Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная кафедра»

**ОТЧЕТ**

по лабораторной работе №1

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Простые структуры данных»

Выполнила:

студентка группы 22ВВП1

Расторгуева К.В.

Приняли:

Акифьев И.В.

Митрохин М.А.

Пенза 2023

**Название**

Простые структуры данных.

**Цель работы**

Повторение простых структур данных: массивов, строк, структур.

**Лабораторное задание**

1. Написать программу, вычисляющую разницу между максимальным и минимальным элементами массива.
2. Написать программу, реализующую инициализацию массива случайными числами.
3. Написать программу, реализующую создание массива произвольного размера, вводимого с клавиатуры.
4. Написать программу, вычисляющую сумму значений в каждом столбце (или строке) двумерного массива.
5. Написать программу, осуществляющую поиск среди структур student структуру с заданными параметрами (фамилией, именем и т.д.).

**Программа**

Задания 1-4:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

int n, m;

int\* pmas;

int i, j;

int sum\_c[1][10]; //сумма столбцов

int sum\_r[10][1]; //сумма строк

int max, min;

int k ; //временная переменная

void main(){

srand(time(NULL));

setlocale(0, "rus");

printf("Введите количество строк в массиве -> ");

scanf("%d", &n);

printf("Введите количество столбцов в массиве -> ");

scanf("%d", &m);

printf("\n");

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

for (i = 0; i < n; i++) { //инициализацию массива произвольного размера случайными числами

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

pmas[i \* m + j] = rand()%100-20;

printf ("%5d ", pmas[i \* m + j]);

}

printf("\n");

}

printf("\n");

min = pmas[m]; //вычисление разницы между максимальным и

max = pmas[m]; //минимальным элементами массива

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

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

if (pmas[i \* m + j] > max) {

max = pmas[i \* m + j];

}

if (pmas[i \* m + j] < min) {

min = pmas[i \* m + j];

}

}

}

printf("Минимальный элемент массива = %d\n", min);

printf("Максимальный элемент массива = %d\n", max);

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

for (i = 0; i < n; i++) { //вычисление суммы значений в каждой строке

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

sum\_r[k][0] += pmas[i \* m + j];

}

k++;

}

printf("Сумма строк:\n");

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

printf("%5d\n", sum\_r[k][0]);

}

printf("\n");

k = 0; //вычисление суммы значений в каждом столбце

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

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

sum\_c[0][k] += pmas[i \* m + j];

}

k++;

}

printf("Сумма столбцов:\n");

for (k = 0; k < m; k++) {

printf("%5d ", sum\_c[0][k]);

}

printf("\n");

free(pmas);

}

Задание 5:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

int col\_student = 3;

int i, user\_answer, user\_answer\_find;

int number = 1;

char tmp\_find[40];

char tmp\_find\_save[40];

char research\_famil[20];

char tmp\_name[20];

char research\_name[20];

char research\_facult[20];

int nomzach\_research;

struct student{

char famil[20];

char name[20];

char facult[20];

int Nomzach;

}stud[3];

int main() {

SetConsoleCP(1251);

setlocale(0, "rus");

printf("1) Ввести данные с клавиатуры\n");

printf("2) Загрузить данные из файла\n");

scanf("%d", &user\_answer);

switch (user\_answer) {

case 1: //ввод данных с клавиатуры

{

for (i = 0; i < col\_student; i++) {

printf("Введите фамилию %d-го студента -> ", number);

scanf("%20s", stud[i].famil);

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите имя %d-го студента -> ", number);

scanf("%20s", stud[i].name);

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите название факультета %d-го студента -> ", number);

scanf("%20s", stud[i].facult);

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите номер зачётной книжки %d-го студента -> ", number);

scanf("%d", &stud[i].Nomzach);

number++;

}

printf("\n");

break;

}

case 2: //ввод данных из файла

{

FILE\* file;

file = fopen("bd.txt", "r");

for (i = 0; i < col\_student; i++) {

fgets(stud[i].famil, 20, file);

strtok(stud[i].famil, "\n");

fgets(stud[i].name, 20, file);

strtok(stud[i].name, "\n");

fgets(stud[i].facult, 20, file);

strtok(stud[i].facult, "\n");

fscanf\_s(file, "%d", &stud[i].Nomzach);

fscanf\_s(file, "\n");

}

printf("\n");

