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»

Студент гр. 9383 _____ Ноздрин В.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Научиться работать с условиями, циклами и оператором switch.

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".

Выполнение работы.

Переменные:

temp - переменная типа char, для считывания пробельных символов

command - переменная типа char, отвечающая за выбор функции

len - переменная типа int, являющаяся счетчиком при считывании массива и потом хранит его длинну (кол-во введенных элементов)

Ввод реализован с помощью функции **scanf()** и цикла **while**.

Для выбора функции из уловия используется оператор **switch**.

Функции:

index_first_negative () **index_last_negative ()**

sum_between_negative () **sum_before_and_after_negative ()**

Все функции реализованы одинаково:

Как аргументы функция получает массив и количество значащий элементов, и с помощью цикла **for** выболняется проход по массиву с проверкой некоторых условий с помощью условного оператора **if**. Функции возвращают значение типа **int**, отвечающее либо за индекс, либо за значение некоторой суммы.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица – Результаты тестирования

№ п/п	Входные данные	Выходные данные
	1 5 -8 9 4 6	1
	98 654 321 654 321	Данные некорректны
	3 5 8 -9 6 5 4 -1 5 -9 5	27

5 Вывод.

Были изучены основные управляющие конструкции языка **C**, такие как условный оператор **if**, циклы **for** и **while** и оператор **switch**.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для обработки команд пользователя использовались условные операторы **if-else** и циклы **for**.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define max_len 100

int index_first_negative (int array[], int len);
int index_last_negative (int array[], int len);
int sum_between_negative (int array[], int len);
int sum_before_and_after_negative (int array[], int len);

int main(){
   char temp, command;
   scanf("%c%c", &command, &temp);
   if (temp != ' ') {
      puts("Данные некорректны\n");
      return 0;
   }

   int len = 0;
   int values[max_len];
   do {
      scanf("%d%c", &values[len], &temp);
      len++;
   } while(temp != '\n');
   switch(command) {
      case '0':
         printf("%d\n",index_first_negative(values, len));
         break;
      case '1':
         printf("%d\n",index_last_negative(values, len));
         break;
      case '2':
         printf("%d\n",sum_between_negative(values, len));
         break;
      case '3':
```

```
            printf("%d\n",sum_before_and_after_negative(values, len));
            break;
        default:
            puts("Данные некорректны");
            break;
    }

    return 0;
}



int index_first_negative (int array[], int len){
    for (int i = 0; i < len; i++){
        if (array[i] < 0){
            return i;
        }
    }
    return -1;
}



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



int sum_between_negative (int array[], int len){
    int sum = 0;
    int id_last_negative = index_last_negative(array, len);
    for (int i = index_first_negative(array, len); i < id_last_negative; i++){
        sum += abs(array[i]);
    }
    return sum;
}
```

```
int sum_before_and_after_negative (int array[], int len){
    int sum = 0;
    for (int i = 0; i < index_first_negative(array, len); i++){
        sum += abs(array[i]);
    }
    for (int i = index_last_negative(array, len); i < len; i++){
        sum += abs(array[i]);
    }
    return sum;
}
```