**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. 1304 _____ Стародубов М.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

**2021**

**Цель работы.**

Научиться использовать условия, оператор switch, циклы, операторы ввода и вывода информации, а также функции в языке программирования Си.

**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.**

Рассмотрим синтаксис условных операторов в языке Си.

Условный оператор *if*:

```
if (<условие>){
        <Инструкции, выполняющиеся в случае истиности условия>
} else{
        <Инструкции, выполняющиеся в случае ложности условия>
}
```

Условный оператор *switch*:

```
switch (<Переменная>){
          case <Значение_1>:
                    <Инструкции>
          case <Значение _2>:
                    <Инструкции>
          …
          case <Значение_N>:
                    <Инструкции>
          default:
                    <Инструкции>
}
```

В случае, если значение переменной не равно ни одному из значений *case*, то выполняются инструкции записанные в ветке *default*. В случае, если значение переменной равно какому-либо значению *case*, то выполняются соответствующие инструкции, а также инструкции, соответствиющие последующим значениям *case*, а также ветка *default*, остановить выполнение последующих инструкций можно с помощью оператора *break*.

Рассмотрим циклы, существующие в языке Си.

Цикл с предусловием:

```
while (<Условие>){
          <Инструкции>
}
```

Инструкции выполняются, пока условие истинно, при этом инструкции могут быть и не выполены ни разу, если непосредственно перед первой итерацией условие было ложным.

Цикл с постусловием:

```
do {
          <Инструкции>
} while (<Условие>);
```

Цикл выполняется, пока условие истинно, при этом инструкции в теле цикла будут выполнены как минимум один раз.

Цикл *for*:

```
for (<Действие до начала цикла>; <Условие продолжения цикла>; <Действие в конце каждой итерации цикла>){
          <Инструкции>
}
```

Перед первой итерацией цикла *for* производиться действие до начала

цикла, а также проверяется условие продолжения, в случае, если условие истинно, выполняются инструкции в теле цикла, а после выполнения инструкций выполняется действие в конце каждой итерации цикла, далее вновь проверяется условие продолжения.

**Выполнение работы.**

Начнем рассмотрение программы с функции *main*. В начале мы объявляем переменную character и указатель на данную переменную *character_pointer*. Здесь и далее во всех функциях переменная *character* будет использоваться для получения и обработки символов, получаемых на вход программе, указатель *cahacter_pointer* позже позволит нам отслеживать, был ли введен символ переноса строки, означающий конец ввода входных данных. В переменную *function_index* мы, используя функцию *get_number* (описание работы функции будет представлено позже), записываем первую цифру из входных данных, с помощью данной переменной мы будем определять, какую функцию использовать для обработки последующих данных. Далее создаем целочисленный массив *numbers*, в который с помощью цикла и функции *get_number* мы запишем поступающий далее на вход массив целых чисел, а с помощью переменной *array_size* мы сможем опретследить его размер. В данном цикле мы используем возвращаемый через указатель функцией *get_number* символ, которым оканчивается ввод каждой цифры. Если данный символ – это символ переноса строки, то ввод данных завершен. Далее с помощью условного оператора *switch* и переменной function_index программа определяет, использование какой функции запрашивает пользователь. В случае, если введенный индексс не соответствует ни одному из предложенных, то программа выведет на экран сообщение „Данные некорректны“ и завершит работу. В остальных случаях будет вызвана соответствующая индексу функция, а после ее завершения программа вываедет на экран полученное от функции значение и

завершит работу.

Сначала рассмотрим работу функции *get_number*, которая возвращает следующее введенное число. Данная функция принимает в качестве аргумента указатель на переменную *character*, чтобы позже вернуть в функцию *main* символ, которым окончился ввод каждого числа. В переменной *result* будет храниться считанное число, которое позже пойдет на вывод функции. С помощью переменной *result_sign* мы определим знак возвращаемого числа. Рассмотрим цикл *for*. С каждой новой итерацией цикла мы получаем новый символ из входных данных, пока не получим символ перевода строки или пробел (данные символы и означают окончание ввода числа). Если на вход поступил символ „-“, значит на вход поступает отрицательное число, и поэтому переменная *result_sign* меняет свое значение на „-1“. Считывание самого числа происходит по следующей схеме: с получением каждой новой цифры числа старое значение *result* умножается на 10 и в конец записывается полученная цифра. Расчет того, какая цифра идет на вход, проходит следующим образом: из индекса полученного символа в таблице ASCII вычитается индекс символа „0“. Полученное таким образом значение и является введенной цифрой. После окончания цикла мы возвращаем в переменную *character* функции *main* значение последнего полученного символа, а в качестве возвращаемого значения передаем произведение *result* и *result_sign*. Данная функция реализована исключительно для избежания дублирования кода, а также чтобы сделать получение входных данных более удобным.

Перейдем к описанию каждой вызываемой в операторе *switch* функции. Стоит заранее отметить, что все эти функции принимают на вход два аргумента: массив целых чисеел и размер данного массива.

Первая функция,  для вызова которой *function_index* должен быть равен 0 – *index_first_negative*. Тело данной функции состоит из одного цикла, который

перебирает начиная с 0 индексы полученного на вход массива, и в случае, если соответствующий данному индексу элемент массива - это отрицательное число, то функция возвращает этот индекс.