}

}

for (i = 0; i < col\_student; i++) { //вывод всего списка студентов

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n",

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

printf("\n");

printf("1) Поиск по имени и фамилии\n");

printf("2) Поиск по факультету\n");

printf("3) Поиск по номеру зачетной книжки\n");

printf("0) Выход\n");

do {

printf("\nВведите выбранные вариант -> ");

scanf("%d", &user\_answer\_find);

switch (user\_answer\_find) {

case 1: //ввод данных с клавиатуры

{

printf("Введите фамилию и имя студента для поиска -> "); //поиск заданного студента

getchar();

fgets(tmp\_find, 20, stdin);

if (strpbrk(tmp\_find, " ") == NULL) {

printf("Данные некорректны");

}

else {

strcpy(tmp\_find\_save, tmp\_find);

strcpy(research\_famil, strtok(tmp\_find, " "));

strcpy(tmp\_name, strpbrk(tmp\_find\_save, " ")); //strbrk - поиск первого вхождения

memmove(research\_name, tmp\_name + 1, (strlen(tmp\_name) - 2));

for (i = 0; i < col\_student; i++) {

if ((strcmp(stud[i].famil, research\_famil) == 0) && (strcmp(stud[i].name, research\_name) == 0)) {

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n",

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

}

break;

}

case 2:

{

printf("Введите факультет -> "); //поиск заданного студента

getchar();

fgets(research\_facult, 20, stdin);

strtok(research\_facult, "\n");

for (i = 0; i < col\_student; i++) {

if (strcmp(stud[i].facult, research\_facult) == 0) {

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n",

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

break;

}

case 3:

{

printf("Введите номер зачетной книжки -> "); //поиск заданного студента

scanf("%d", &nomzach\_research);

for (i = 0; i < col\_student; i++) {

if (stud[i].Nomzach == nomzach\_research) {

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n",

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

break;

}

}

} while (user\_answer\_find != 0);

return 0;

}

**Пояснительный текст к программе**

Первые строки программы указывают на необходимость включения в файл несколькмх заголовочных файлов (*\*.h*). Именно они содержат основные функции, которые использованы в программе (*printf(), scanf(), srand(), time()*).

Задание 1-4.

После объявления переменных следует объявление главной функции *main()* с указанием имени функции и типа возвращаемого значения (*void* – функция ничего не возвращает). Далее описываются функции *srand()* и *time()*, которые позволяют каждый раз заполнять массив уникальными случайными числами при повторном запуске программы. Функции *srand()* и *time()* являются базой для функции генерации случайных значений *rand()*. Выражение *srand(время (NULL))* использует внутренние часы компьютера для управления выбором начального значения. Поскольку время постоянно меняется, начальное значение так же постоянно меняется.

//Простые структуры данных - лабораторная работа №1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

//объявление переменных

int n, m; //количество строк, столбцов в массиве

int\* pmas; //указатель на массив

int i, j; //переменные для организации циклов

int sum\_c[1][10]; //переменная для подсчета суммы столбцов в массиве

int sum\_r[10][1]; //переменная для подсчета суммы строк в массиве

int max, min; //переменные для поиска максимального и минимального значений в массиве

int k; //временная переменная

void main(){

srand(time(NULL)); //функция генерации уникальных значений при каждом запуске программы

setlocale(0, "rus"); //подключение русского языка

printf("Введите количество строк в массиве -> "); //вывод информационного сообщения для пользователя

scanf("%d", &n); //считывание количества строк из консоли

printf("Введите количество столбцов в массиве -> "); //вывод информационного сообщения для пользователя

scanf("%d", &m); //считывание количества столбцов из консоли

printf("\n");

После введения пользователем размера массива происходит его инициализация. Для этого используется функция динамического выделения памяти malloc. Размер выделенной памяти равен произведению количества строк, количества столбцов и размера типа данных одного элемента массива. В данном случае используется целочисленный тип данный – int. Следовательно, размер одной ячейки массива равен 4 байта.

pmas = (int\*)malloc(n\*m \* sizeof(int)); //динамическое выделение памяти, равной размеру массива.

