Министерство образования и науки Российской Федерации

ФГБОУ ВО «Пензенский государственный университет»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Пенза 2025

**Отчет**

по лабораторной работе №1

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Простые структуры данных»

Выполнил: студент группы 24ВВИ1

Марушкин Д.И.

Егорова К.А.

Винникова Е.С

Проверил: к.т.н., доцент

Юрова О.В.

Деев М.В.

**Лабораторные задания:**

Задание 1: написать программу, вычисляющую разницу между максимальным и минимальным элементами массива.

Задание 2: написать программу, реализующую инициализацию массива случайными числами.

Задание 3: написать программу, реализующую создание массива произвольного размера, вводимого с клавиатуры.

Задание 4: написать программу, вычисляющую сумму значений в каждом столбце (или строке) двумерного массива.

Задание 5: написать программу, осуществляющую поиск среди структур student структуру с заданными параметрами (фамилией, именем и т.д.).

**Пояснительный текст к программе:**

**1. Назначение программы**

Программа представляет собой проект, включающий несколько функциональных модулей:

1. Работа с одномерными массивами:
   * Инициализация массива случайными числами.
   * Вычисление разницы между максимальным и минимальным элементами.
2. Работа с двумерными массивами (матрицами):
   * Инициализация матрицы случайными числами.
   * Вычисление суммы значений по столбцам.
3. Работа с базой студентов:
   * Добавление и удаление студентов.
   * Поиск студентов по имени.
   * Вывод полного списка студентов.и.

**2. Общая структура программы**

1. **main.cpp** — основной модуль, реализует меню и логику программы.
2. **array\_utils.h / array\_utils.cpp** — функции для работы с массивами: инициализация, вывод и вычисление разницы элементов.
3. **matrix\_utils.h / matrix\_utils.cpp** — функции для работы с матрицами: инициализация, вывод и суммирование столбцов.
4. **student\_utils.h / student\_utils.cpp** — база студентов на основе std::list, функции добавления, удаления, поиска и вывода.

**3. Использование std::list для студентов**

Для хранения студентов используется контейнер **std::list**. Основные причины:

* **Быстрое добавление и удаление элементов** в любом месте списка без сдвига остальных элементов.
* Итераторы к списку остаются валидными после вставки или удаления других элементов (в отличие от std::vector).

Каждый студент хранится в структуре Student с полями: id, name, lastName, nomZach. Класс StudentDatabase управляет списком студентов и поддерживает автоинкремент ID.

**4. Использование std::random для массивов и матриц**

Для генерации случайных чисел используется стандартная библиотека <random>:

* std::random\_device — источник энтропии.
* std::mt19937 — генератор случайных чисел Мерсенна-Твистера.
* std::uniform\_int\_distribution<int> — равномерное распределение в заданном диапазоне.

Преимущества такого подхода:

* Надёжные и качественные случайные числа.
* Лёгкая настройка диапазона значений.
* Современный и безопасный способ генерации чисел в C++17 и выше.

**5. Работа программы**

1. Пользователь выбирает пункт меню:
   * Работа с массивом, матрицей или база студентов.
2. В зависимости от выбора выполняется соответствующая функция:
   * Для массивов: создаются случайные элементы, вычисляется разница между максимумом и минимумом.
   * Для матриц: создаются случайные значения, вычисляется сумма по столбцам.
   * Для студентов: добавление, удаление, поиск и вывод списка.
3. Все действия сопровождаются выводом промежуточных и итоговых результатов.

Программа построена с использованием **динамической памяти для массивов и матриц**, что позволяет работать с произвольными размерами данных.

