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»

Выполнили:

студенты группы 22BВВ1

Митрошин Ю.Е

Коннов А.Д

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2023

**Название**

Простые структуры данных

**Общие сведения.**

К простым структурам данных языка Си относятся массивы, строки (массивы символов). Структуры (struct) – простейшая составная структура данных.

**Массив** – это конечная совокупность данных одного типа.

Массивы могут состоять из целых чисел, чисел с плавающей запятой, символов и других типов данных. Существуют даже массивы массивов, – многомерные массивы.

Любой массив в программе должен быть описан: вначале указывается тип массива. затем его имя, после имени массива добавляют квадратные скобки ([ ]) внутри которых обычно стоит число, показывающее количество элементов массива. Например, запись

int x[10];

определяет x как массив из 10 целых чисел.

Для многомерных массивов количество пар квадратных скобок равно размерности массива, а число внутри скобок показывает размер массива по данному измерению. Например, описание двумерного массива выглядит так:

int a[2][5];.

Такое описание можно трактовать как матрицу из 2 строк и 5 столбцов.

Для обращения к некоторому элементу массива указывают его имя и индекс, заключенный в квадратные скобки (для многомерного массива - несколько индексов, заключенные в отдельные квадратные скобки):

a[1][3], x[i], a[0][k+2] и т.д.

В случае, если индексом является переменная, то она должна быть целого типа.

! Важно запомнить, что индексация массивов в языке Си начинается с нуля т.е. обращение к первому элементу массива выглядит так:

 a[0]; a[0][0];

Индекс может быть не только целой константой или целой переменной, но и любым выражением целого типа. Переменная с индексами в программе используется наравне с простой переменной (например, в операторе присваивания, в функциях ввода- вывода).

**Динамический массив**

Для использования функций динамического распределения памяти необходимо подключение библиотеки <malloc.h> или <stdlib.h> (в зависимости от компилятора и используемого стандарта языка).

Функции динамического выделения памяти

void\* malloc(*РазмерМассиваВБайтах*);  
 void\* calloc(*ЧислоЭлементов*, *РазмерЭлементаВБайтах*);

выделяют блок памяти, размером *РазмерМассиваВБайтах* или *ЧислоЭлементов*\**РазмерЭлементаВБайтах* байт, и возвращает указатель на начало блока.

Элементами массива могут быть как данные простых типов, так и составных типов (например, структуры), поэтому для точного определения размера элемента массива в общем случае используется функция

int sizeof(*ТипЭлемента*);

Эта функция возвращает количество байт, занимаемое элементом указанного типа.

Память, динамически выделенная с использованием функций calloc(), malloc(), должна быть освобождена после окончания использования функцией

 free(*указатель*);

Для примера, код

**int** \*a;  // указатель на массив

**int**  n;

...

n = 10;

   // Выделение памяти

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

...

a[1] = a[1] + a[3];

...

free(a); // Освобождение памяти

создаёт массив размером 10 элементов и после некоторых преобразований высвобождает занимаемую им память.

Для **генерации случайных чисел** в языке Си используется функция rand(), которая содержится в библиотеке <stdlib.h>.

Функция возвращает случайное целое число в диапазоне от нуля до RAND\_MAX. Глобальная константа RAND\_MAX является максимальным значением, которое может возвращаться функцией rand, это значение определено в заголовочном файле stdlib.h.

Чтобы ограничить сверху случайные числа, можно воспользоваться операцией получения остатка от деления. Остаток от деления на число всегда меньше этого числа. Поэтому для ограничения сверху случайных чисел возьмём остаток от деления на граничное значение.

int k;

k = rand()%100;

В приведенном примере значение k не превысит 99. Т.е. для rand()%n возвращается  любое число в диапазоне [0, n). Если необходимо получить числа в диапазоне [0, n], то необходимо изменить выражение на rand()%(n+1).

Для получения чисел из некоторого отрезка [min, max] необходимо найти остаток от деления на длину этого отрезка и выполнить сдвиг возвращаемого остатка на нижнюю границу отрезка

int k;

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

При повторных запусках программы функция rand() будет возвращать одинаковые последовательности чисел. Для получения уникальных последовательностей необходимо инициализировать генератор случайных чисел различными значениями. Для этого используется функция srand(), аргументом которой является инициализирующее целое число.

В качестве уникального для каждого запуска программы числа чаще всего используется текущее время, возвращаемое функцией time(), прототип которой описан в библиотеке time.h. Передав в time() в качестве параметра NULL, мы получим целое число, которое можно передать в srand(), установив, таким образом уникальное инициализирующее значение генератора

srand(time(NULL));

int k;

k = rand();

теперь генерируемая последовательность чисел k будет уникальной.

**Структура** - это совокупность переменных одного или нескольких типов, сгруппированных в один элемент.

Объявление структуры осуществляется с помощью ключевого слова struct, за которым идет ее тип и далее список элементов, заключенных в фигурные скобки:

     struct тип { тип элемента\_1 имя элемента\_1;

                    .........

                  тип элемента\_n имя элемента\_n;

                };

Именем элемента может быть любой идентификатор.

