|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет  имени Н. Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н. Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 6

ПО КУРСУ

«Алгоритмические языки»

на тему «Работа с потоками ввода и вывода в языке Си++»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-13 |  |  |  | В. С. Ажгирей |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | М. В. Малахов |
|  |  |  |  |  | (И.О. Фамилия) |

2023

Введение

Цели и задачи работы

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих ввод/вывод в файлы. Для достижения цели необходимо выполнить следующие задачи:

- изучить необходимые учебные материалы, посвященные потокам ввода и вывода языка Си++;

- разработать программы на языке Си++ для решения заданных вариантов заданий;

- отладить программы;

- выполнить решение контрольного примера небольшой размерности с помощью программы и ручной расчет контрольного примера;

- подготовить отчет по лабораторной работе.

Условия для 1 варианта

В лабораторной работе необходимо разработать приложение, работающее в различных режимах: чтения, записи, очистки файла. Также задается текстовый или двоичный режим открытия файла. В режиме записи исходные данные вводятся с клавиатуры (количество объектов, заданных структурным типом, данные этих объектов), введенные данные ***дозаписываются*** в файл, не затирая предыдущую информацию. В режиме чтения данные, сохраненные в режиме записи в файле, читаются из файла и выводятся на консоль. В режиме очистки файла производится удаление всей информации из файла (по окончании работы данного режима файл на ПК должен существовать и не иметь содержимого).

Режимы приложения задаются как параметры командной строки и сохраняются в глобальные переменные: вначале чтение или запись, затем текстовый или двоичный режим, далее имя файла, например:

write binary MyFile.dat режим записи, файл в двоичном режиме, имя файла MyFile.dat;

read text MyFile.txt режим чтения, файл в текстовом режиме, имя файла MyFile.txt.

Также в ПО необходимо обрабатывать ситуацию, при которой читаемый файл оказался битым, т.е. в нем хранятся некорректные данные, например, поле типа int хранит строковое значение.

ПО должно содержать несколько осмысленных функций, разработанных студентом.

***Необходимо использовать библиотеку fstream.***

1. Структура «студент» (поля: ФИО, группа, номер зачетной книжки, массив 4-х оценок за сессию)

Основная часть

Исходный текст файла main.cpp:

#include "sources.h"

std::string mode, representation, fileName;

int main(int argc, char \*argv[])

{

    if (argc < 4)

    {

        std::cout << "Error: Insufficient command line arguments" << std::endl;

        return 1;

    }

    mode = argv[1];

    representation = argv[2];

    fileName = argv[3];

    fileManager(mode, representation, fileName);

    return 0;

}

Исходный текст файла sources.h:

#pragma once

#include <fstream>

#include <iostream>

#include <string>

#include <vector>

struct Student

{

    std::string full\_name;

    unsigned short int study\_group;

    unsigned int credit\_card\_number;

    unsigned short int grades[4];

};

void fileManager(const std::string mode, const std::string representation, const std::string fileName);

Исходный текст файла sources.cpp:

#include "sources.h"

void printStudent(const Student &student)

{

    std::cout << "Full name: " << student.full\_name << std::endl;

    std::cout << "Study group: " << student.study\_group << std::endl;

    std::cout << "Credit card number: " << student.credit\_card\_number << std::endl;

    std::cout << "Assessment for exams: ";

    for (size\_t i = 0; i < 4; ++i)

    {

        std::cout << student.grades[i] << ' ';

    }

    std::cout << '\n'

              << "#############################################" << std::endl;

}

void fileManager(const std::string mode, const std::string representation, const std::string fileName)

{

    if (mode == "-r")

    {

        std::ifstream readFile;

        if (representation == "bin")

        {

            readFile.open(fileName, std::ios::binary);

        }

        else

        {

            readFile.open(fileName);

        }

        if (!readFile)

        {

            std::cerr << "Failed to open file" << std::endl;

            return;

        }

        Student student;

        if (representation == "bin")

        {

            char currentByte;

            while (readFile.get(currentByte))

            {

                student.full\_name = "";

                char endString = '\0';

                while (currentByte != endString)

                {

                    student.full\_name += currentByte;

                    readFile.get(currentByte);

                }

                readFile.read((char \*)&student.study\_group, sizeof(unsigned short int));

                readFile.read((char \*)&student.credit\_card\_number, sizeof(unsigned int));

                for (size\_t i = 0; i < 4; ++i)

                {

                    readFile.read((char \*)&student.grades[i], sizeof(unsigned short int));

                }

                printStudent(student);

            }

        }

        else

        {

            std::string study\_group, credit\_card\_number, mark1, mark2, mark3, mark4;

            while (readFile >> student.full\_name >> study\_group >> credit\_card\_number >> mark1 >> mark2 >> mark3 >> mark4)

