

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра Математического Обеспечения и Применения ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Программирование»**  
**Тема: Условия, циклы, оператор switch**

Студент гр. 0382

\_\_\_\_\_

Кондратов Ю.А.

Преподаватель

\_\_\_\_\_

Жангиров Т.Р.

Санкт-Петербург

2020

### **Цель работы.**

Изучение базовых конструкций языка Си.

### **Задание.**

Написать программу, выделив каждую подзадачу в функцию.

На вход программе подаётся одно из значений 0,1,2,3 и массив целых чисел размера не больше 100. Числа разделены пробелами. Строка заканчивается символом перевода строки.

В зависимости от значения, функция должна выводить следующее:

- 0: максимальное число в массиве (функция `max`);
- 1: минимальное число в массиве (функция `min`);
- 2: разницу между максимальным и минимальным элементом (функция `diff`);
- 3: сумму элементов массива, расположенных до первого минимального элемента (функция `sum`);

иначе необходимо вывести строку «Данные некорректны».

## **ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.**

В данной работе были использованы такие конструкции языка Си как:

- Функции стандартной библиотеки ввода-вывода:
  - `printf()` - выводит принимаемые значения на консоль;
  - `scanf()` - считывает входные данные из консоли;
- Операторы:
  - `if(){}` - если выражение в круглых скобках верно, выполняет блок кода в фигурных скобках;
  - `switch(){case x: ; default:}` - в зависимости от значения переменной в круглых скобках, выполняет блок когда, находящийся после «`case x:`»,

где  $x$  — значение переменной в круглых скобках. Если  $x$  не соответствует ни одному *case*, то выполняет блок кода, находящийся после «*default:*».

- Циклы:
  - *while()*{ } - на каждой итерации проверяется выражение в круглых скобках, если оно верно — выполняется блок кода в фигурных скобках, иначе — производится выход из цикла;
  - *for(<переменная>, <выражение 1>, <выражение 2>)*{ } - первым аргументом является переменная цикла, далее, если верно выражение 1 — выполняется блок кода в фигурных скобках и выражение 2, которое зачастую связано с переменной цикла;
- Пользовательские функции:
  - *<тип\_возвращаемого\_значения> имя\_функции (список\_параметров\_функции) {return <возвращаемое\_значение>;}* - при вызове в функции *main* выполняет блок кода в фигурных скобках, используя переданные параметры, и возвращает значение после оператора *return* (если тип возвращаемого значения не *void*).

### **Выполнение работы.**

Для решения поставленных задач необходимо считать данные, обработать их и вывести результат на консоль.

Для считывания входных данных используются переменные:

- *option* типа *int* — в этой переменной хранится значение управляющего символа (0, 1, 2 или 3);
- *array* массив типа *int* размера 100 элементов — массив, предназначенный для хранения массива целых чисел, введенных пользователем;

- *index* типа *int* с начальным значением ноль — переменная, хранящая текущее значения индекса нового элемента массива;
- *c* типа *char* — переменная, в которой хранится символ, введенный после числа.

Далее с помощью функции *scanf* в переменную *option* считывается управляющее значение, после чего с помощью цикла *while*, в каждой итерации которого проверяются условия:  $index < 100$  и  $c \neq ' '$ , и функцией *scanf* считывается очередной целочисленный элемент массива и следующий за ним символ, также значение переменной *index* увеличивается на 1 при помощи постфиксного инкремента.

При помощи оператора *switch*, в зависимости от значения переменной *option*, функцией *printf* выводится на консоль:

- значение функции *max* если  $option == 0$ ;
- значение функции *min* если  $option == 1$ ;
- значение функции *diff* если  $option == 2$ ;
- значение функции *sum* если  $option == 3$ ;
- строка «Данные некорректны» если *option* имеет другое значение.

Описание используемых функций:

1. Функция *int max (int ar[], int len)*.

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len*, хранящую длину массива. В целочисленную переменную *ans* записывается значение элемента массива с индексом 0 в качестве начального максимума.

Далее с помощью цикла *for* все элементы массива с индексами от 1 до значения длины массива проверяются оператором *if* на соответствие условию  $ar[i] > ans$ . Если условие верно, то значение предыдущего максимума,

записанное в переменной *ans* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден максимальный элемент массива.

