Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №1  
по дисциплине: ««Логика и основы алгоритмизации в инженерных задачах»  
на тему «Простые структуры данных»

Выполнили студенты группы 22ВВВ2:  
Беляев Д.

Приняли:  
Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Название**

Простые структуры данных

**Цель работы**

Вспомнить основные моменты при работе с простыми структурами 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.

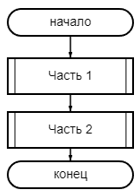
Задание 4: написать программу, вычисляющую сумму значений в каждом столбце (или строке) двумерного массива.

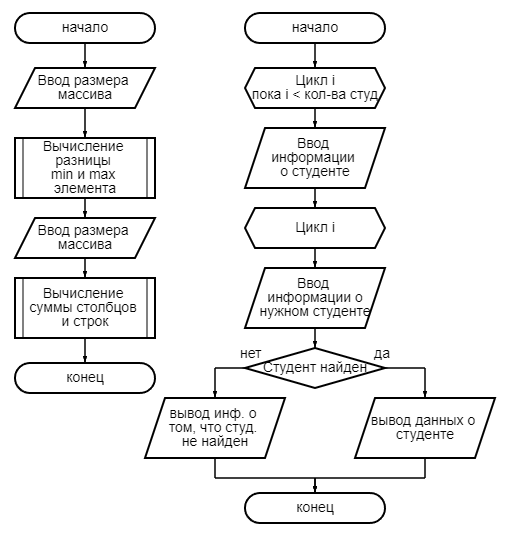
Задание 5: написать программу, осуществляющую поиск среди структур student структуру с заданными параметрами (фамилией, именем и т.д.).

**Метод решения задач**

Объявили и инициализировали переменные в С. С помощью отдельных функций создадим массивы, заполненные случайными числами, вычислим разницу минимального и максимального элементов одномерного массива, сумму строк и столбцов двумерного массива, нахождение элемента в массиве по его параметрам.

**Блок-схема программы**

****



**Листинг**

* main.c

#include <stdio.h>

#include "steps.h"

#include <time.h>

int main()

{

Part1();

Part2();

return 0;

}

* arrayPart.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

const int SINGLE\_ARRAY\_SIZE;

const int DOUBLE\_ARRAY\_SIZE\_X;

const int DOUBLE\_ARRAY\_SIZE\_Y;

#pragma region SingleArray

int\* GenerateSingleArray(int size)

{

int\* array = (int\*)malloc(size \* sizeof(int));

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

{

array[i] = 10 + rand() % 90;

}

return array;

}

int FindMinMaxDiff(int\* array, int size)

{

int min = \*array;

int max = \*array;

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

{

int curEl = array[i];

if (curEl < min)

min = curEl;

if (curEl > max)

max = curEl;

}

printf("Min: %d, Max: %d\n", min, max);

return max - min;

}

#pragma endregion

#pragma region DoubleArray

int\*\* GenerateDoubleArray(int sizeX, int sizeY)

{

int\*\* array = (int\*\*)malloc(sizeX \* sizeof(int\*));

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

{

array[i] = GenerateSingleArray(sizeY);

}

return array;

}

void CalculateColumnRowSum(int\*\* array, int sizeX, int sizeY)

{

int rowSum = 0;

printf("Rows sum:\n");

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

{

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

{

rowSum += array[i][j];

}

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

rowSum = 0;

}

printf("----------\n");

printf("Columns sum:\n");

int colSum = 0;

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

{

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

{

colSum += array[j][i];

}

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

colSum = 0;

}

}

#pragma endregion

void writeArray(int\* array, int size)

{

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

{

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

}

printf("\n");

}

void Part1()

{

srand(time(NULL));

// Single array part

printf("----------\n");

printf("Insert single array length: ");

scanf("%d",&SINGLE\_ARRAY\_SIZE);

int\* singleArray = GenerateSingleArray(SINGLE\_ARRAY\_SIZE);

printf("Array: ");

writeArray(singleArray, SINGLE\_ARRAY\_SIZE);

printf("Min/Max difference: %d \n", FindMinMaxDiff(singleArray, SINGLE\_ARRAY\_SIZE));

free(singleArray);

printf("----------\n");