//размер массива = количество строк \* количество столбцов \* размер типа данных одного элемента массива

Заполнение массива случайными числами происходит с помощью вложенного цикла. Вложенный цикл for (j = 0; j < m; j++) организует проход по элементам строки. Внешний цикл for (i = 0; i < n; i++) организует переход на следующую строку. Необходимо обратить внимание, что индексация производится с нуля. Пример: рассмотрим элемент двумерного массива, индекс которого [1][2]. Для ссылки на данный элемент необходимо выражение *pmas*[6]. Индекс 6 вычисляется следующим выражением:

, где *i* – номер строки элемента, *m* – общее количество столбцов в массиве, *j* – номер столбца элемента.

//инициализация массива произвольного размера случайными числами

for (i = 0; i < n; i++) { //проход по строкам

for (j = 0; j < m; j++) { //проход по столбцам

pmas[i \* m + j] = rand()%100-20; //запись случайного значения [-20; 79] в элемент[i][j] массива

printf ("%5d ", pmas[i \* m + j]); //вывод элемента с использованием форматированного вывода

}

printf("\n");

}

printf("\n");

Вычисление минимального и максимального элемента происходит посредством сравнения текущего элемента с произвольным элементом, выбранного в качестве начального. Если текущий элемент больше/меньше максимального/минимального, то максимальный/минимальный элемент перезаписывается.

//вычисление разницы между максимальным и минимальным элементами массива

min = pmas[m]; //запись в пермеенную min одного из элементов массива max = pmas[m]; //запись в пермеенную max одного из элементов массива

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

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

if (pmas[i \* m + j] > max) { //сравнение элемента массива с max

max = pmas[i \* m + j]; //запись в переменную max большего значения

}

if (pmas[i \* m + j] < min) { //сравнение элемента массива с min

min = pmas[i \* m + j]; //запись в переменную min меньшего значения

}

}

}

printf("Минимальный элемент массива = %d\n", min); //вывод минимального значения

printf("Максимальный элемент массива = %d\n", max); //вывод максимального значения

printf("max - min = %d\n\n", max - min); //вывод разницы между максимальным и минимальным значениями

Массивы *sum\_r*[ ][ ] и *sum\_c*[ ][ ] являются вектором-столбцом и вектором-строкой.

*sum\_r*[ ][ ] – матрица, имеющая единственный столбец (вектор-столбец).

*sum\_с*[ ][ ] – матрица, имеющая единственную строку (вектор-строка).

Вычисление суммы значений в каждой строке/столбце осуществляется посредством формирования массива *sum\_r*[ ][ ] / *sum\_c*[ ][ ].

Для вычисления суммы значений в каждой строке производится обход по массиву. Сумма значений k-ой строки записывается в *sum\_r*[*k*][0]. Аналогично подсчитывается сумма значений в каждом столбце.

//вычисление суммы значений в каждой строке

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

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

sum\_r[k][0] += pmas[i \* m + j]; //прибавление к k-ому элементу массива sum\_r элемнетов строки

}

k++;

}

printf("Сумма строк:\n");

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

printf("%5d\n", sum\_r[k][0]); //форматированный вывод столбца с суммами строк

}

printf("\n");

k = 0; //вычисление суммы значений в каждом столбце

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

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

sum\_c[0][k] += pmas[i \* m + j]; //прибавление к k-ому элементу массива sum\_c элемнетов столбца

}

k++;

}

printf("Сумма столбцов:\n");

for (k = 0; k < m; k++) {

printf("%5d ", sum\_c[0][k]); //форматированный вывод строки с суммами строк

}

printf("\n");

free(pmas); //освобождение памяти, динамически выделенной при помощи функции malloc

}

Функция *free*() используется для освобождения ранее выделенной динамической памяти с помощью функции *malloc*()

Задание 5.

#define \_CRT\_SECURE\_NO\_WARNINGS

//подключение библиотек

#include <Windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

//объявление переменных

int col\_student = 3; //количество студентов

int i, user\_answer, user\_answer\_find; //перемнные для организации циклов, чтения ответов пользователя

int number = 1; //нумерация для вывода

char tmp\_find[40]; //временная переменная для организации поиска

char tmp\_find\_save[40]; //временная переменная для организации поиска

char research\_famil[20]; //переменная для организации поиска по фамилии студента

char tmp\_name[20]; //временная переменная для организации поиска по имени студента

char research\_name[20]; //переменная для организации поиска по имени студента

char research\_facult[20]; //переменная для организации поиска по факультету студента

int nomzach\_research; //переменная для организации поиска по номеру зачетки студента

struct student{ //объявление структуры

char famil[20];

char name[20];

char facult[20];

int Nomzach;

