсу обучения\n"

388 << "6 - По оценке\n";

389 cout << "Тип сортировки: ";

390

391 allowedValues\_i = new int[6]{1, 2, 3, 4, 5, 6};

392 int sortMode = getInput(allowedValues\_i, 6) - 1;

393 delete[] allowedValues\_i;

394 allowedValues\_i = nullptr;

395

396 if (sortMode == 5)

397 {

398 cout << "\nВведите номер оценки (1 - 3), по которой требуется сортировка: ";

399 allowedValues\_i = new int[3]{1, 2, 3};

400 sortMode += getInput(allowedValues\_i, 3) - 1;

401 delete[] allowedValues\_i;

402 allowedValues\_i = nullptr;

403 }

404 cout << "\nВыберите направление сортировки:\n"

405 << "1 - По убыванию\n"

406 << "2 - По возрастанию\n";

407 cout << "Направление сортировки: ";

408 allowedValues\_i = new int[2]{1, 2};

409 bool descendingSort = getInput(allowedValues\_i, 2) - 1;

410 delete[] allowedValues\_i;

411 allowedValues\_i = nullptr;

412

413 sortStudentList(students, studentsNumber, sortMode, descendingSort);

414 displayStudentList(students, studentsNumber);

415 }

416

417 void printStudentList(Student \*students, int studentsNumber)

418 {

419 cout << "--------------------------------------------------------------------------------------------------------------\n"

420 << format("| {:5} | {:20} | {:20} | {:20} | {:12} | {:4} | {:7} |\n", "#", "Surname", "Name", "Patronymic", "Birth Year", "Curs", "Marks")

421 << "--------------------------------------------------------------------------------------------------------------\n";

422 for (int i = 0; i < studentsNumber; i++)

423 {

424 cout << format("| {:5} | {:20} | {:20} | {:20} | {:12} | {:4} | {:1}, {:1}, {:1} |\n", i + 1, students[i].surname, students[i].name, students[i].patronymic, students[i].birthYear, students[i].course, students[i].marks[0], students[i].marks[1], students[i].marks[2])

425 << "--------------------------------------------------------------------------------------------------------------\n";

426 }

427 }

428

429 void displayStudentList(Student \*students, int studentsNumber)

430 {

431 if (!isStudents(studentsNumber))

432 return;

433 cout << "Укажите, куда произвести вывод:\n "

434 << "1 - Консоль\n "

435 << "2 - Текстовый файл\n ";

436 cout << "Точка вывода: ";

437

438 allowedValues\_i = new int[2]{1, 2};

439 int fileOutput = getInput(allowedValues\_i, 2) - 1;

440 delete[] allowedValues\_i;

441 allowedValues\_i = nullptr;

442

443 if (fileOutput)

444 {

445 fstream file("output.txt", ios::out);

446 file << "-------------------------------------------------------------------------------------------------------------------------\n"

447 << format("| {:5} | {:20} | {:20} | {:20} | {:12} | {:4} | {:7} |\n", "№", "Фамилия", "Имя", "Отчество", "Год рождения", "Курс", "Оценки")

448 << "-------------------------------------------------------------------------------------------------------------------------\n";

449 for (int i = 0; i < studentsNumber; i++)

450 {

451 file << format("| {:5} | {:20} | {:20} | {:20} | {:12} | {:4} | {:1}, {:1}, {:1} |\n", i + 1, students[i].surname, students[i].name, students[i].patronymic, students[i].birthYear, students[i].course, students[i].marks[0], students[i].marks[1], students[i].marks[2])

452 << "-------------------------------------------------------------------------------------------------------------------------\n";

453 }

454 cout << "\nВывод сохранен в файл\"output.txt\".\n\nДля возврата в меню нажмите enter . . . ";

455 }

456 else

457 {

458 printStudentList(students, studentsNumber);

459 cout << "\nДля возврата в меню нажмите enter... ";

460 }

461 getchar(); // ожидание нажатия клавиши

462 }

463

464 void specifyFilePath(char \*path, int binaryFile)

465 {

466 cout << "\nУкажите файл списка студентов:\n "

467 << "1 - Файл по умолчанию(" << (binaryFile ? "student\_list.bin" : "student\_list.txt") << ")\n "

468 << "2 - Указать собственный путь\n ";

469 cout << "Файл: ";

470 int allowedValues\_i[2] = {1, 2};

471 int specifyFile = getInput(allowedValues\_i, 2) - 1;

472

473 if (specifyFile)

474 {

475 cout << "\nУкажите путь к файлу списка студентов: ";

476 strcpy(path, getInput(1000));

477 }

478 else

479 {

480 strcpy(path, binaryFile ? "student\_list.bin" : "student\_list.txt");