// Double array size

printf("----------\n");

printf("Insert double array length x: ");

scanf("%d", &DOUBLE\_ARRAY\_SIZE\_X);

getchar();

printf("Insert double array length y: ");

scanf("%d", &DOUBLE\_ARRAY\_SIZE\_Y);

int\*\* doubleArray = GenerateDoubleArray(DOUBLE\_ARRAY\_SIZE\_X, DOUBLE\_ARRAY\_SIZE\_Y);

printf("Array:\n");

for (int i = 0; i < DOUBLE\_ARRAY\_SIZE\_X; i++)

{

writeArray(doubleArray[i], DOUBLE\_ARRAY\_SIZE\_Y);

}

printf("----------\n");

CalculateColumnRowSum(doubleArray, DOUBLE\_ARRAY\_SIZE\_X, DOUBLE\_ARRAY\_SIZE\_Y);

for (int i = 0; i < DOUBLE\_ARRAY\_SIZE\_X; i++)

{

free(doubleArray[i]);

}

free(doubleArray);

printf("----------\n");

}

* structPart.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define STRING\_SIZE 20

struct Student\_s

{

int gradebook;

char surname[STRING\_SIZE];

char name[STRING\_SIZE];

char faculty[STRING\_SIZE];

} StudentDefault = {-1, "\n", "\n", "\n" };

typedef struct Student\_s Student;

Student CreateNewStudent()

{

Student newStudent;

printf("Insert student surname: ");

fgets(newStudent.surname, sizeof(newStudent.surname), stdin);

fseek(stdin, 0, SEEK\_END);

printf("Insert student name: ");

fgets(newStudent.name, sizeof(newStudent.name), stdin);

fseek(stdin, 0, SEEK\_END);

printf("Insert student faculty: ");

fgets(newStudent.faculty, sizeof(newStudent.faculty), stdin);

fseek(stdin, 0, SEEK\_END);

printf("Insert student's gradebook number: ");

scanf("%d", &newStudent.gradebook);

fseek(stdin, 0, SEEK\_END);

return newStudent;

}

Student\* FindSudentByName(Student\* inst, int instSize, Student wantFind)

{

Student tester = StudentDefault;

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

{

Student cur = \*(inst + i\*sizeof(Student));

if (strcmp(tester.name, wantFind.name) != 0)

if (strcmp(cur.name, wantFind.name) != 0)

continue;

if (strcmp(tester.surname, wantFind.surname) != 0)

if (strcmp(cur.surname, wantFind.surname) != 0)

continue;

if (strcmp(tester.faculty, wantFind.faculty) != 0)

if (strcmp(cur.faculty, wantFind.faculty) != 0)

continue;

if (tester.gradebook != wantFind.gradebook)

if (cur.gradebook != wantFind.gradebook)

continue;

return &cur;

}

return NULL;

}

void Part2()

{

fseek(stdin, 0, SEEK\_END);

printf("----------\n");

int instSize = 2;

printf("Count od students: ");

scanf("%d", &instSize);

fseek(stdin, 0, SEEK\_END);

Student\* institute = (Student\*)malloc(instSize \* sizeof(Student));

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

{

printf("----------\n");

\*(institute + i\* sizeof(Student)) = CreateNewStudent();

printf("----------\n");

}

printf("----------\nWhat student do you want to find?\n");

Student wantFind = CreateNewStudent();

Student\* finded = FindSudentByName(institute, instSize, wantFind);

if (finded == NULL)

{

printf("No student with this params");

printf("----------\n");

return;

}

Student stud = \*finded;

printf("----------\n");

printf("Student Finded:\nname %ssurname %sfaculty %sgradebook %d\n", stud.name, stud.surname, stud.faculty, stud.gradebook);

printf("----------\n");

printf("----------\n");

free(institute);

}

* steps.h

#ifndef STEPS\_H

#define STEPS\_H

// array part

void Part1();

// struct part

void Part2();

