

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

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Программирование»**  
**Тема: Управляющие конструкции языка Си**

Студент гр. 0382

Злобин А. С.

Преподаватель

Чайка К. В.

Санкт-Петербург

2020

### **Цель работы.**

Изучить основные управляющие конструкции C: Условия, циклы, оператор switch

### **Задание.**

Напишите программу, выделив каждую подзадачу в отдельную функцию. Реализуйте программу, на вход которой подается 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 : индекс первого отрицательного элемента. (index\_first\_negative)

1 : индекс последнего отрицательного элемента. (index\_last\_negative)

2 : Найти сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). (sum\_between\_negative)

3 : Найти сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). (sum\_before\_and\_after\_negative)

иначе необходимо вывести строку "Данные некорректны".

### **Основные теоретические положения.**

В данной лабораторной работе были использованы следующие конструкции языка C:

- Функции библиотеки `stdio.h`:
  - `printf()`—функция выводит на консоль значение аргумента
  - `scanf()`—функция ввода данных из консоли
- Функция библиотеки `stdlib.h`:
  - `abs()`—функция получения модуля числа
- Циклы:

- `while() {}`—каждая итерация проверяет, выполняется ли условие в круглых скобках, если оно верно, то выполняется код в фигурных скобках, а если неверно, то происходит выход из цикла
- `for(<переменная>; <условие>; <выражение_1>)`—код в теле цикла будет исполняться до тех пор, пока объявленная в цикле переменная будет удовлетворять условию цикла, выражение\_1 каким-либо способом меняет значение этой переменной
- Операторы:
  - `if() {} ... else {}`—если выполняется условия, указанное в круглых скобках, то выполняется код в фигурных скобках после `if`, иначе—в фигурных скобках после `else` (`else` является обязательной частью конструкции)
  - `switch(<переменная>){case x:... break; ... default:...break;}`—от значения переменной в круглых скобках зависит, какой кейс будет выполняться (например, если переменная имеет значение `x`—выполнится `case x`). Если же не будет кейса с таким значением, то выполнится код из блока `default`
- Функции:
  - `<тип_функции> имя_функции(<аргумент_1>, ... , <argument_n>) {}`—при вызове данной функции в главной(`main`) функции выполняется код в фигурных скобках, а затем возвращает значение оператором `return` (если тип функции не `void`)

### **Выполнение работы.**

В начале программы необходимо подключить следующие библиотеки:

- `stdio.h`—используется для подключения ввода-вывода (`printf()`, `scanf()`)
- `stdlib.h`—используются для доступа к функции `abs()`, которая позволяет получить модуль числа

Затем объявим переменные:

массив `mass[100]` — будет хранить входной массив целых чисел

`c` — переменная типа `char`, которая будет хранить символ, разделяющий элементы массива

Для того чтобы приступить к решению задачи, необходимо считать данные. Это осуществляется с помощью функции `scanf()`. В начале считывается переменная `kode` типа `int`, определяющая, какую из подзадач необходимо решить. Далее в цикле `while` осуществляется считывание по две переменные за цикл: `mass[i]` и `c`. Цикл `while` будет выполняться до тех пор, пока переменная `c` не равна символу новой строки (`'\n'`). При этом переменная `i`, отвечающая за обращение к элементам массива, после выхода из цикла будет иметь значение количества элементов массива.

Далее в зависимости от значения `kode` будет вызываться одна из функций `index_first_negative()`, `index_last_negative()`, `sum_between_negative()`, `sum_before_and_after_negative()` с помощью оператора `switch`.

1. Первую подзадачу решает функция `index_first_negative()`. Она получает на вход массив `fmass` и число `max` типа `int` (количество элементов массива, поступившего на вход программе) и возвращает значение типа `int`. В цикле `while` значение переменной `i` (тип `int`) увеличивается на единицу с каждой новой итерацией. Цикл прерывается когда значение `mass[k]` становится меньше 0. Функция возвращает номер первого отрицательного элемента.
2. Вторую подзадачу решает функция `index_last_negative()`. Она получает на вход массив `fmass` и число `max` типа `int` (количество элементов массива, поступившего на вход программе) и возвращает значение типа `int`. В цикле `while` значение переменной `i` (тип `int`) уменьшается на единицу, начиная с номера последнего элемента массива (`max`), с каждой новой итерацией. Цикл прерывается когда значение `mass[k]` становится меньше 0. Функция возвращает номер последнего отрицательного элемента.
3. Третью подзадачу решает функция `sum_between_negative()`. Она получает на вход массив `fmass` и число `max` типа `int` (количество элементов массива,

поступившего на вход программе) и возвращает значение типа `int`. В функции объявляются переменные типа `int`:

- `k` — счётчик для перебора элементов массива
- `start` — первый элемент суммы
- `stop` — последний элемент суммы
- `summ=0` — значение суммы

Переменные `start` и `stop` принимают значения функций `index_first_negative(fmass, max)` и `index_last_negative(fmass, max)` соответственно. В цикле `for` складываются элементы от `start` включительно до `stop` не включительно. Функция возвращает эту сумму.