481 }

482 }

483

484 int importFromFile(Student \*&students, int &studentsNumber, const char \*path, int binaryFile)

485 {

486 ifstream file(path, binaryFile ? ios::binary : ios::in);

487 if (!file.is\_open())

488 {

489

490 cout << "\nФайл списка студентов не найден. Импорт прерван.\n\nДля возврата в меню нажмите enter... ";

491 getchar();

492 return -1;

493 }

494

495 file.seekg(0, ios::end);

496 int tempStudentsNumber = file.tellg() / (binaryFile ? sizeof(Student) : 69);

497 file.seekg(0, ios::beg);

498 Student \*tempStudents = new Student[tempStudentsNumber];

499

500 if (binaryFile)

501 {

502 for (int i = 0; i < tempStudentsNumber; i++)

503 file.read((char \*)&tempStudents[i], sizeof(tempStudents[i]));

504 }

505 else

506 {

507 for (int i = 0; i < tempStudentsNumber; i++)

508 {

509 file.read(tempStudents[i].surname, 20);

510 tempStudents[i].surname[20] = '\0';

511

512 file.read(tempStudents[i].name, 20);

513 tempStudents[i].name[20] = '\0';

514

515 file.read(tempStudents[i].patronymic, 20);

516 tempStudents[i].patronymic[20] = '\0';

517

518 char birthYear[5] = "0";

519 file.read(birthYear, 4);

520 tempStudents[i].birthYear = atoi(birthYear);

521

522 char course[2] = "0";

523 file.read(course, 1);

524 tempStudents[i].course = atoi(course);

525

526 char mark[2] = "0";

527 file.read(mark, 1);

528 tempStudents[i].marks[0] = atoi(mark);

529

530 file.read(mark, 1);

531 tempStudents[i].marks[1] = atoi(mark);

532

533 file.read(mark, 1);

534 tempStudents[i].marks[2] = atoi(mark);

535

536 file.seekg(1, ios::cur);

537 }

538 }

539

540 if (file.fail() || file.bad())

541 {

542 file.close();

543 delete[] tempStudents;

544

545 cout << "\nФайл списка студентов поврежден. Импорт прерван.\n\nДля возврата в меню нажмите enter... ";

546 getchar();

547

548 return -1;

549 }

550 file.close();

551 delete[] students;

552

553 students = tempStudents;

554 studentsNumber = tempStudentsNumber;

555 return 0;

556 }

557

558 void importStudentList(Student \*&students, int &studentsNumber)

559 {

560 if (studentsNumber != 0)

561 {

562 cout << "\nТекущий спискок студентов будет перезаписан. Продолжить выполнение импорта?" << endl;

563 cout << "1 - Да" << endl;

564 cout << "2 - Нет" << endl;

565 cout << "Выбор: ";

566

567 allowedValues\_i = new int[2]{1, 2};

568 int opt = getInput(allowedValues\_i, 2) - 1;

569 delete[] allowedValues\_i;

570 allowedValues\_i = nullptr;

571

572 if (opt)

573 return;

574 }

575

576 int binaryFile{-1};

577

578 cout << "Укажите тип файла для импорта списка студентов (0 - отмена операции):\n"

579 << "1 - Текстовый (.txt)\n"

580 << "2 - Бинарный (.bin)\n";

581 cout << "Тип файла: ";

582 allowedValues\_i = new int[3]{0, 1, 2};

583 binaryFile = getInput(allowedValues\_i, 3);

584 delete[] allowedValues\_i;

585 allowedValues\_i = nullptr;

586

587 if (binaryFile == 0)

588 {

589 cout << "\nИмпорт прерван.\n\nДля возврата в меню нажмите enter... ";

590 getchar();

591 return;

592 }

593 binaryFile--;

594

595 char path[1001];

596

597 specifyFilePath(path, binaryFile);

598 importFromFile(students, studentsNumber, path, binaryFile);

599

600 cout << "\nИмпорт выполнен успешно!\n\nДля возврата в меню нажмите enter... ";

601 getchar();

602 }

603

604 void exportToFile(Student \*&students, int &studentsNumber, const char \*path, int binaryFile)

605 {

606 ofstream file(path, binaryFile ? ios::binary : ios::out);

607 if (!file.is\_open())

608 {

609

610 cout << "\nФайл списка студентов не найден. Экспорт прерван.\n\nДля возврата в меню нажмите enter... ";

611 getchar();

612 return;

613 }

614

615 if (binaryFile)

616 {

617 for (int i = 0; i < studentsNumber; i++)

618 file.write((char \*)&students[i], sizeof(students[i]));

619 }

620 else

621 {

622 for (int i = 0; i < studentsNumber; i++)

623 {

624 file << setw(20) << students[i].surname;

