Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №1

по курсу «Логика и основы алгоритмизации в инженерных

задачах»

на тему «Простые структуры данных»

Выполнил: ст. гр. 24ВВВ1

Будников А.С.

Принял: к.т.н, доцент

Юрова О. В.

2025

**Цель работы:**

Изучение и повторение простых структур данных.

**Лабораторное задание**

**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.).

**Описание метода решения задач**

**Задание 1**: Для вычисления разницы между максимальным и минимальным элементами массив предварительно отсортирован с помощью метода *sort()* из заголовочного файла *algorithm,* а затем вычислена разница между последним и первым элементом отсортированного массива.

**Задание 2**: Использован метод *rand()* из заголовочного файла *cstdlib.*

**Задание 3**: Для ввода размера массива использован стандартный поток *istream* с соответствующей проверкой на границы.

**Задание 4**: Использован цикл *while* с проверкой на номер стобцадля прохода по строке массива.

**Задание 5**: Использован массив структур *Student* с некоторым количеством полей. Поиск в массиве структур осуществляется с помощью шаблонной функции, возвращающей *vector* структур *Student*, удовлетворяющих условию, заданному предикатом.

**Результаты работы программы:**

**Вывод:**

В ходе выполнения работы были повторены простые структуры данных и динамическое использования памяти.

**Листинг:**

**Файл *flheaders.hpp***

#ifndef FLHEADERS\_H

#define FLHEADERS\_H

#include <iostream>

#include <cstdlib>

#include <stdexcept>

#include <ctime>

#include <locale>

#include <algorithm>

#include <string>

#include <iterator>

#include <limits>

#include <vector>

struct Student {

std::string surname;

std::string firstname;

std::string faculty;

int id;

};

inline Student studs[5] = {

{"Будников", "Алексей", "ФВТ", 12},

{"Иванов", "Иван", "ФФМЕН", 344},

{"Васильев", "Василий", "ЛФ", 993},

{"Гурин", "Алексей", "ФВТ", 32},

{"Захаров", "Артём", "ФВТ", 48},

};;

int\* generate\_random\_array(int size, int min, int max);

void print\_array(const int\* array, int size);

int calculate\_difference(int\* array, int size);

int\*\* create\_2DArray(int rows, int cols, int min, int max);

void print\_2DArray(int\*\* matrix, int rows, int cols);

void free\_2DArray(int\*\* matrix, int rows);

int calculate\_sum\_in\_row(int\*\* matrix, int row\_index, int cols, int rows);

template<class Pred>

std::vector<Student> find\_student(struct Student st[], int n, Pred pred) {

std::vector <Student> sts;

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

if (pred(st[i])) {

sts.push\_back(st[i]);

}

return sts;

}

bool contains\_substring(const Student& student, const std::string& substring);

bool contains\_any(const Student& student, const std::string& substring);

void initialize();

bool validate\_input(std::istream& input, int& value, const char\* prompt);

bool validate\_input(std::istream& input, int& value1, int& value2, const char\* prompt);

bool validate\_input(std::istream& input, std::string& value, const char\* prompt);

void print\_student(const std::vector<Student> &sts);

#endif // FLHEADERS\_H

**Файл main.cpp**

#include "flheaders.hpp"

using namespace std;

int main() {

try {

initialize();

const int min = -40;

const int max = 40;

int size;

if (!validate\_input(cin, size, "Введите размер массива: "))

throw runtime\_error("Ошибка ввода");

int \*array = generate\_random\_array(size, min, max);

cout << "Сгенерированный массив: ";

print\_array(array, size);

cout << endl;

cout << "Разница: " << calculate\_difference(array, size) << endl;

free(array);

int cols, rows;

if (!validate\_input(cin, rows, cols, "Введите количество строк и столбцов (rows cols): "))

throw runtime\_error("Ошибка ввода");

int \*\*matrix = create\_2DArray(rows, cols, min, max);

cout << "Сгенерированная матрица:" << endl;

print\_2DArray(matrix, rows, cols);

int row\_index;

if (!validate\_input(cin, row\_index, "Введите номер строки: "))

throw runtime\_error("Ошибка ввода");

cout << "Сумма чисел в строке под номером " << row\_index << ": " << calculate\_sum\_in\_row(matrix, row\_index, cols, rows) << endl;

free\_2DArray(matrix, rows);

string search\_term;

if (!validate\_input(cin, search\_term, "Введите строку для поиска: "))

throw runtime\_error("Ошибка ввода");

vector<Student> res = find\_student(studs, std::size(studs), [search\_term](Student &s) {

return contains\_any(s, search\_term);

