МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Кафедра «Вычислительной техники»

**ОТЧЁТ**

по лабораторной работе №1  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Простые структуры данных»

Выполнил:

Студент группы 23ВВВ2

Чупраков С. В.

Приняли:

Митрохин М. А.  
Юрова О.В.

Пенза 2024

**Цель работы**

Изучение и повторение простых структур данных.

**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.).

**Листинг**

Задание 1

﻿// Laba\_1.cpp

// part\_1

#include <iostream>

int main()

{

int min = INT\_MAX, max = INT\_MIN;

int a[10] = { 0 };

for (int i = 0; i < 10; i++) {

a[i] = i;

printf("%d ", a[i]);

}

for (int i = 0; i < 10; i++) {

if (a[i] < min) {

min = a[i];

};

}

for (int i = 0; i < 10; i++) {

if (a[i] < min) {

min = a[i];

};

}

printf("\n%d", max - min);

printf("\n%d", max);

printf("\n%d", min);

return 0;

}

Задание 2

﻿// Laba\_1.cpp

// part\_2

#include <iostream>

int main()

{

int min = 0, max = 100, m = 0;

int a[10] = { 0 };

srand(time(NULL));

for (int i = 0; i < 10; i++) {

m = min + rand() % (max - min + 1);

a[i] = m;

printf("%d ", a[i]);

}

return 0;

}

Задание 3

﻿// Laba\_1.cpp

// part\_3

#include <iostream>

int main()

{

int\* a;

int n, z = 0;

printf("Input massive size: ");

scanf\_s("%d", &n);

a = (int\*)malloc(n \* sizeof(int));

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

printf("Input element num %d : ", i + 1);

scanf\_s("%d", &z);

a[i] = z;

}

printf("\n");

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

printf("%d ", a[i]);

}

return 0;

}

Задание 4

﻿// Laba\_1.cpp

// part\_4

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <malloc.h>

int main()

{

int\* a;

int i, j, n, m;

int sum=0 ;

printf("Input row: ");

scanf("%d", &n);

printf("Input column: ");

scanf("%d", &m);

a = (int\*)malloc(n \* m \* sizeof(int));

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

{

for (j = 0; j < m; j++)

{

printf("a[%d][%d] = ", i, j);

scanf("%d", (a + i \* m + j));

}

}

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

for (j = 0; j < m; j++)

{

sum = sum + \*(a + i \* m + j);

}

printf("Sum in the row num %d = %d\n", i+1, sum);

sum = 0;

}

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

{

for (j = 0; j < m; j++)

{

printf("%4d ", \*(a + i \* m + j));

}

printf("\n");

}

free(a);

getchar();

return 0;

}

Задание 5

// Laba\_1.cpp

// part\_5

# #define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct student {

char Name[50];

char Surname[50];

char Faculty[50];

int Age;

};

int main() {

struct student\* students;

int num\_students, i;

char search\_name[50], search\_surname[50];

int found = 0;

printf("Enter the number of students: ");

scanf("%d", &num\_students);

students = (struct student\*)malloc(num\_students \* sizeof(struct student));

if (students == NULL) {

printf("Memory allocation failed\n");

return 1;

}

printf("Enter information about the students:\n");

for (i = 0; i < num\_students; i++) {

printf("Student %d:\n", i + 1);

printf("Name: ");

scanf("%s", students[i].Name);

printf("Surname: ");

scanf("%s", students[i].Surname);

printf("Faculty: ");

scanf("%s", students[i].Faculty);

printf("Age: ");

scanf("%d", &students[i].Age);

}

printf("\nEnter the name and surname to search: ");

scanf("%s %s", search\_name, search\_surname);

for (i = 0; i < num\_students; i++) {

if (strcmp(students[i].Name, search\_name) == 0 && strcmp(students[i].Surname, search\_surname) == 0) {

found = 1;

printf("\nInformation about the student:\n");

printf("Name: %s\n", students[i].Name);

printf("Surname: %s\n", students[i].Surname);

printf("Faculty: %s\n", students[i].Faculty);

printf("Age: %d\n", students[i].Age);

break;

}

}

if (!found) {

printf("\nStudent not found\n");

}

free(students);

return 0;

}

**Пояснительный текст к программе**

Задание 1

Программа создает массив из 10 элементов и заполняет его значениями от 0 до 9. Затем она находит минимальное и максимальное значение в массиве и выводит их разницу, а также сами минимальное и максимальное значения.

Задание 2

Программа создает массив из 10 элементов и заполняет его случайными значениями с использованием функции srand() в пределах максимального и минимального числа.

Задание 3

Программа запрашивает у пользователя размер массива и элементы этого массива, а затем выводит введенные пользователем элементы массива. malloc() - функция для выделения памяти под массив.

Задание 4

Программа предназначена для вычисления суммы элементов в каждой строке двумерного динамического массива.

Функции, используемые в программе:

malloc() - выделяет динамическую память под массив.

free() - освобождает выделенную динамическую память.

Программа запрашивает у пользователя количество строк и столбцов двумерного массива, затем заполняет массив значениями, вводимыми с клавиатуры. После этого вычисляет сумму элементов в каждой строке массива и выводит результат, а также исходный массив. В конце освобождает выделенную память.

Задание 5

Программа предоставляет возможность пользователю ввести информацию о студентах: имя, фамилию, факультет и возраст. Затем пользователю предлагается ввести имя и фамилию студента для поиска. Если такой студент есть в базе данных, программа выводит информацию о нем (имя, фамилия, факультет, возраст). Если студент не найден, программа сообщает об этом.

Функции, используемые в программе:

int main() - основная функция программы, которая включает в себя ввод данных о студентах, поиск студента по имени и фамилии и вывод информации о нем.

malloc() - выделяет динамическую память под массив структур student.