**Трассировка программы:**

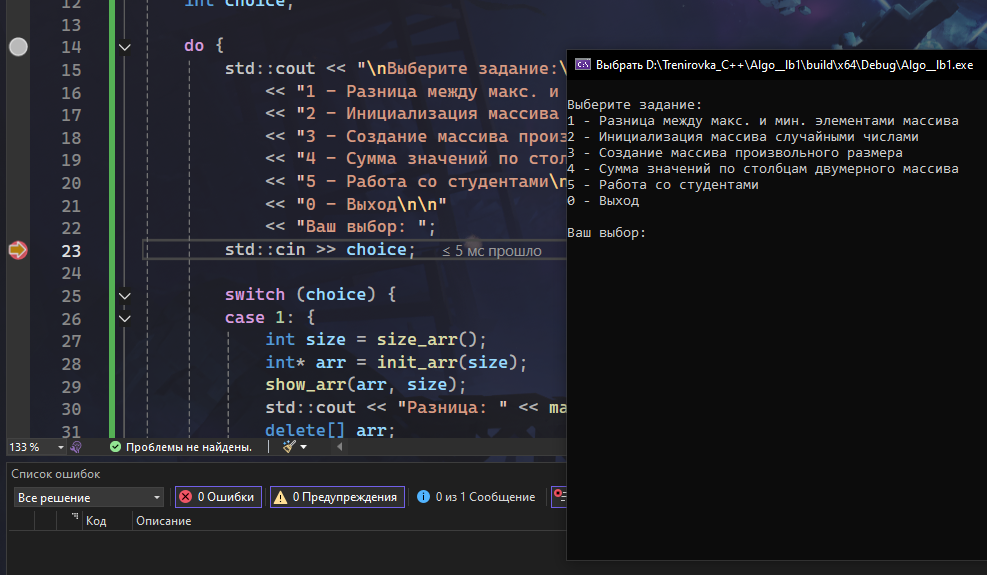


Рисунок 1 –Первичный запуск и демонстрация главного меню программы.

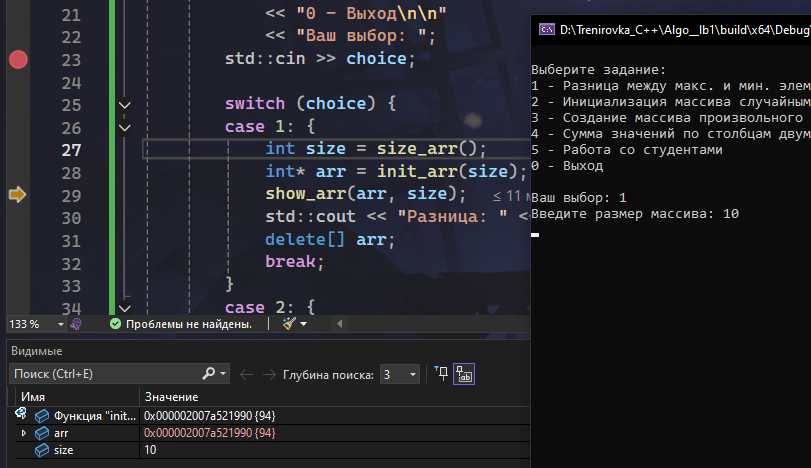


Рисунок 2 – Определение массива для первого задания.

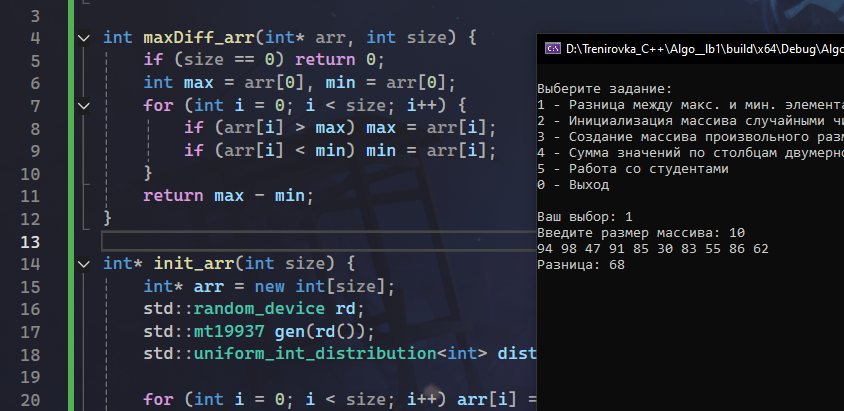


Рисунок 3 – Результат работы первого задания.