            {

                student.study\_group = (unsigned short int)std::stoi(study\_group);

                student.credit\_card\_number = (unsigned int)std::stoi(credit\_card\_number);

                student.grades[0] = (unsigned short int)std::stoi(mark1);

                student.grades[1] = (unsigned short int)std::stoi(mark2);

                student.grades[2] = (unsigned short int)std::stoi(mark3);

                student.grades[3] = (unsigned short int)std::stoi(mark4);

                printStudent(student);

            }

        }

        readFile.close();

    }

    else if (mode == "-w")

    {

        std::ofstream writeFile;

        if (representation == "bin")

        {

            writeFile.open(fileName, std::ios::binary | std::ios::app);

        }

        else

        {

            writeFile.open(fileName, std::ios::app);

        }

        if (!writeFile)

        {

            std::cerr << "Failed to open file" << std::endl;

            return;

        }

        unsigned int n;

        std::cout << "Enter count of students: ";

        std::cin >> n;

        for (size\_t i = 0; i < n; ++i)

        {

            std::string full\_name;

            std::cout << "Enter the student's name: ";

            std::cin >> full\_name;

            unsigned short study\_group;

            std::cout << "Enter the study group: ";

            std::cin >> study\_group;

            unsigned int credit\_card\_number;

            std::cout << "Enter credit card number: ";

            std::cin >> credit\_card\_number;

            unsigned short int mark1, mark2, mark3, mark4;

            std::cout << "Enter 4 assessments for exams: ";

            std::cin >> mark1 >> mark2 >> mark3 >> mark4;

            unsigned short int grades[] = {mark1, mark2, mark3, mark4};

            if (representation == "bin")

            {

                writeFile.write(full\_name.c\_str(), full\_name.size() + 1);

                writeFile.write((char \*)&study\_group, sizeof(unsigned short int));

                writeFile.write((char \*)&credit\_card\_number, sizeof(unsigned int));

                for (size\_t i = 0; i < 4; ++i)

                {

                    writeFile.write((char \*)&grades[i], sizeof(unsigned short int));

                }

            }

            else

            {

                writeFile << full\_name << " " << study\_group << " " << credit\_card\_number << " " << mark1 << " " << mark2 << " " << mark3 << " " << mark4 << std::endl;

            }

            std::cout << "#############################################" << std::endl;

        }

        writeFile.close();

    }

    else

    {

        std::ofstream clearFile;

        if (representation == "bin")

        {

            clearFile.open(fileName, std::ios::binary | std::ios::trunc);

        }

        else

        {

            clearFile.open(fileName, std::ios::trunc);

        }

        clearFile.close();

    }

}

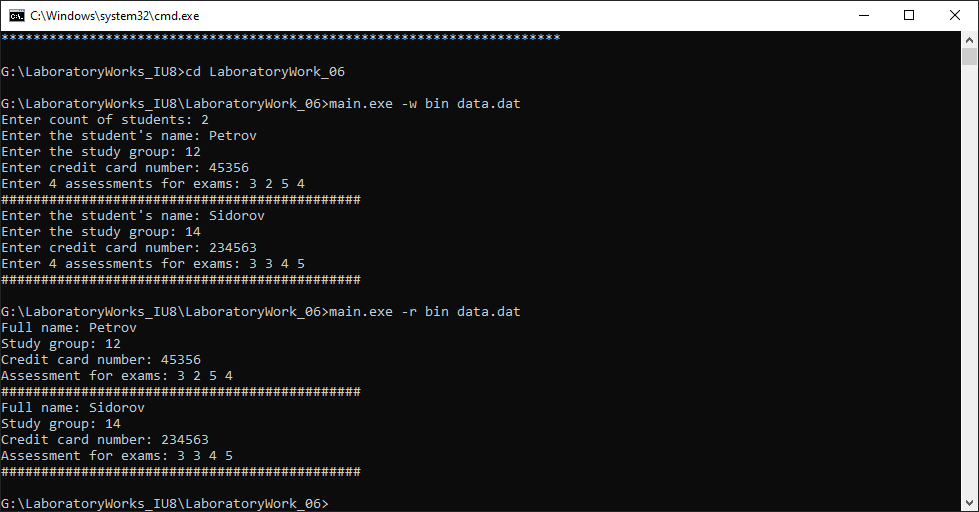
Снимки выполнения работы программы

Рисунок 1 – Запуск программы в режиме бинарной записи в файл data.dat . Запуск программы в режиме чтения бинарных данных из файла data.dat .