}stud[3];

После объявления переменных и структуры *student* следует объявление функции *main()* с указанием имени функции и типа возвращаемого значения (*int*). В случае успешного завершения программы функция возвращает значение «1» (return 0;). Далее пользователю предлагается данные загрузить из файла или ввести с клавиатуры.

int main() {

SetConsoleCP(1251); //подключение русского языка

setlocale(0, "rus");

printf("1) Ввести данные с клавиатуры\n"); //вывод информационного сообщения

printf("2) Загрузить данные из файла\n"); //вывод информационного сообщения

scanf("%d", &user\_answer); //чтение ответа пользователя

switch (user\_answer) {

При вводе данных с клавиатуры организуются четыре цикла, во время которых происходит запись данных, вводимых пользователем, в соответствующие поля структуры *student*.

case 1: //ввод данных с клавиатуры

{

for (i = 0; i < col\_student; i++) {

printf("Введите фамилию %d-го студента -> ", number);

scanf("%20s", stud[i].famil); //считывание фамилии студента

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите имя %d-го студента -> ", number);

scanf("%20s", stud[i].name); //считывание имени студента

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите название факультета %d-го студента -> ", number);

scanf("%20s", stud[i].facult); //считывание факультета студента

number++;

}

number = 1;

for (i = 0; i < col\_student; i++) {

printf("Введите номер зачётной книжки %d-го студента -> ", number);

scanf("%d", &stud[i].Nomzach); //считывание номера зачетки студента

number++;

}

printf("\n");

break;

}

При считывании строки из файла символ новой строки “\*n*” прекращает работу функции *fgets()*, но он считается допустимым, и поэтому добавляется к считываемым символам строки. Для того чтобы отсечь символ новой строки “\*n*” используется функция *strtok()*, которая возвращает все символы строки до “\*n*”.

case 2: //ввод данных из файла

{

FILE\* file;

file = fopen("bd.txt", "r"); //открытие файла для считывания

for (i = 0; i < col\_student; i++) {

fgets(stud[i].famil, 20, file); //построчное считывание файла

strtok(stud[i].famil, "\n"); //отсечение "/n" в конце строки

fgets(stud[i].name, 20, file);

strtok(stud[i].name, "\n");

fgets(stud[i].facult, 20, file);

strtok(stud[i].facult, "\n");

fscanf\_s(file, "%d", &stud[i].Nomzach);

fscanf\_s(file, "\n");

}

printf("\n");

}

}

После ввода данных осуществляется вывод полного списка студентов.

for (i = 0; i < col\_student; i++) { //вывод всего списка студентов

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n",

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

printf("\n");

Далее предлагается меню с конкретными параметрами для поиска.

printf("1) Поиск по имени и фамилии\n"); //вывод информационных сообщений

printf("2) Поиск по факультету\n");

printf("3) Поиск по номеру зачетной книжки\n");

printf("0) Выход\n");

Поиск по конкретным параметрам организуется с помощью оператора множественного выбора *switch case* по переменной *user\_answer\_find*. Для зацикливания программы оператор *switch case* прописывается внутри цикла *do while*, выход из которого возможен при введении пользователем значения «0».

Поиск студента по фамилии и имени происходит в том случае, если указано не менее 2 слов. Это проверяется следующим условием

if (strpbrk(tmp\_find, " ") == NULL) { //количество слов меньше двух

printf("Данные некорректны");

}

Если введены корректные данные, то начинается их обработка. Копия исходных данных сохраняется в *tmp\_find\_save*. Для нахождения фамилии с помощью функции *strcpy()* отсекается символ начала новой строки «\*n*». Результат без «\*n*» сохраняется *в research\_famil*. Для нахождения имени функция *strpbrk()* находит первое вхождение «\_» в исходной строке, функция *strcpy()* копирует символы строки, начиная с «\_».Функция *memmove()* отсекает «\_» в начале строки и «\*n*» в конце строки.

else {

strcpy(tmp\_find\_save, tmp\_find); //сохранение копии строки в tmp\_find\_save

strcpy(research\_famil, strtok(tmp\_find, " ")); //копирование символов до пробела в research\_famil

strcpy(tmp\_name, strpbrk(tmp\_find\_save, " ")); //сохранение имени в tmp\_name (strbrk - поиск первого вхождения)

memmove(research\_name, tmp\_name + 1, (strlen(tmp\_name) - 2)); //отсечение пробела до имени и "\n" после имени

for (i = 0; i < col\_student; i++) {

if ((strcmp(stud[i].famil, research\_famil) == 0) && (strcmp(stud[i].name, research\_name) == 0)) { //сравнение полей famil и name у каждого студента в структуре student со строкой, введенной пользователем