С помощью оператора *return* функцией *main* будет возвращено значение элемента *ans*.

## 2. Функция *int min (int ar[], int len)*.

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len*, хранящую длину массива. В целочисленную переменную *ans* записывается значение элемента массива с индексом 0 в качестве начального минимума.

Далее с помощью цикла *for* все элементы массива с индексами от 1 до значения длины массива проверяются оператором *if* на соответствие условию  $ar[i] < ans$ . Если условие верно, то значение предыдущего минимума, записанное в переменной *ans* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден минимальный элемент массива.

С помощью оператора *return* функцией *main* будет возвращено значение элемента *ans*.

## 3. Функция *int diff (int ar[], int len)*.

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len*, хранящую длину массива.

Далее с помощью функции *max* находится максимальное значение элемента массива, а с помощью функции *min* — минимальное.

Функция возвращает разность значения *max* и значения *min*.

## 4. Функция *int sum (int ar[], int len)*.

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len*, хранящую длину массива. В целочисленную переменную *ans* записывается значение 0 в качестве начального значения суммы.

Далее с помощью цикла *for*, в каждой итерации которого к целочисленной переменной *i*, начальное значение которой равно 0, прибавляется единица, и, пока выполнено условие  $ar[i] > \min(ar, len)$  к переменной *ans* прибавляется значение *i*-го элемента массива, находится сумма всех элементов массива до первого минимального и записывается в переменную *ans*, значение которой возвращает функция.

Разработанный программный код см. в приложении А.

### Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 34 56 23 56 21 23 100\n	100	Программа работает правильно
2.	0 5 5 5 5 5\n	5	Программа работает правильно
3.	1 34 56 73 45 73 -100 100\n	-100	Программа работает правильно
4.	1 5 5 5 5 5\n	5	Программа работает правильно
5.	2 -100 2 3 4 5 6 100\n	200	Программа работает правильно
6.	2 5 5 5 5 5\n	0	Программа работает правильно
7.	3 1 1 1 1 0 1 1 0\n	4	Программа работает правильно
8.	3 1 1 1 1 1 1 1 1\n	0	Программа работает правильно

## Выводы.

В ходе работы были изучены основные управляющие конструкции языка Си.

Разработана программа, выполняющая считывание исходных с помощью функции `scanf()` и цикла `while(){}`  в переменную *option* и массив `array[100]`, условием которого было равенство переменной *c*, хранящей код символа между числами, коду символа пробела, написаны функции для обработки входных результатов, подробное описание которых приведено в разделе «выполнение работы», с помощью оператора `switch(){}`  и функции `printf()` реализован вывод результата определённой функции в зависимости от входного управляющего значения *option*:

- если *option* = 0 — выводится результат функции `int max()`;
- если *option* = 1 — выводится результат функции `int min()`;
- если *option* = 2 — выводится результат функции `int diff()`;
- если *option* = 3 — выводится результат функции `int sum()`;

Если значение *option* не соответствует ни одному из перечисленных — выводится строка «Данные некорректны».

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>

int max(int ar[], int len){
    int ans = ar[0];
    for (int i = 1; i < len; i++){
        if (ar[i] > ans){
            ans = ar[i];
        }
    }
    return ans;
}

int min(int ar[], int len){
    int ans = ar[0];
    for (int i = 1; i < len; i++){
        if (ar[i] < ans){
            ans = ar[i];
        }
    }
    return ans;
}

int diff(int ar[], int len){
    int ans = max(ar, len) - min(ar, len);
    return ans;
}

int sum(int ar[], int len){
    int ans = 0;
    for(int i = 0; ar[i] > min(ar, len); i++){
        ans += ar[i];
    }
    return ans;
}

int main(){

    int array[100], index = 0, option;
    char c = ' ';

    scanf("%d", &option);

    while(index < 100 && c == ' '){
        scanf("%d%c", &array[index++], &c);
    }

    switch (option){
        case 0:
```



```
        printf("%d", max(array, index));  
        break;  
    case 1:  
        printf("%d", min(array, index));  
        break;  
    case 2:  
        printf("%d", diff(array, index));  
        break;  
    case 3:  
        printf("%d", sum(array, index));  
        break;  
    default:  
        printf("Данные некорректны");  
    }  
  
    return 0;  
}
```