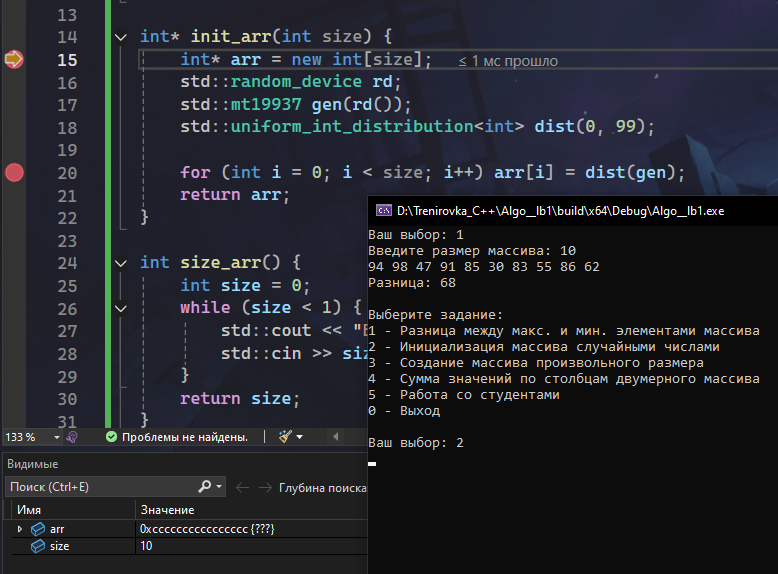


Рисунок 4 – Инициализация массива случайными числами для 2 задания.

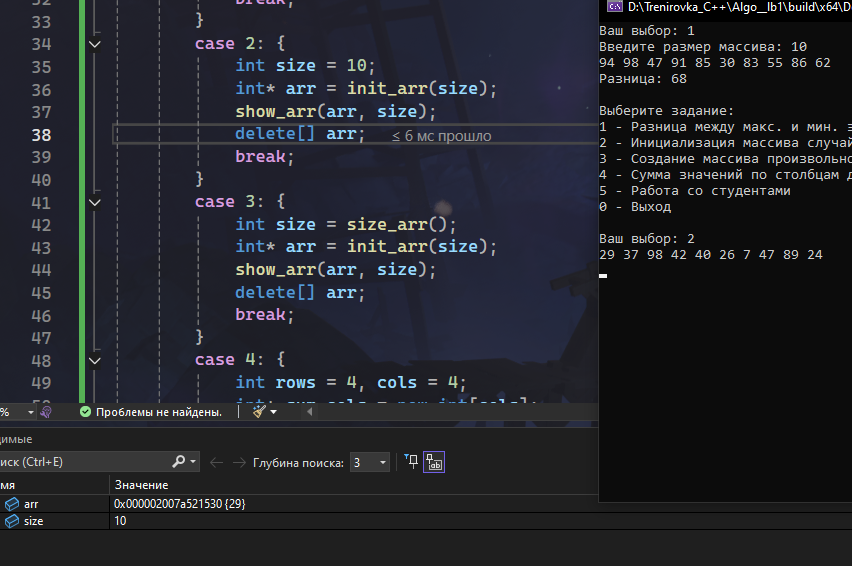


Рисунок 5 – Результат работы второго задания.

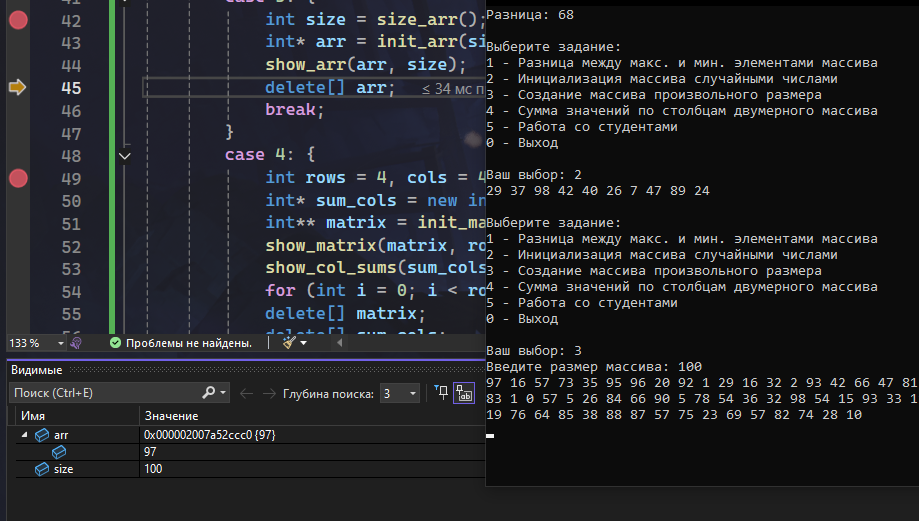


Рисунок 6 – результат работы третьего задания.