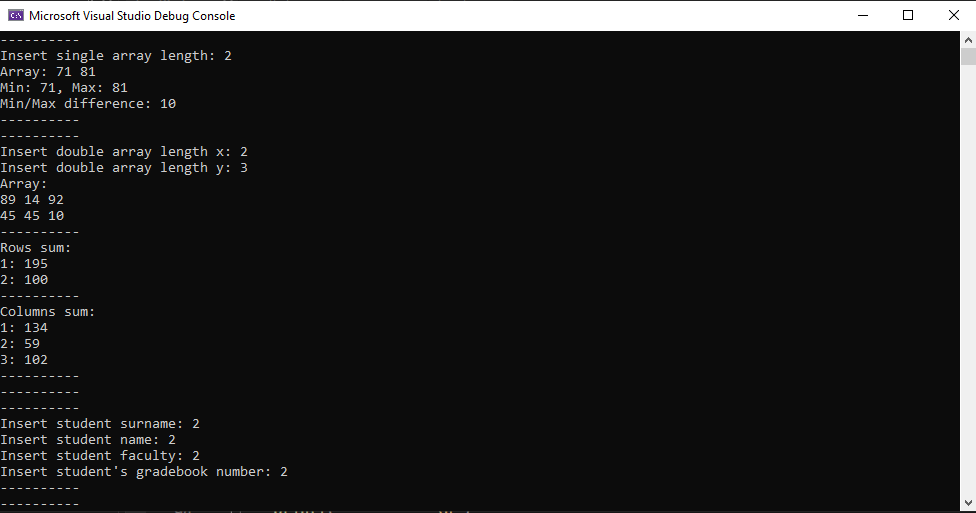
#endif

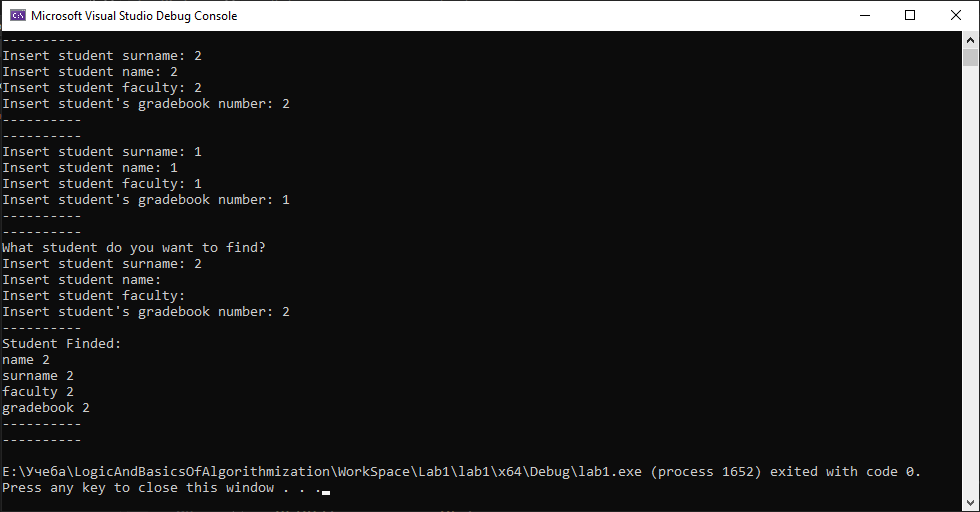
**Пояснительный текст к программе**

В первой части программы производится работа с массивами. В первую очередь создаем массив, заполненный случайными числами, и выводим его на экран. Далее находим максимальный и минимальный элемент, выводим их, выводим разницу между ними, очищаем память. Во вторую очередь создаем двумерный массив, заполненный случайными числами, и выводим его на экран. Далее находим сумму элементов в каждом столбце и строчке, выводим их. Очищаем память.

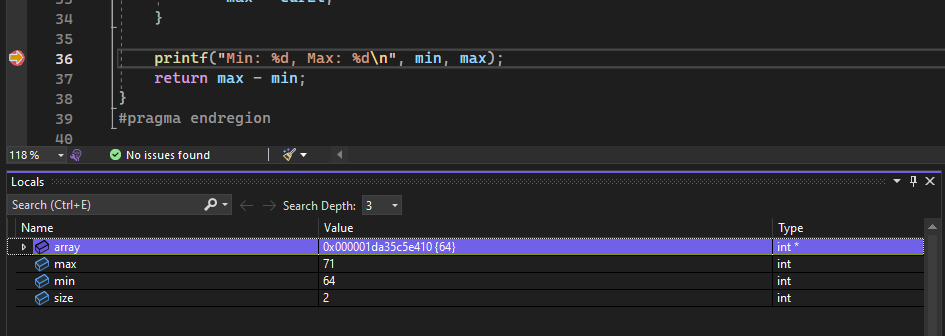
Во второй части программы производится работа со структурами. В первую очередь создаём «институт», состоящий из определенного количества учеников. Задаем информацию у них. Далее запрашиваем информацию об ученике, которого нужно найти. Если он найден – выводим на экран информацию о нем, иначе выводим сообщение о том, что такого студента нет.