4. Четвёртую подзадачу решает функция `sum_before_and_after_negative()`. Она получает на вход массив `fmass` и число `max` типа `int` (количество элементов массива, поступившего на вход программе) и возвращает значение типа `int`. В функции объявляются переменные типа `int`:

- `k` — счётчик для перебора элементов массива
- `start` — первый элемент суммы последних чисел
- `stop` — последний элемент суммы первых чисел
- `summ=0` — значение суммы

Переменные `start` и `stop` принимают значения функций `i` `index_last_negative(fmass, max)` и `index_first_negative(fmass, max)` соответственно. В цикле `for` складываются элементы от 0 элемента массива до `stop` не включительно и от `start` включительно до последнего элемента массива (`max`). Функция возвращает эту сумму.

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 1 2 3 4 -5 2 3 4 -5 4 3 2 -7 -8 5 8 9	4	Программа работает верно
2.	1 1 2 3 4 -5 2 3 4 -5 4 3 2 -7 -8 5 8 9	13	Программа работает верно
3.	2 1 2 3 4 -5 2 3 4 -5 4 3 2 -7 -8 5 8 9	35	Программа работает верно
4.	3 1 2 3 4 -5 2 3 4 -5 4 3 2 -7 -8 5 8 9	40	Программа работает верно
5.	10 1 2 3 4 -5 2 3 4 -5 4 3 2 - 7 -8 5 8 9	Данные некорректны	Программа работает верно

### **Выводы.**

Были изучены основные управляющие конструкции C: условия, циклы, оператор switch.

Была написана программа, считывающая данные с помощью функции `scanf()`, и выводящей результат с помощью `printf()`. Далее программа вызывает одну из четырёх функций, в зависимости от значения переменной `kode`. Это происходило с помощью оператора `switch`. Если значение `kode = 0`, то вызывалась функция `index_first_negative`. Если значение `kode = 1`, то вызывалась функция `index_last_negative`. Если значение `kode = 2`, то вызывалась функция `sum_between_negative`. Если значение `kode = 3`, то вызывалась функция `sum_before_and_after_negative`. После этого выполнялась соответствующая подзадача.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>

int index_first_negative(int fmass[], int max)
{
    int k=0;
    while (fmass[k]>=0)
        k++;
    return k;
}
int index_last_negative(int fmass[], int max)
{
    int k = max;
    while (fmass[k]>=0)
        k--;
    return k;
}
int sum_between_negative(int fmass[], int max)
{
    int k, start, stop, summ=0;
    start = index_first_negative(fmass, max);
    stop = index_last_negative(fmass, max);
    for (k = start; k<stop; k++)
        summ += abs(fmass[k]);
    return summ;
}
int sum_before_and_after_negative(int fmass[], int max)
{
    int k, start, stop, summ=0;
    stop = index_first_negative(fmass, max);
    start = index_last_negative(fmass, max);
    for (k = 0; k<stop; k++)
        summ += abs(fmass[k]);
    for (k=start; k<=max; k++)
        summ += abs(fmass[k]);
    return summ;
}
int main()
{
    int mass[100] = { 0 };
    int kode, i=0, result;
    scanf ("%d", &kode);
    char c = ' ';
    while (c != '\n')
    {
        scanf("%d%c", &mass[i], &c);
        i++;
    }
    switch (kode)
    {
```

```

        case 0:
            result = index_first_negative(mass, i);
            break;
        case 1:
            result = index_last_negative(mass, i);
            break;
        case 2:
            result = sum_between_negative(mass, i);
            break;
        case 3:
            result = sum_before_and_after_negative(mass, i);
            break;
        default:
            printf ("%s", "Данные некорректны");
            break;
    }
    if (kode >=0 && kode<=3)
        printf ("%d", result);
    return 0;
}

```