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: Условия, циклы, оператор switch

Студентка гр. 0382 _____ Михайлова О.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Научиться работать с массивами, циклами for и while, условным оператором if, оператором множественного выбора switch и выделять подзадачи программы в отдельные функции.

Задание.

Вариант 6.

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.

Функция нахождения модуля числа находится в заголовочном файле stdlib.h стандартной библиотеки языка Си.

Оператор множественного выбора switch:

Switch (<выражение>)

{ case <константное выражение 1>: <операторы 1>

…

case <константное выражение N>:<операторы N>

[default: <операторы>]

}

Выполняет поочередное сравнение выражения со списком константных выражений. При совпадении, выполнение программы начинается с соответствующего оператора. В случае, если совпадений не было, выполняется необязательная ветка **default**. Чтобы этого операторы после первого совпадения не выполнялись далее один за другим, следует использовать оператор **break.**

Условный оператор if:

If (<выражение>) <оператор 1> [else <оператор 2>]

Если выражение интерпретируется как истина, то оператор1 выполняется. Может иметь необязательную ветку **else**, путь выполнения программы пойдет в случае, если выражение ложно.

Цикл с предусловием while:

while (<выражение>) <оператор>

На каждой итерации цикла происходит вычисление выражения и если оно истинно, то выполняется тело цикла.

Цикл со счетчиком:

for ([<начальное выражение>]; [<условное выражение>]; [<выражение приращения>]) <оператор>

Условием продолжения цикла является некоторое выражение, в цикле со счетчиком есть еще 2 блока — начальное выражение, выполняемое один раз перед первым началом цикла и выражение приращения, выполняемое после каждой итерации цикла. Любая из трех частей оператора **for** может быть опущена.

Выполнение работы.

Переменные:

#define max_size 100 – макрос, обозначающий максимальный размер массива;

Int arr[max_size] – массив;

Int arr_size – переменная, в которую записывается размер массива;

Char sym – символ-разделитель (пробел);

Int command – переменная для считывания значения;

Int res0 – переменная, в которую записывается результат при вводе значения 0;

Int res1 – переменная, в которую записывается результат при вводе значения 1;

Int i – счетчик в цикле;

Int sum – переменная, в которую записывается сумма элементов массива;

Int first – переменная, в которую записывается индекс первого отрицательного элемента массива;

Int last – переменная, в которую записывается индекс последнего отрицательного элемента массива.

Функции:

Функция int index_first_negative (int arr[N], int arr_size) принимает на вход массив arr размера arr_size, находит и возвращает индекс первого отрицательного элемента массива.

Функция int index_last_negative (int arr[N], int arr_size) принимает на вход массив arr размера arr_size, находит и возвращает индекс последнего отрицательного элемента массива.

Функция int sum_between_negative (int arr[N], int arr_size) принимает на вход массив arr размера arr_size. Переменным first и last присваиваются значения индексов первого и последнего отрицательных элементов массива соответственно. Далее функция считает и выводит сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент).

Функция int sum_before_and_after_negative (int arr[N], int arr_size) принимает на вход массив arr размера arr_size. Переменным first и last присваиваются значения индексов первого и последнего отрицательных элементов массива соответственно. Далее функция сначала считает сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент), затем прибавляет сумму модулей элементов массива, расположенных после последнего отрицательного элемента (включая элемент) и выводит общую сумму.

В функции main на вход поступает сначала значение, которое обозначает команду, выбранную пользователем, затем массив. Оператор множественного выбора switch вызывает нужную команду и выводит результат. Если поступает значение, не указанное в операторе switch, то выводится сообщение «Данные некорректны». При корректной работе программа возвращает 0.

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 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	3	Получили индекс первого отрицательного элемента массива.
2.	2 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11	226	Получили сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент).
3.	1 4 -5 76 34 2 6 1 -8 -2 3 43 17 -13 4	12	Получили индекс последнего отрицательного элемента массива.
4.	4 4 -5 76 34 2 6 1 -8 -2	Данные	В операторе switch не

	3 43 17 -13 4	некорректны	указан случай, когда значение равно 4, поэтому получили «Данные некорректны».
5.	3 5 -6 7 24 35 9 -8 4 -1 -12 4 -5 6	16	Получили сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент).

Выводы.

Были изучены основные управляющие конструкции языка: циклы for и while, условный оператор if, оператор множественного выбора switch.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для выбора команды используется оператор switch. Для обработки команд в функциях используются условные операторы if и while, а также цикл for.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: pr_lb_1

```
#include <stdio.h>
#include <stdlib.h>
#define N 100

int index_first_negative (int arr[N], int arr_size){
    for (int i=0; i<arr_size; i++){
        if (arr[i]<0){
            return i;
        }
    }
}

int index_last_negative (int arr[N], int arr_size){
    for (int i=(arr_size-1); i>=0; i--){
        if (arr[i]<0){
            return i;
        }
    }
}

int sum_between_negative (int arr[N], int arr_size){
    int sum = 0;
    int first, last, i;
    first = index_first_negative (arr, arr_size);
    last = index_last_negative (arr, arr_size);

    for (i=first; i<last; i++){
        sum=sum+abs(arr[i]);
    }

    printf ("%d", sum);
}

int sum_before_and_after_negative (int arr[N], int arr_size){
    int sum = 0;
    int first, last, i;
    first = index_first_negative (arr, arr_size);
    last = index_last_negative (arr, arr_size);

    for (i=0; i<first; i++){
        sum=sum+abs(arr[i]);
    }

    for (i=last; i<arr_size; i++){
        sum=sum+abs(arr[i]);
    }

    printf ("%d", sum);

}

int main(){
    int arr[N];
    int arr_size = 0;
```

```
    char sym = ' ';
    int res0, res1;

    int x;
    scanf ("%d", &x);

    while ((arr_size<N) && (sym == ' ')){
        scanf ("%d%c", &arr[arr_size++], &sym);
    }

    switch (x){
        case 0:
         res0=index_first_negative (arr, arr_size);
         printf ("%d", res0);
         break;
        case 1:
         res1=index_last_negative (arr, arr_size);
         printf ("%d", res1);
         break;
        case 2:
         sum_between_negative (arr, arr_size);
         break;
        case 3:
         sum_before_and_after_negative (arr, arr_size);
         break;
        default:
         printf ("Данные некорректны");
    }
    return 0;
}
```