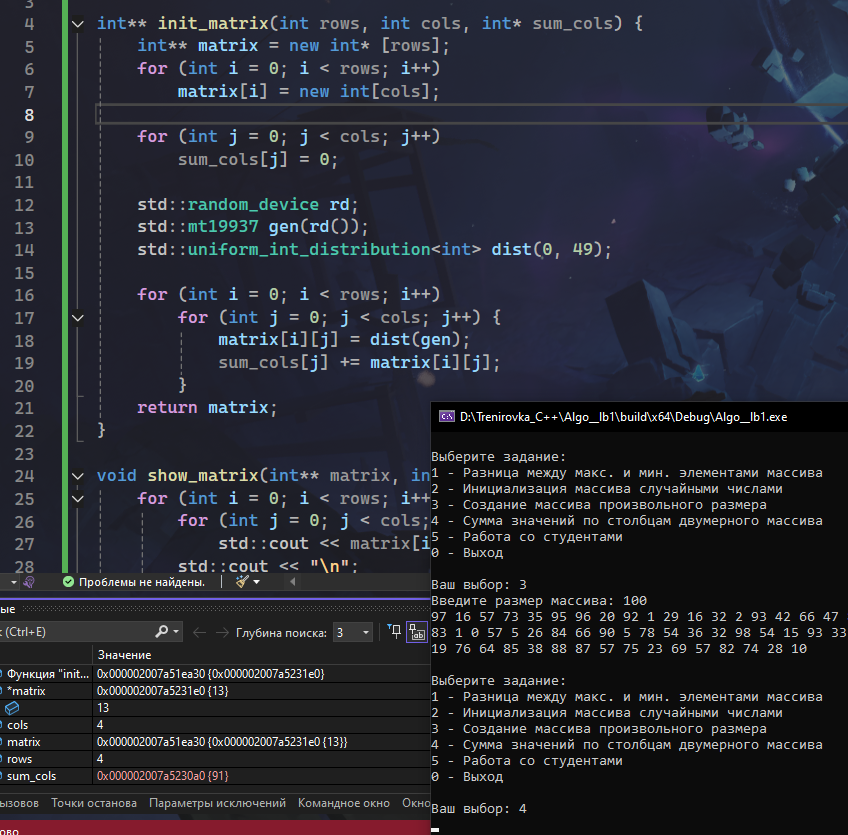


Рисунок 7 – Инициализация матрицы.

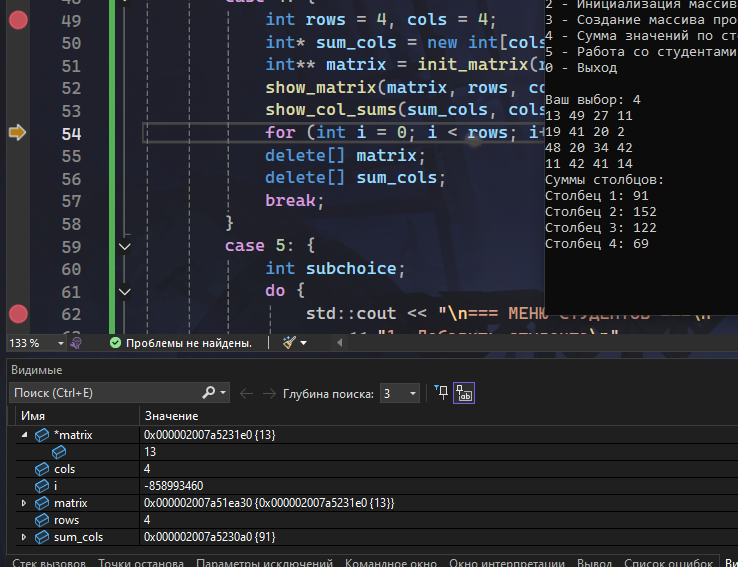


Рисунок 8 – Результат работы четвертого задания.