Определение структуры:

struct student

{

  int Nomzach;

  char famil[20];

  char name[20], facult[20];

 };

После фигурной скобки, заканчивающей список элементов, могут инициализироваться структуры заданного типа, например:

     struct student{...} Ivanov, Petrov;

Доступ к отдельным полям структуры осуществляется с помощью оператора . (оператор точка). Например, доступ к полю name в структуре Ivanov осуществляется так:

Ivanov.name;

**Практическая часть**

*Массивы*

*Пример работы с массивом*

Обратить массив и вывести его на экран

**#include** <stdio.h>

**int**   **main**(**void**)

    {

**int** temp,n=10,i=0;

**int** a[10]={0,1,2,3,4,5,6,7,8,9};

**while**(i<n/2)

        {

         temp=a[i];

            a[i]=a[9-i];

            a[9-i]=temp;

            i++;

        }

        i=0;

**while**(i<n) **printf**(" %d",a[i++]);

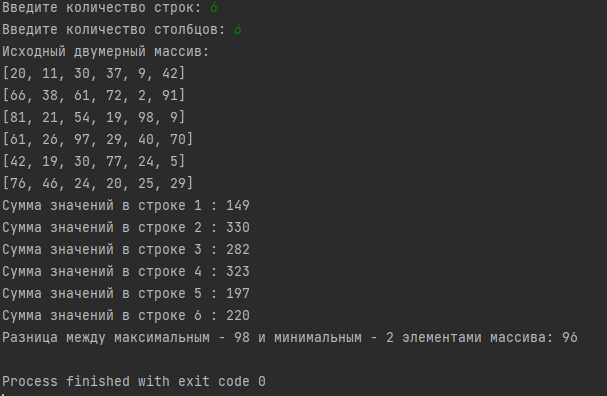
}

**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, вводим0ого с клавиатуры.

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



*Структуры*

*Пример работы со структурами*

Создать структуры student, ввести данные и вывести их на экран.

**int**   **main**(**void**)

{

**setvbuf**(stdin, NULL, \_IONBF, 0);

**setvbuf**(stdout, NULL, \_IONBF, 0);

**int** i;

**struct** student

{

**char** famil[20];

**char** name[20], facult[20];

**int** Nomzach;

} stud[3];

**for**(i=0;i<3;i++)

{

**printf**("Введите фамилию студента\n"); **scanf** ("%20s",stud[i].famil);

}

**for**(i=0;i<3;i++)

{

**printf**("Введите имя студента %s\n",stud[i].famil); **scanf** ("%20s",stud[i].name);

}

**for**(i=0;i<3;i++)

{

**printf**("Введите название факультета студента %s %s\n",stud[i].famil,stud[i].name); **scanf** ("%20s",stud[i].facult);

}

**for**(i=0;i<3;i++)

{

**printf**("Введите номер зачётной книжки студента %s %s\n",stud[i].famil,stud[i].name); **scanf** ("%d",&stud[i].Nomzach);

}

**for**(i=0;i<3;i++)

{

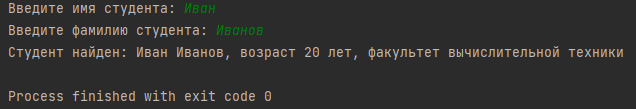
**printf**("Cтудент %s %s обучается на факультете %s, номер зачётной книжки %d \n",stud[i].famil,stud[i].name,

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

}

}

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



**Пояснительный текст к программе**

**Листинг**

Для заданий №1-4

import random

def calculate\_row\_sum(row):

row\_sum = 0

for element in row:

row\_sum += element

return row\_sum

rows = int(input("Введите количество строк: "))

cols = int(input("Введите количество столбцов: ")) # (задание 3)

# Инициализация двумерного массива случайными числами от 1 до 100 (задание 2,4)

array = []

for i in range(rows):

row = []

for i in range(cols):

row.append(random.randint(1, 100))

array.append(row)

# Вывод исходного двумерного массива

print("Исходный двумерный массив:")

for row in array:

print(row)

# Вычисление суммы значений в каждой строке и вывод результатов (задание 4)

row\_sums = []

for i in range(len(array)):

row\_sum = calculate\_row\_sum(array[i])

row\_sums.append(row\_sum)

print("Сумма значений в строке", i + 1, ":", row\_sum)

# Вычисление и вывод разности между максимальным и минимальным элементом массива (задание 1)

max\_element = array[0][0]

min\_element = array[0][0]

for row in array:

for element in row:

if element > max\_element:

max\_element = element

if element < min\_element:

min\_element = element

difference = max\_element - min\_element

print(f"Разница между максимальным - {max\_element} и минимальным - {min\_element} элементами массива:", difference)

Для задания №5

# Определение структуры student

class Student:

def \_\_init\_\_(self, first\_name, last\_name, age, facult):

self.first\_name = first\_name

self.last\_name = last\_name

self.age = age

self.facult = facult

# Функция для поиска студента по параметрам

def find\_student(students, first\_name, last\_name):

for student in students:

if student.first\_name == first\_name and student.last\_name == last\_name:

return student

return None

# Пример списка студентов

students = [

Student("Иван", "Иванов", 20, "факультет вычислительной техники"),

Student("Петр", "Петров", 22, "Факультет экономики и управления"),

Student("Мария", "Сидорова", 19, "Юридический факультет")

]

# Заданные параметры для поиска

search\_first\_name = input("Введите имя студента: ")

search\_last\_name = input("Введите фамилию студента: ")

# Вызываем функцию поиска

found\_student = find\_student(students, search\_first\_name, search\_last\_name)

if found\_student:

print(f"Студент найден: {found\_student.first\_name} {found\_student.last\_name}, возраст {found\_student.age} лет, {found\_student.facult}")

else:

print("Студент с заданными параметрами не найден")

**Вывод:** изучили простые структуры данных.