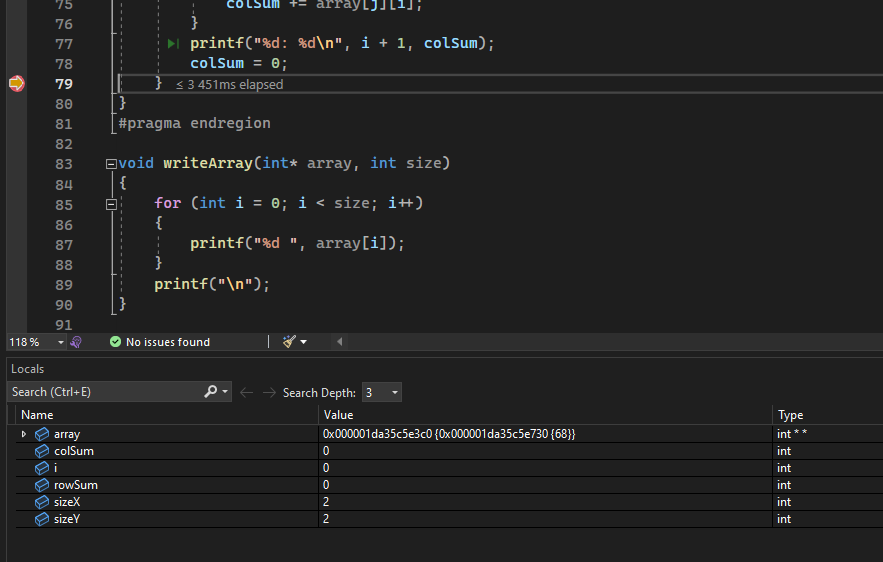
**Результат работы программы**

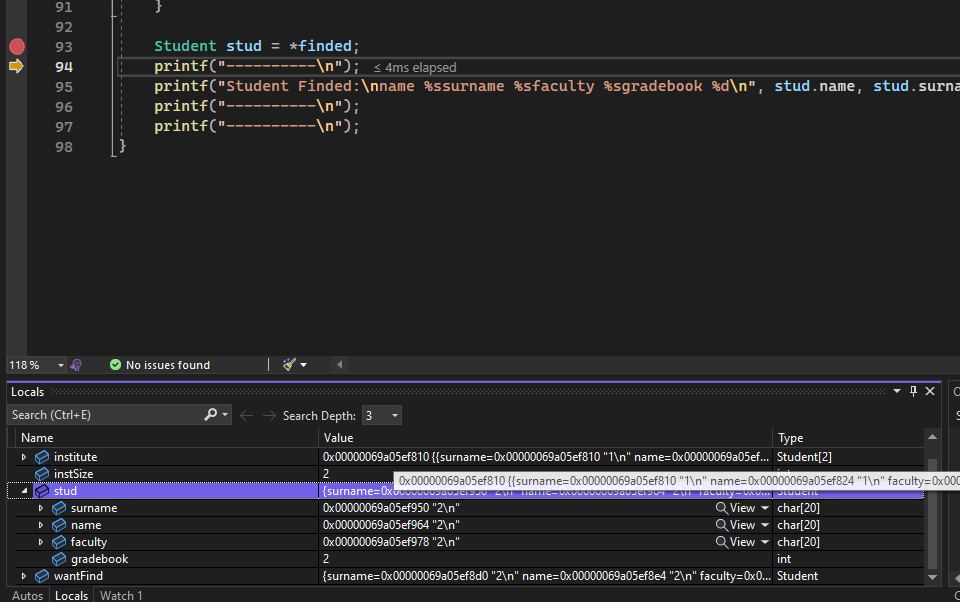




**Протокол трассировки программы**







**Вывод**

Я вспомнил основные моменты при работе с простыми структурами данных