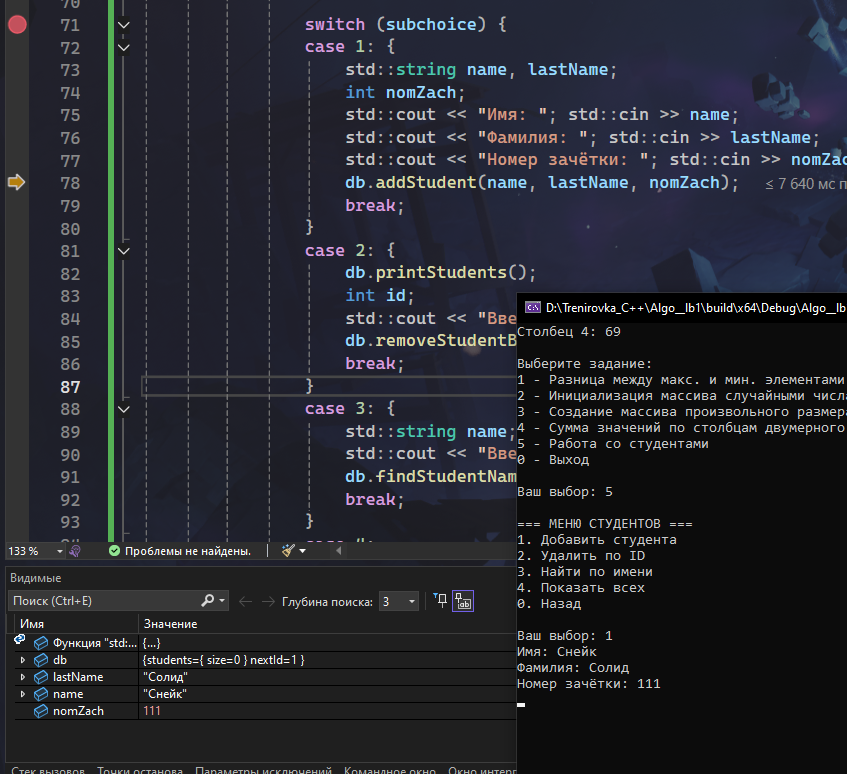


Рисунок 10 – Добавление студента.

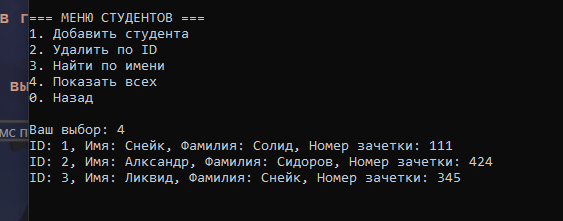


Рисунок 11 – Показать список студентов.

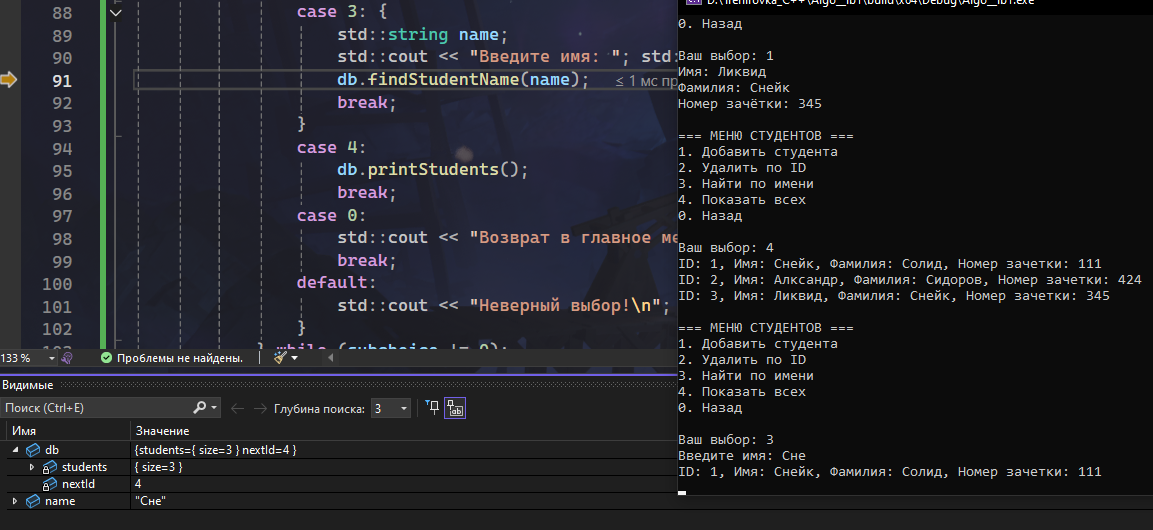


Рисунок 12 – Поиск студента по имени.