После завершения работы с массивом студентов, память освобождается с помощью функции free().

### Результаты работы программы

Результаты работы программы показаны на рисунках 1-5.

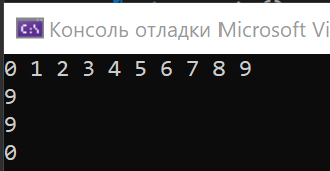


Рисунок 1 — Результаты работы программы пункта №1

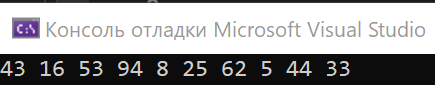


Рисунок 2 — Результаты работы программы пункта №2

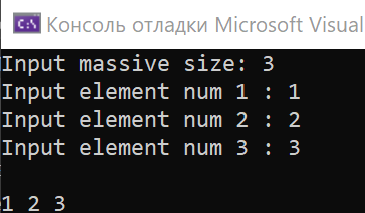


Рисунок 3 — Результаты работы программы пункта №3

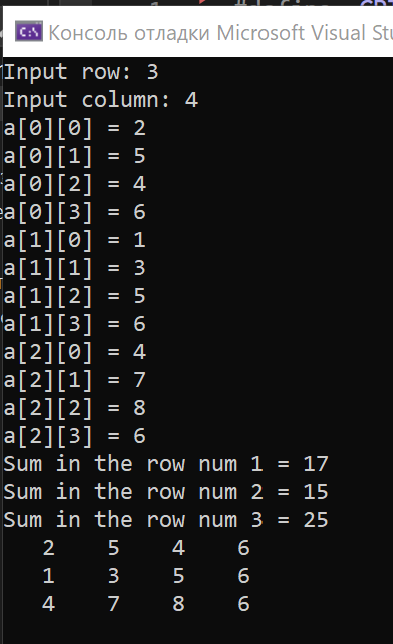


Рисунок 4 — Результаты работы программы пункта №4

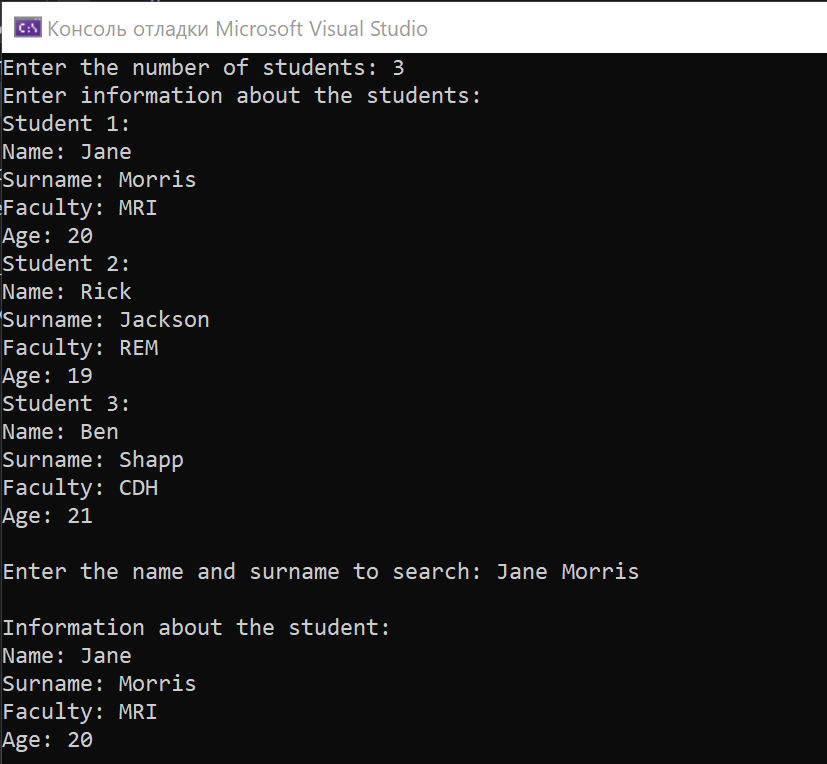


Рисунок 5 — Результаты работы программы пункта №5

### Тестировка

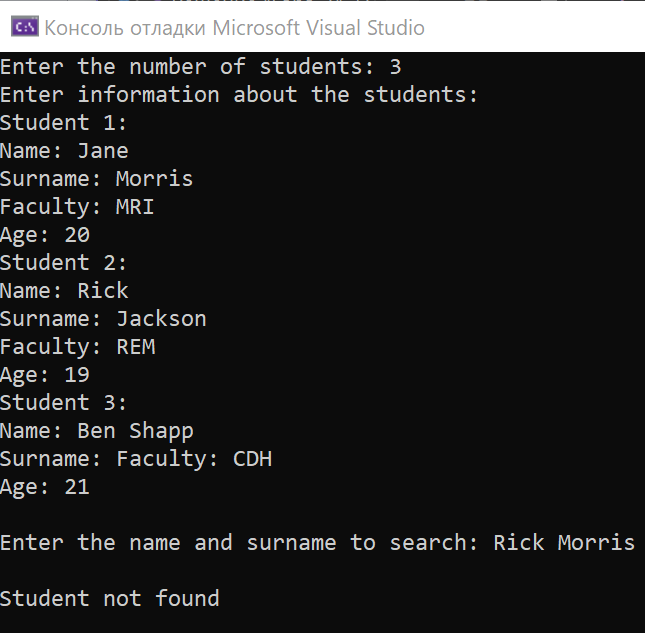


Рисунок 6 — Тестировка программы пункта №5

### Вывод

В ходе выполнения лабораторной работы была разработана программы, выполняющие задания Лабораторной работы №1. В процессе выполнения работы были использованы знания о простейших структурах данных.