Следующая функция – *index_last_negative*. *function_index* для вызова этой функции должен быть равен 1. Тело данной функции также, как и в функции *index_first_negative*, состоит из одного цикла, но с одним отличием – функция перебирает индексы начиная с последнего, в порядке убывания. В случае, если соответствующий данному индексу элемент массива отрицательный, то программа возвращает данный индекс.

Далее рассмотрим функцию *sum_between_negative*. Для ее вызова *function_index* должен быть равен 2. В переменную *sum* будет записана сумма чисел, расположенных между первым и последним отрицательными числами в массиве. В цикле мы перебираем индексы элементов, начиная с индекса первого отрицательного элемента включительно, заканчивая индексом последнего отрицательного элемента невключительно. Данные индексы мы получаем используя описаенные выше функции. В теле цикла в переменную *sum* мы суммируем модули всех элементов, соответствующих перебираемым индексам. Функция возвращает полученную сумму.

Последняя функция – *sum_before_and_after_negative*. Для ее вызова значение *function_index* должно быть равно 3. В переменной *sum* также будет записана сумма элементов массива. В теле функции присутствует два цикла. Первый цикл с помощью функции *index_first_negative* получает индекс первого отрицательного элемента, и перебирает все индексы, начиная с нуля, до индекса первого отрицательного элемента невключительно. В теле цикла в переменной *sum* суммируются модули элементов массива, соответствующие данным индексам. Во втором цикле с помощью функции *index_last_negative* мы получаем индекс последнего отрицательного элемента, и начиная с него

перебираем индексы до конца массива. В теле цикла мы также суммируем в переменную *sum* модули элементов массива, соответствующих перебираемым в этом цикле индексам. Функция возвращает записанную в переменной *sum* сумму.

**Выводы.**

В ходе выполнения лабораторной работы были изучены основные управляющие конструкции языка программирования Си, такие как условия, циклы, оператор switch, функции.

Разработана программа, считывающая с клавиатуры входные данные и команды пользователя, и в зависимости от выбранной команды вызывающая одну из четырех функций:

0 : функция возвращает индекс первого отрицательного элемента.

1 : функция возвращает индекс последнего отрицательного элемента.

2 : функция возвращает сумму модулей элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент).

3 : функция возвращает сумму модулей элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент).

**Тестирование.**

Данные тестирования представлены в таблице 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 | Результат корректен |
| 2. | 1 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11 | 20 | Результат корректен |
| 3. | 2 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11 | 226 | Результат корректен |
| 4. | 3 1 16 2 -18 -22 15 -3 13 0 -6 1 9 24 1 -18 15 28 20 -17 16 -11 | 30 | Результат корректен |
| 5. | 0 17 9 7 20 21 -28 20 12 -24 15 -6 5 11 4 27 | 5 | Результат корректен |
| 6. | 1 -22 23 22 -28 4 6 18 21 25 19 19 -28 9 26 0 -15 11 25 12 | 15 | Результат корректен |
| 7. | 2 26 22 27 26 11 -18 25 4 2 1 22 -9 19 26 19 15 -27 -4 0 29 6 21 2 | 187 | Результат корректен |
| 8. | 3 22 -19 11 24 7 24 -21 -17 9 29 11 15 25 22 17 7 26 24 10 2 14 | 250 | Результат корректен |

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define DEFAULT_SIZE 100

int get_number(int *character_pointer){
	int cheracter;
	int result = 0;
	int result_sign = 1;
	for (cheracter = getchar(); (cheracter != ' ') && (cheracter != '\n');
cheracter = getchar()){
		if (cheracter != '-'){
			result *= 10;
			result += cheracter - '0';
		} else result_sign = -1;
	}
	*character_pointer = cheracter;
	return result*result_sign;
}

int index_first_negative(int numbers[], int array_size){
	for (int i = 0; i < array_size; i++){
		if (numbers[i] < 0){
			return i;
		}
	}
}

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

int sum_between_negative(int numbers[], int array_size){
	int sum = 0;
	for (int i = index_first_negative(numbers, array_size); i <
index_last_negative(numbers, array_size); i++){
		sum += abs(numbers[i]);
	}
	return sum;
}

int sum_before_and_after_negative(int numbers[], int array_size){
	int sum = 0;
	for (int i = 0; i < index_first_negative(numbers, array_size); i++){
		sum += abs(numbers[i]);
```

```
        }
        for (int i = index_last_negative(numbers, array_size); i < array_size;
i++){
                sum += abs(numbers[i]);
        }
        return sum;
}

int main(){
        int cheracter;
        int *character_pointer;
        character_pointer = &cheracter;

        int function_index = get_number(character_pointer);

        int numbers[DEFAULT_SIZE];
        int array_size = 0;
        for (int i = 0; cheracter != '\n'; i++){
                numbers[i] = get_number(character_pointer);
                array_size++;
        }

        switch (function_index){
                case 0:
                        printf("%d\n", index_first_negative(numbers,
array_size));
                        break;
                case 1:
                        printf("%d\n", index_last_negative(numbers,
array_size));
                        break;
                case 2:
                        printf("%d\n", sum_between_negative(numbers,
array_size));
                        break;
                case 3:
                        printf("%d\n",
sum_before_and_after_negative(numbers, array_size));
                        break;
                default:
                    printf("Данные некорректны");
        }
}
```