625 file << setw(20) << students[i].name;

626 file << setw(20) << students[i].patronymic;

627 file << setw(4) << students[i].birthYear;

628 file << setw(1) << students[i].course;

629 file << setw(1) << students[i].marks[0];

630 file << setw(1) << students[i].marks[1];

631 file << setw(1) << students[i].marks[2] << endl;

632 }

633 }

634 file.close();

635 }

636

637 void exportStudentList(Student \*students, int studentsNumber)

638 {

639 if (!isStudents(studentsNumber))

640 return;

641

642 int binaryFile{-1};

643

644 cout << "Укажите тип файла для экспорта списка студентов (0 - отмена операции):\n "

645 << "1 - Текстовый (.txt)\n "

646 << "2 - Бинарный (.bin)\n ";

647 cout << "Тип файла: ";

648

649 allowedValues\_i = new int[3]{0, 1, 2};

650 binaryFile = getInput(allowedValues\_i, 3);

651 delete[] allowedValues\_i;

652 allowedValues\_i = nullptr;

653

654 if (binaryFile == 0)

655 {

656 cout << "\nЭкспорт прерван.\n\nДля возврата в меню нажмите enter... ";

657 getchar();

658 return;

659 }

660 binaryFile--;

661

662 char path[1001];

663

664 specifyFilePath(path, binaryFile);

665 exportToFile(students, studentsNumber, path, binaryFile);

666

667 cout << "\nЭкспорт выполнен успешно!\n\nДля возврата в меню нажмите enter... ";

668 getchar();

669 }

670

671 void transformStudentList()

672 {

673 int tempStudentsNumber{0};

674 Student \*tempStudents = new Student[tempStudentsNumber];

675 char path[1001];

676

677 cout << "Укажите исходный файл списка студентов для трансформирования (0 - отмена операции):\n "

678 << "1 - Файл по умолчанию (student\_list.txt)\n "

679 << "2 - Указать собственный путь\n ";

680 cout << "Исходный файл для трансформирования: ";

681

682 allowedValues\_i = new int[3]{0, 1, 2};

683 int specifyFile = getInput(allowedValues\_i, 2);

684 delete[] allowedValues\_i;

685 allowedValues\_i = nullptr;

686

687 if (specifyFile == 0)

688 {

689 cout << "\nТрансформирование списка студентов прервано.\n\nДля возврата в меню нажмите enter... ";

690 getchar();

691 delete[] tempStudents;

692 return;

693 }

694 specifyFile--;

695

696 if (specifyFile)

697 {

698 cout << "\nУкажите путь к исходному файлу списка студентов: ";

699 strcpy(path, getInput(1000));

700 }

701 else

702 {

703 strcpy(path, "student\_list.txt");

704 }

705 if (!importFromFile(tempStudents, tempStudentsNumber, path, 0))

706 {

707 strcpy(path, "student\_list.bin");

708 cout << "\nУкажите конечный файл списка студентов для трансформирования :\n "

709 << "1 - Файл по умолчанию(student\_list.bin)\n "

710 << "2 - Указать собственный путь\n ";

711 cout << "Конечный файл для трансформирования: ";

712 allowedValues\_i = new int[2]{1, 2};

713 bool specifyFile = getInput(allowedValues\_i, 2) - 1;

714 delete[] allowedValues\_i;

715 allowedValues\_i = nullptr;

716 if (specifyFile)

717 {

718 cout << "\nУкажите путь к конечному файлу списка студентов:";

719 strcpy(path, getInput(1000));

720 }

721 else

722 {

723 strcpy(path, "student\_list.bin");

724 }

725 exportToFile(tempStudents, tempStudentsNumber, path, 1);

726 cout << "\nТрансформирование списка студентов выполнено успешно !\n\nДля возврата в меню нажмите enter... ";

727 getchar();

728 }

729 else

730 {

731 cout << "\nОшибка при чтении файла списка студентов. Трансформирование прервано.\n\nДля возврата в меню нажмите enter...";

732 getchar();

733 delete[] tempStudents;

734 }

735 }

736

737 void swapStudents(Student &student1, Student &student2)

738 {

739 Student tmp = student1;

740 student1 = student2;

741 student2 = tmp;

742 }

743

744 void sortByName(Student \*&students, int j, bool ascendingSort)

745 {

746 if ((ascendingSort && strcmp(students[j].name, students[j + 1].name) > 0) ||

747 (!ascendingSort && strcmp(students[j].name, students[j + 1].name) < 0))

748 {

749 swapStudents(students[j], students[j + 1]);

750 }

751 }

752

753 void sortBySurname(Student \*&students, int j, bool ascendingSort)

