ИСТ-АДМО-202Б

Трубин В.Д.

Орлов К.О.

3241.506215.000 ПЗ

Геоинформационных систем

Языки программирования

Уфа – 2025

Динамические массивы 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 динамических массивов в c++

# Корректирование программы

За основу была взята программа из лабораторной работы номер 1

Основной целью стояло усовершенствовать ее таким образом, что бы программа работала не с фиксированным количеством дисциплин в файле, а с вариативным

За этой целью в программу были добавлены динамические массивы(int\* marks, string\* subjects), однако при использовании динамических массивов можно столкнуться с утечкой памяти, поэтому в программе так же был реализована ручная чистка памяти

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <map>

using namespace std;

// Структура студента

struct Student {

string fullName;

int\* marks;

int subjectCount;

Student(int subjectCount) : subjectCount(subjectCount) {

marks = new int[subjectCount];

}

~Student() {

delete[] marks;

}

};

// Класс обработки результатов экзаменов

class ExamResults {

private:

string\* subjects;

Student\*\* students;

int subjectCount;

int studentCount;

// Преобразование текстовой оценки в числовую

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:

ExamResults() = default;

~ExamResults() {

delete[] subjects;

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

delete students[i];

delete[] students;

}

void readFromFile(const string& filename) {

ifstream file(filename);

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

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

return;

}

string line;

// 1. Считаем количество предметов

if (getline(file, line)) {

istringstream stream(line);

string tmp;

int count = 0;

while (stream >> tmp)

++count;

subjectCount = count;

subjects = new string[subjectCount];

stream.clear();

stream.str(line);

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

stream >> subjects[i];

}

// 2. Подготовим массив студентов (максимум 100)

int capacity = 100;

students = new Student \* [capacity];

studentCount = 0;

// 3. Читаем студентов

while (getline(file, line)) {

istringstream stream(line);

string name;

stream >> name;

if (name.empty()) continue;

Student\* student = new Student(subjectCount);

student->fullName = name;

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

string mark;

if (stream >> mark)

student->marks[i] = markToNumeric(mark);

else

student->marks[i] = 0; // На случай отсутствия оценки

}

if (studentCount < capacity)

students[studentCount++] = student;

else

delete student; // если превышен лимит, не добавляем

}

file.close();

}

void printResults() const {

if (subjectCount == 0 || studentCount == 0) {

cout << "No data available.\n";

return;

}

cout << "Subjects: ";

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

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

cout << "\n";

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

cout << students[i]->fullName << ": ";

for (int j = 0; j < subjectCount; ++j)

cout << students[i]->marks[j] << " ";

cout << "| Average score: " << calculateAverage(\*students[i]) << "\n";

}

}

double calculateAverage(const Student& student) const {

int sum = 0;

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

sum += student.marks[i];

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

}

void writeToFile(const string& filename) const {

ofstream file(filename);

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

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 < subjectCount; ++j)

file << students[i]->marks[j] << " ";

file << endl;

}

file.close();

}

void compareResults(const ExamResults& other) const {

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

map<string, double> 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 (map<string, double>::const\_iterator it = currentResults.begin(); it != currentResults.end(); ++it) {

string name = it->first;

double currentAvg = it->second;

cout << "Student: " << name << "\n";

map<string, double>::const\_iterator prevIt = previousResults.find(name);

if (prevIt != previousResults.end()) {

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

cout << " Previous year: " << prevIt->second << "\n";

}

else {

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

}

}

}

};

// Точка входа

int main() {

ExamResults currentYearResults;

ExamResults lastYearResults;

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

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

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

currentYearResults.printResults();

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

lastYearResults.printResults();

currentYearResults.compareResults(lastYearResults);

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

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

return 0;

}

Заключение

В ходе выполнения данной лабораторной работы была изучена и практически освоена работа с динамическими массивами в языке программирования C++. На основе программы из предыдущей лабораторной работы была реализована модификация, позволяющая обрабатывать вариативное количество дисциплин для каждого студента, что значительно повысило гибкость и универсальность программы.

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