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n", //вывод при совпадении

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

}

Поиск студента по факультету происходит с помощью функции *strcmp()*.Функция возвращает нулевое значение при равенстве строк. Если строки равны организуется вывод полей структуры *i*-го студента.

case 2: //поиск студента по факультету

{

printf("Введите факультет -> ");

getchar(); //ожидание ввода данных пользователем

fgets(research\_facult, 20, stdin); //чтение данных пользоввателя

strtok(research\_facult, "\n"); //отсечение "/n" в конце строки

for (i = 0; i < col\_student; i++) {

if (strcmp(stud[i].facult, research\_facult) == 0) { //сравнение поля facult у каждого студента в структуре student со строкой, введенной пользователем

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n", //вывод при совпадении

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

break;

}

Аналогично поиску студента по факультету организован поиск студента по номеру зачетной книжки. За тем исключением, что сравнение происходит не строк, а целочисленных переменных (тип *int*)

case 3: //поиск студента по номеру зачетной книжки

{

printf("Введите номер зачетной книжки -> ");

scanf("%d", &nomzach\_research); //чтение данных пользоввателя

for (i = 0; i < col\_student; i++) {

if (stud[i].Nomzach == nomzach\_research) { //сравнение поля Nomzach у каждого студента в структуре student с числом, введенным пользователем

printf("Cтудент %s %s обучается на факультете %s, номер зачётной книжки % d \n", //вывод при совпадении

stud[i].famil, stud[i].name, stud[i].facult, stud[i].Nomzach);

}

}

break;

}

Назначение отдельных операторов программы указывается в виде комментариев. Программа завершает свою работу после нажатия на любую клавишу на клавиатуре.

**Результат выполнения программы**

Результат работы программы представлены на рис. 1-2.