754 {

755 if ((ascendingSort && strcmp(students[j].surname, students[j + 1].surname) > 0) ||

756 (!ascendingSort && strcmp(students[j].surname, students[j + 1].surname) < 0))

757 {

758 swapStudents(students[j], students[j + 1]);

759 }

760 }

761

762 void sortByPatronymic(Student \*&students, int j, bool ascendingSort)

763 {

764 if ((ascendingSort && strcmp(students[j].patronymic, students[j + 1].patronymic) > 0) ||

765 (!ascendingSort && strcmp(students[j].patronymic, students[j + 1].patronymic) < 0))

766 {

767 swapStudents(students[j], students[j + 1]);

768 }

769 }

770

771 void sortByBirthYear(Student \*&students, int j, bool ascendingSort)

772 {

773 if ((ascendingSort && students[j].birthYear > students[j + 1].birthYear) ||

774 (!ascendingSort && students[j].birthYear < students[j + 1].birthYear))

775 {

776 swapStudents(students[j], students[j + 1]);

777 }

778 }

779

780 void sortByCourse(Student \*&students, int j, bool ascendingSort)

781 {

782 if ((ascendingSort && students[j].course > students[j + 1].course) ||

783 (!ascendingSort && students[j].course < students[j + 1].course))

784 {

785 swapStudents(students[j], students[j + 1]);

786 }

787 }

788

789 void sortByMark(Student \*&students, int j, int markNumber, bool ascendingSort)

790 {

791 if ((ascendingSort && students[j].marks[markNumber] > students[j + 1].marks[markNumber]) ||

792 (!ascendingSort && students[j].marks[markNumber] < students[j + 1].marks[markNumber]))

793 {

794 swapStudents(students[j], students[j + 1]);

795 }

796 }

797

798 void sortStudentList(Student \*&students, int studentsNumber, int sortMode, bool ascendingSort)

799 {

800 for (int i = 0; i < studentsNumber - 1; i++)

801 {

802 for (int j = 0; j < studentsNumber - 1; j++)

803 {

804 switch (sortMode)

805 {

806 case 0:

807 sortBySurname(students, j, ascendingSort);

808 break;

809 case 1:

810 sortByName(students, j, ascendingSort);

811 break;

812 case 2:

813 sortByPatronymic(students, j, ascendingSort);

814 break;

815 case 3:

816 sortByBirthYear(students, j, ascendingSort);

817 break;

818 case 4:

819 sortByCourse(students, j, ascendingSort);

820 break;

821 case 5:

822 sortByMark(students, j, 0, ascendingSort);

823 break;

824 case 6:

825 sortByMark(students, j, 1, ascendingSort);

826 break;

827 case 7:

828 sortByMark(students, j, 2, ascendingSort);

829 break;

830 }

831 }

832 }

833 }

834

835 int getInput(int \*templ, int templNum)

836 {

837 while (true)

838 {

839 int inp;

840 cin >> inp;

841 if (cin.fail())

842 {

843 cin.clear();

844 clearInput();

845 cout << "Некорректный ввод. Пожалуйста, введите правильное значение : ";

846 continue;

847 }

848 else

849 {

850 clearInput();

851 }

852

853 if (templ != nullptr)

854 {

855 if (templNum != 0)

856 {

857 for (int i = 0; i < templNum; i++)

858 {

859 if (inp == templ[i])

860 {

861 return inp;

862 }

863 }

864 cout << "Некорректный ввод. Пожалуйста, введите правильное значение : ";

865 }

866 else

867 {

868 cout << "Error: function getInput() received templNum = 0 with specified templ\n ";

869 return -1;

870 }

871 }

872 else

873 {

874 return inp;

875 }

876 }

877 }

878

879 char \*getInput(int strLen)

880 {

881 char \*str = new char[strLen];

882 cin.getline(str, strLen);

883 return str;

884 }

885

886 bool isStudents(int studentsNumber)

887 {

888 if (studentsNumber == 0)

889 {

890 cout << "\nСписок студентов пуст. Операция невозможена.\n"

891 << endl

892 << "Для возврата в меню нажмите enter... ";

893 getchar();

894 return false;

895 }

896 return true;

897 }

898

899 void printMenu()

900 {

901 cout << "Выберите необходимое действие:\n"

902 << "1 - Добавление студента\n"

903 << "2 - Изменение студента\n"

904 << "3 - Удаление студента\n"

905 << "4 - Вывод списка студентов\n"

906 << "5 - Вывод сортированного списка студентов\n"

907 << "6 - Импорт списка студентов\n"

908 << "7 - Экспорт списка студентов\n"

909 << "8 - Трансформирование текстового файла списка студентов в бинарный\n"

910 << "0 - Выход\n";

911 cout << "Действие: ";

912 }

913