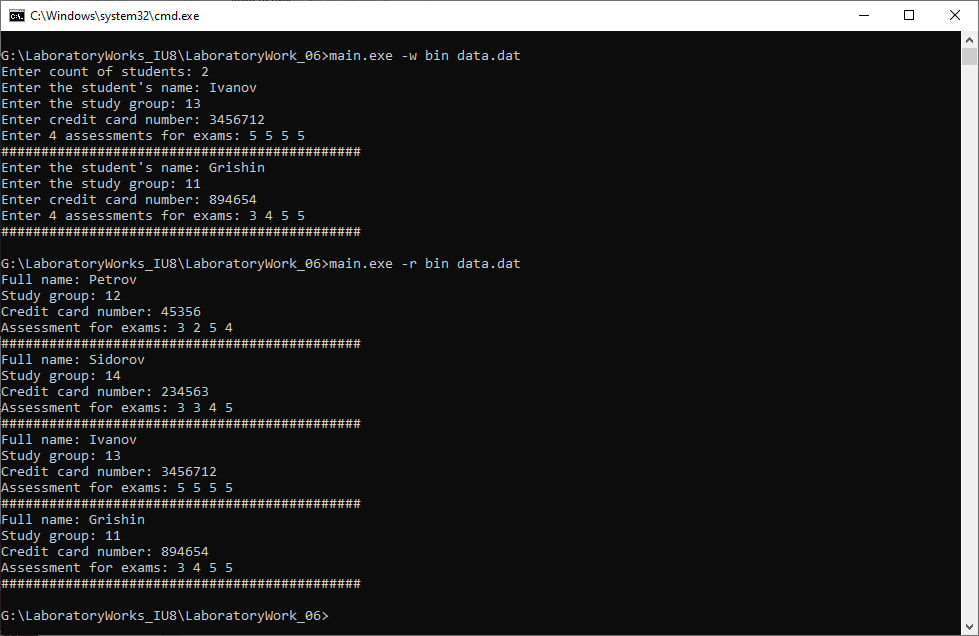


Рисунок 2 – Запуск программы в режиме бинарной дозаписи в файл data.dat . Запуск программы в режиме чтения бинарных данных из файла data.dat .

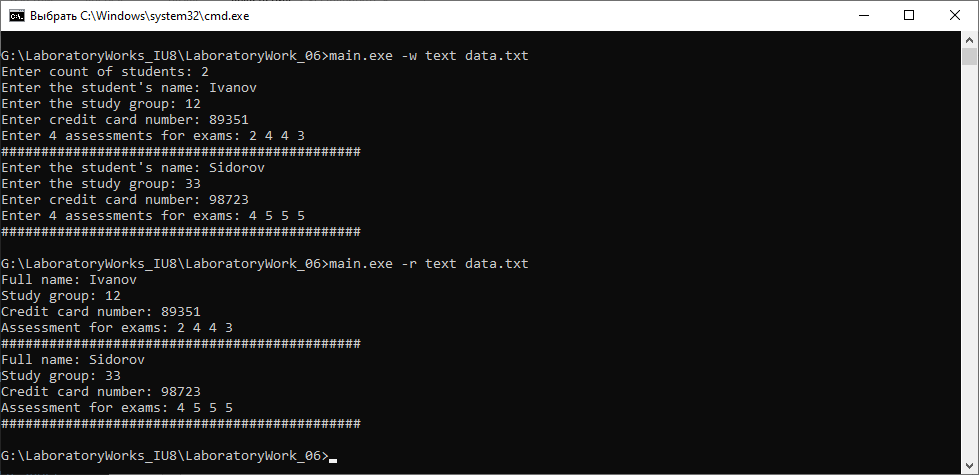


Рисунок 3 – Запуск программы в режиме текстовой записи в файл data.txt . Запуск программы в режиме чтения текстовых данных из файла data.txt .

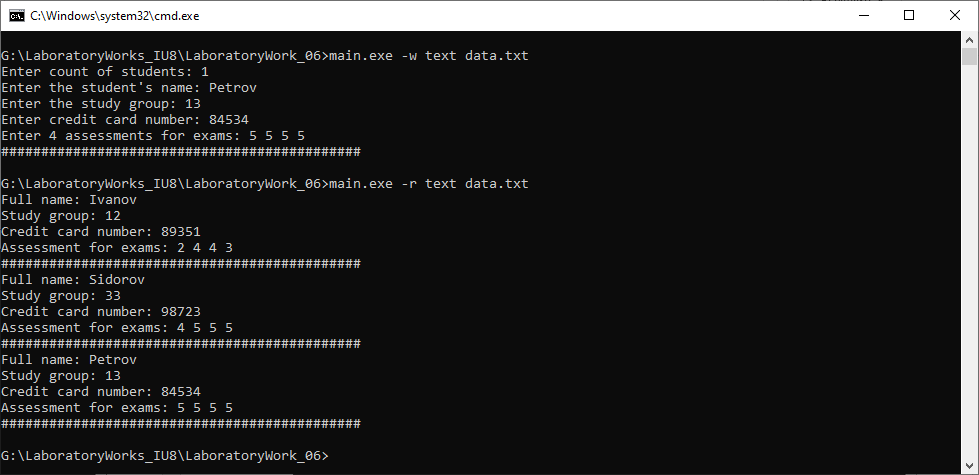


Рисунок 4 – Запуск программы в режиме текстовой дозаписи в файл data.txt . Запуск программы в режиме чтения текстовых данных из файла data.txt .

Заключение

Задачи лабораторной работы были решены, результаты проверены. Изучена на практике работа с файлами и библиотека fstream в языке С/С++, а также глобальные переменные.