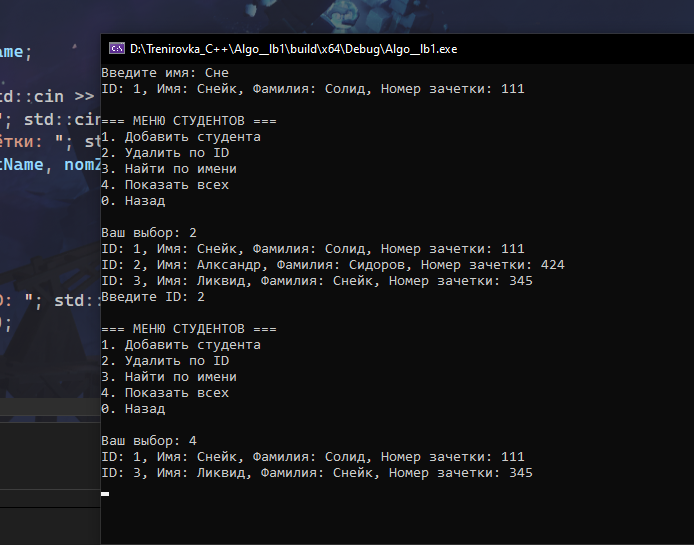


Рисунок 13 – Удаление студента.

**Вывод:**

При разработке данной программы мы научились использовать базовые структуры данных.

**Листинг программы:**

main.cpp

#include <iostream>

#include <Windows.h>

#include "array\_utils.h"

#include "matrix\_utils.h"

#include "student\_utils.h"

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

StudentDatabase db;

int choice;

do {

std::cout << "\nВыберите задание:\n"

<< "1 - Разница между макс. и мин. элементами массива\n"

<< "2 - Инициализация массива случайными числами\n"

<< "3 - Создание массива произвольного размера\n"

<< "4 - Сумма значений по столбцам двумерного массива\n"

<< "5 - Работа со студентами\n"

<< "0 - Выход\n\n"

<< "Ваш выбор: ";

std::cin >> choice;

switch (choice) {

case 1: {

int size = size\_arr();

int\* arr = init\_arr(size);

show\_arr(arr, size);

std::cout << "Разница: " << maxDiff\_arr(arr, size) << "\n";

delete[] arr;

break;

}

case 2: {

int size = 10;

int\* arr = init\_arr(size);

show\_arr(arr, size);

delete[] arr;

break;

}

case 3: {

int size = size\_arr();

int\* arr = init\_arr(size);

show\_arr(arr, size);

delete[] arr;

break;

}

case 4: {

int rows = 4, cols = 4;

int\* sum\_cols = new int[cols];

int\*\* matrix = init\_matrix(rows, cols, sum\_cols);

show\_matrix(matrix, rows, cols);

show\_col\_sums(sum\_cols, cols);

for (int i = 0; i < rows; i++) delete[] matrix[i];

delete[] matrix;

delete[] sum\_cols;

break;

}

case 5: {

int subchoice;

do {

std::cout << "\n=== МЕНЮ СТУДЕНТОВ ===\n"

<< "1. Добавить студента\n"

<< "2. Удалить по ID\n"

<< "3. Найти по имени\n"

<< "4. Показать всех\n"

<< "0. Назад\n\n"

<< "Ваш выбор: ";

std::cin >> subchoice;

switch (subchoice) {

case 1: {

std::string name, lastName;

int nomZach;

std::cout << "Имя: "; std::cin >> name;

std::cout << "Фамилия: "; std::cin >> lastName;

std::cout << "Номер зачётки: "; std::cin >> nomZach;

db.addStudent(name, lastName, nomZach);

break;

}

case 2: {

db.printStudents();

int id;

std::cout << "Введите ID: "; std::cin >> id;

db.removeStudentById(id);

break;

}

case 3: {

std::string name;

std::cout << "Введите имя: "; std::cin >> name;

db.findStudentName(name);

break;