});

print\_student(res);

} catch (const exception& e) {

cerr << "Ошибка: " << e.what() << endl;

return EXIT\_FAILURE;

}

return EXIT\_SUCCESS;

}

**Файл array\_operations.cpp**

#include "flheaders.hpp"

using namespace std;

int\* generate\_random\_array(int size, int min, int max) {

int \*array = static\_cast<int\*>(malloc(size \* sizeof(int)));

if (!array) throw runtime\_error("Ошибка выделения памяти");

for (int i = 0; i < size; i++) array[i] = min + (rand() % (max - min + 1));

return array;

}

void print\_array(const int\* array, int size) {

for (int i = 0; i < size; i++) cout << array[i] << " ";

}

int calculate\_difference(int\* array, int size) {

if (size <= 1) return 0;

sort(array, array + size);

int diff = array[size - 1] - array[0];

return diff;

}

int\*\* create\_2DArray(int rows, int cols, int min, int max) {

int \*\*array = static\_cast<int\*\*>(malloc(rows \* sizeof(int\*)));

if (!array) throw runtime\_error("Ошибка выделения памяти");

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

array[i] = static\_cast<int\*>(malloc(cols \* sizeof(int)));

if (!array[i]) {

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

free(array[j]);

}

free(array);

throw runtime\_error("Ошибка выделения памяти");

}

}

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

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

array[i][j] = min + (std::rand() % (max - min + 1));

}

}

return array;

}

void print\_2DArray(int\*\* matrix, int rows, int cols) {

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

for (int j = 0; j < cols; j++) cout << matrix[i][j] << " ";

cout << endl;

}

}

int calculate\_sum\_in\_row(int\*\* matrix, int row\_index, int cols, int rows) {

int sum = 0;

if (matrix == nullptr

|| row\_index - 1 <= 0

|| row\_index - 1 > rows

|| cols <= 0) return 0;

int i = 0;

while (i < cols) {

sum += matrix[row\_index - 1][i];

i++;

}

return sum;

}

void free\_2DArray(int\*\* matrix, int rows) {

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

free(matrix);

}

**Файл utilities.cpp**

#include "flheaders.hpp"

using namespace std;

#ifdef max

#undef max

#endif

#ifdef \_WIN32

#include <windows.h>

#endif

void initialize() {

srand(static\_cast<unsigned int>(time(nullptr)));

locale::global(locale(""));

#ifdef \_WIN32

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

#endif

}

bool validate\_input(istream& input, int& value, const char\* prompt) {

cout << prompt;

if (!(input >> value) || value <= 0) {

input.clear();

// игнорируем ввод до конца строки

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

return false;

}

return true;

}

bool validate\_input(istream& input, string& value, const char\* prompt) {

cout << prompt;

if (!(input >> value) || value == " " ) {

input.clear();

// игнорируем ввод до конца строки

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

return false;

}

return true;

}

bool validate\_input(istream& input, int& value1, int& value2, const char\* prompt) {

cout << prompt;

if (!(input >> value1 >> value2) || value1 <= 0 || value2 <= 0) {

input.clear();

// игнорируем ввод до конца строки

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

return false;

}

return true;

}

void print\_student(Student \*st) {

if (st) cout << "Найден: " << st->id << " | " << st->surname << " " << st->firstname << " | " << st -> faculty << "\n";

else cout << "Не найден" << endl;

}

void print\_student(const std::vector<Student> &sts) {

if(sts.size() == 0) cout << "Не найден" << endl;

else

for (const auto& st : sts) {

cout << "Найден: " << st.id << " | " << st.surname << " " << st.firstname << " | " << st.faculty << "\n";

}

}

bool contains\_substring(const Student& student, const std::string& substring) {

return student.surname.find(substring) != std::string::npos ||

student.firstname.find(substring) != std::string::npos ||

student.faculty.find(substring) != std::string::npos;

}

bool contains\_any(const Student& student, const std::string& substring) {

return student.surname.find(substring) != std::string::npos ||

student.firstname.find(substring) != std::string::npos ||

student.faculty.find(substring) != std::string::npos ||

to\_string(student.id).find(substring) != std::string::npos;

}