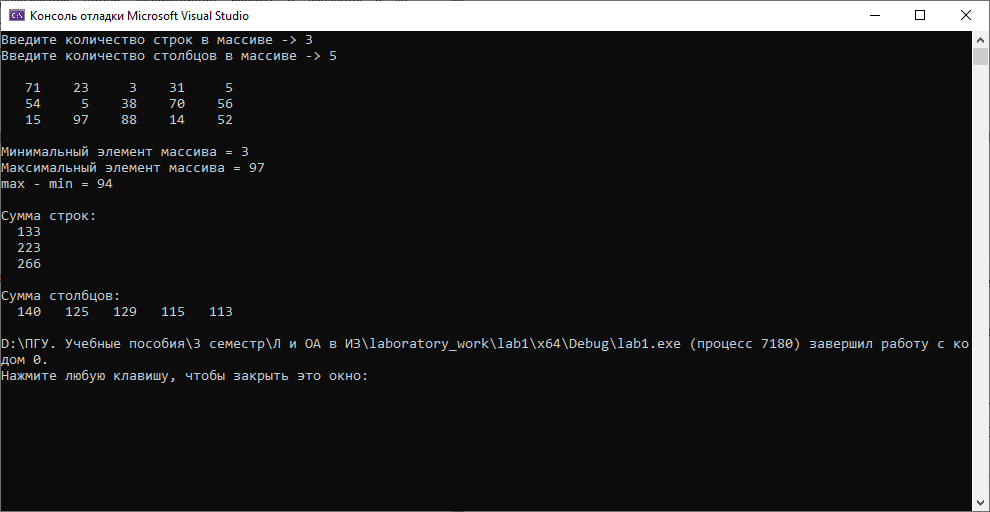


Рисунок 1 – работа с массивом

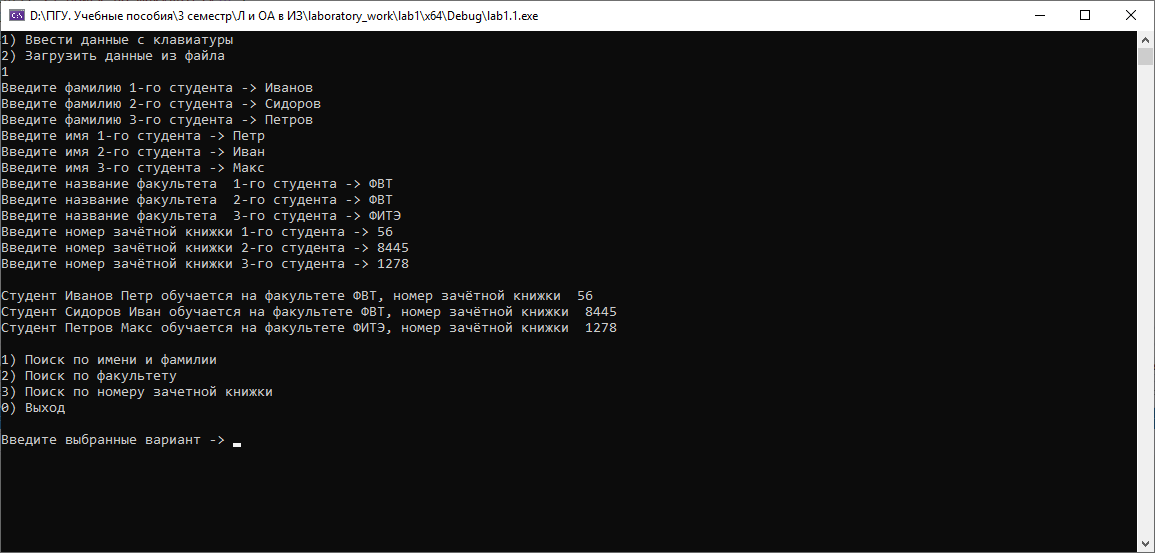


Рисунок 2 – ввод данных пользователем

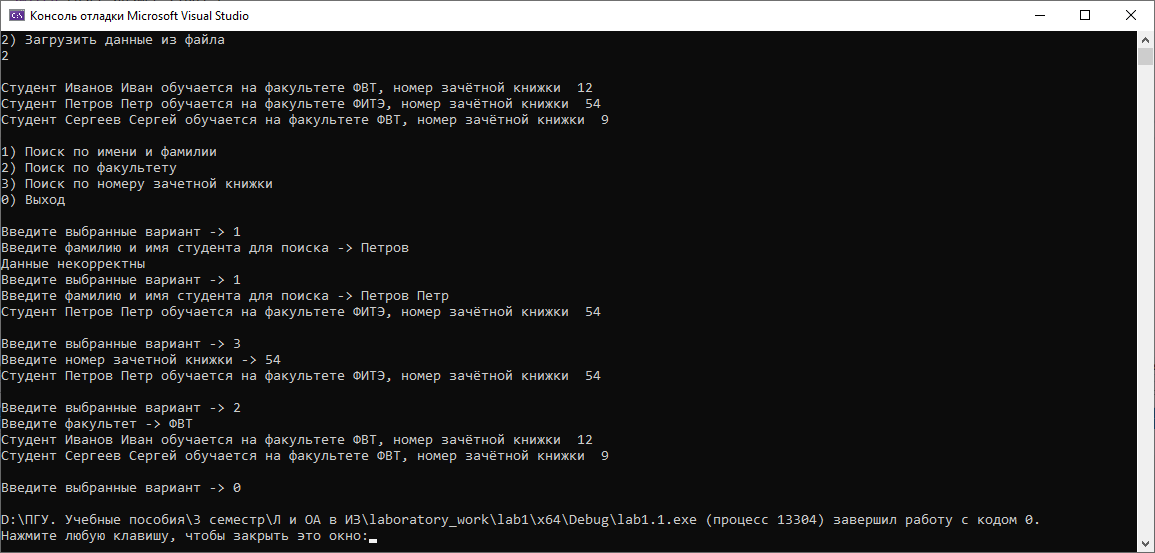


Рисунок 3 – работа со структурой

Результаты вычисления вручную совпали с вычислениями программы.

**Вывод**

В ходе выполнения лабораторной работы была разработана на языке Си программа, осуществляющая работу с двумерными массивами и структурами:

– вычисление разницы между максимальным и минимальным элементами массива

– инициализация массива случайными числами

– создание массива произвольного размера, вводимого с клавиатуры

– вычисление суммы значений в каждом столбце (строке) двумерного массива

– осуществление поиска среди структур структуры с заданными параметрами

Результаты работы программы совпали с результатами расчета вручную.