}

case 4:

db.printStudents();

break;

case 0:

std::cout << "Возврат в главное меню...\n";

break;

default:

std::cout << "Неверный выбор!\n";

}

} while (subchoice != 0);

break;

}

case 0:

std::cout << "Выход...\n";

break;

default:

std::cout << "Неверный выбор!\n";

}

} while (choice != 0);

return 0;

}

student\_utils.h

#pragma once

#include <list>

#include <string>

#include <iostream>

struct Student {

int id;

std::string name;

std::string lastName;

int nomZach;

};

class StudentDatabase {

public:

StudentDatabase();

void addStudent(const std::string& name, const std::string& lastName, int nomZach);

void removeStudentById(int id);

void printStudents() const;

void findStudentName(const std::string& name) const;

private:

std::list<Student> students;

int nextId;

};

student\_utils.cpp

#include "student\_utils.h"

StudentDatabase::StudentDatabase() : nextId(1) {}

void StudentDatabase::addStudent(const std::string& name, const std::string& lastName, int nomZach) {

students.push\_back({ nextId++, name, lastName, nomZach });

}

void StudentDatabase::removeStudentById(int id) {

for (auto it = students.begin(); it != students.end(); )

if (it->id == id) it = students.erase(it);

else ++it;

}

void StudentDatabase::printStudents() const {

if (students.empty()) {

std::cout << "Список студентов пуст.\n";

return;

}

for (const auto& s : students)

std::cout << "ID: " << s.id

<< ", Имя: " << s.name

<< ", Фамилия: " << s.lastName

<< ", Номер зачетки: " << s.nomZach << "\n";

}

void StudentDatabase::findStudentName(const std::string& name) const {

bool found = false;

for (const auto& s : students)

if (s.name.find(name) == 0) {

std::cout << "ID: " << s.id

<< ", Имя: " << s.name

<< ", Фамилия: " << s.lastName

<< ", Номер зачетки: " << s.nomZach << "\n";

found = true;

}

if (!found)

std::cout << "Студенты с именем, начинающимся на \"" << name << "\" не найдены.\n";

}

array\_utils.h

#pragma once

#include <iostream>

int maxDiff\_arr(int\* arr, int size);

int\* init\_arr(int size);

int size\_arr();

void show\_arr(int\* arr, int size);

array\_utils.cpp

#include "array\_utils.h"

#include <random>

int maxDiff\_arr(int\* arr, int size) {

if (size == 0) return 0;

int max = arr[0], min = arr[0];

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

if (arr[i] > max) max = arr[i];

if (arr[i] < min) min = arr[i];

}

return max - min;

}

int\* init\_arr(int size) {

int\* arr = new int[size];

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dist(0, 99);

for (int i = 0; i < size; i++) arr[i] = dist(gen);

return arr;

}

int size\_arr() {

int size = 0;

while (size < 1) {

std::cout << "Введите размер массива: ";

std::cin >> size;

}

return size;

}

void show\_arr(int\* arr, int size) {

for (int i = 0; i < size; i++) std::cout << arr[i] << " ";

std::cout << "\n";

}

matrix\_utils.h

#pragma once

#include <iostream>

int\*\* init\_matrix(int rows, int cols, int\* sum\_cols);

void show\_matrix(int\*\* matrix, int rows, int cols);

void show\_col\_sums(int\* sum\_cols, int cols);

matrix\_utils.cpp

#include "matrix\_utils.h"

#include <random>

int\*\* init\_matrix(int rows, int cols, int\* sum\_cols) {

int\*\* matrix = new int\* [rows];

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

matrix[i] = new int[cols];

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

sum\_cols[j] = 0;

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dist(0, 49);

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

for (int j = 0; j < cols; j++) {

matrix[i][j] = dist(gen);

sum\_cols[j] += matrix[i][j];

}

return matrix;

}

void show\_matrix(int\*\* matrix, int rows, int cols) {

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

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

std::cout << matrix[i][j] << " ";

std::cout << "\n";

}

}

void show\_col\_sums(int\* sum\_cols, int cols) {

std::cout << "Суммы столбцов:\n";

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

std::cout << "Столбец " << j + 1 << ": " << sum\_cols[j] << "\n";

}