ИСТ-АДМО-202Б

Трубин В.Д.

Орлов К.О.

3241.506115.000 ПЗ

Геоинформационных систем

Языки программирования

Уфа – 2025

Объекты и классы в C++   
Вариант 2

Содержание

[Введение 3](#_Toc198795382)

[1 Создание программы 4](#_Toc198795383)

[Заключение 9](#_Toc198795384)

[Список литературы 10](#_Toc198795385)

Введение

Цель работы – изучение и практическое освоение основ объектно-ориентированного программирования в c++

# Создание программы

Программа на C++ предназначена для обработки и сравнения экзаменационных результатов студентов за текущий и предыдущий учебный год. Реализованы основные функции:

* Чтение данных из файлов
* Преобразование текстовых оценок в числовые
* Вычисление средних баллов
* Сравнение результатов между годами

Класс Student содержит:

* fullName - полное имя студента
* marks - массив оценок
* subjectCount - количество предметов

Класс ExamResults - основной функциональный класс программы.

Поля класса:

* subjects[] - названия предметов
* students[] - список студентов
* Счетчики studentCount и subjectCount

Основные методы:

* markToNumeric() - преобразование оценок:
* readFromFile() - чтение данных:
* printResults() - вывод:
* calculateAverage() - вычисление среднего балла
* writeToFile() - сохранение результатов
* compareResults() - сравнение результатов по годам

Главная функция (main)

* Вывод результатов
* Сохранение в файлы (output\_current\_year.txt, output\_last\_year.txt)

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <iomanip>

#include <map>

using namespace std;

const int MAX\_STUDENTS = 100; // Maximum number of students

const int MAX\_SUBJECTS = 10; // Maximum number of subjects

// Structure to store student information

struct Student {

string fullName; // Full name

int marks[MAX\_SUBJECTS]; // Marks for subjects

int subjectCount; // Number of subjects

};

// Class to handle exam results

class ExamResults {

private:

string subjects[MAX\_SUBJECTS]; // Subjects names

Student students[MAX\_STUDENTS]; // List of students

int studentCount; // Number of students

int subjectCount; // Number of subjects

// Convert mark to numeric score (now using English assessment system)

int markToNumeric(const string& mark) {

if (mark == "unsatisfactory") return 2;

if (mark == "satisfactory") return 3;

if (mark == "good") return 4;

if (mark == "excellent") return 5;

return 0;

}

public:

// Constructor

ExamResults() : studentCount(0), subjectCount(0) {}

// Read data from file

void readFromFile(const string& filename) {

ifstream file(filename);

if (!file) {

cerr << "Unable to open file: " << filename << endl;

return;

}

string line;

// Read subject names

if (getline(file, line)) {

istringstream stream(line);

string subject;

while (stream >> subject && subjectCount < MAX\_SUBJECTS) {

subjects[subjectCount++] = subject;

}

}

// Read student data

while (getline(file, line) && studentCount < MAX\_STUDENTS) {

istringstream stream(line);

Student student;

student.subjectCount = 0;

stream >> student.fullName;

string mark;

while (stream >> mark && student.subjectCount < subjectCount) {

student.marks[student.subjectCount++] = markToNumeric(mark);

}

students[studentCount++] = student;

}

file.close();

}

// Print results to the screen

void printResults() {

cout << "Subjects: ";

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

cout << subjects[i] << " ";

}

cout << endl;

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

cout << students[i].fullName << ": ";

for (int j = 0; j < students[i].subjectCount; ++j) {

cout << students[i].marks[j] << " ";

}

cout << "| Average score: " << calculateAverage(students[i]) << endl;

}

}

// Calculate student's average score

double calculateAverage(const Student& student) {

if (student.subjectCount == 0) return 0.0;

int sum = 0;

for (int i = 0; i < student.subjectCount; ++i) {

sum += student.marks[i];

}

return static\_cast<double>(sum) / student.subjectCount;

}

// Write results to a file

void writeToFile(const string& filename) {

ofstream file(filename);

if (!file) {

cerr << "Unable to open file for writing: " << filename << endl;

return;

}

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

file << subjects[i] << " ";

}

file << endl;

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

file << students[i].fullName << " ";

for (int j = 0; j < students[i].subjectCount; ++j) {

file << students[i].marks[j] << " ";

}

file << endl;

}

file.close();

}

// Compare results of students for current and previous year

void compareResults(const ExamResults& other) {

cout << "Comparison of results for the current and previous year:\n";

map<string, int> currentResults, previousResults;

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

currentResults[students[i].fullName] = calculateAverage(students[i]);

}

for (int i = 0; i < other.studentCount; ++i) {

previousResults[other.students[i].fullName] = calculateAverage(other.students[i]);

}

for (const auto& student : currentResults) {

cout << "Student: " << student.first << "\n";

if (previousResults.find(student.first) != previousResults.end()) {

cout << " Current year: " << student.second << "\n";

cout << " Previous year: " << previousResults[student.first] << "\n";

}

else {

cout << " No data for the previous year.\n";

}

}

}

};

int main() {

ExamResults currentYearResults;

ExamResults lastYearResults;

// Read data for current and previous year

currentYearResults.readFromFile("current\_year.txt");

lastYearResults.readFromFile("last\_year.txt");

// Print results to screen

cout << "Current year results:\n";

currentYearResults.printResults();

cout << "\nPrevious year results:\n";

lastYearResults.printResults();

// Compare results

currentYearResults.compareResults(lastYearResults);

// Write results to files

currentYearResults.writeToFile("output\_current\_year.txt");

lastYearResults.writeToFile("output\_last\_year.txt");

return 0;

}

Заключение

Входе данной лабораторной работы была успешно реализована программа, анализирующая успеваемость студентов и их средние баллы, данная программа была написана методами ООП и демонстрирует основные принципы данного метода программирования

Список литературы

1. Кубенский, А. А. Функциональное программирование : учебник и практикум для академического бакалавриата / А. А. Кубенский. — М. : Издательство Юрайт, 2019. — 348 с.
2. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2019. — 137 с