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. 0381 _____ Самойлов З. А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Научиться разделять задачу на подзадачи и соответствующие функции, обрабатывать и работать с массивами, пользоваться оператором ветвления switch.

Задание.

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 нулевого элемента. (index_first_zero)

1 : индекс последнего нулевого элемента. (index_last_zero)

2 : Найти сумму модулей элементов массива, расположенных от первого нулевого элемента и до последнего. (sum_between)

3 : Найти сумму модулей элементов массива, расположенных до первого нулевого элемента и после последнего. (sum_before_and_after)

Иначе необходимо вывести строку "Данные некорректны".

Выполнение работы.

Описание переменных:

numbers[N] — массив размера N для записи передаваемых чисел и последующей работы с ними.

N — константа, определяющая максимальный размер массива

par — параметр, передаваемый в начале ввода;

index_first_zero — индекс первого нуля в строке (отсчет индексов начинается с нулевого элемента);

index_last_zero — индекс последнего нуля в строке;

n — переменная для проверки ввода пробела в строке и ограничения посредством символа перевода строки.

lenght — количество чисел, записанных в массив.

Описание функций:

1. **main()** — глобальная функция, являющаяся входной точкой программы. В данной функии производится считывание вводимых данных и, при необходимости, нахождение модуля. Затем данные используются в конструкции switch. В зависимости от значения переменной par, выполняются следующие действия:

При par = 0, вызывается функция index_f_zero(int size, int arr[]). При правильно введенных данных её возвращаемое значение сохраняется в переменную index_first_zero и выводится на экран. Иначе выводится «Данные некорректны».

При par = 1, ызывается функция index_l_zero(int size, int arr[]). При правильно введенных данных её возвращаемое значение сохраняется в переменную index_last_zero и выводится на экран. Иначе выводится «Данные некорректны».

При par = 2, поочередно вызываются функции index_f_zero(int size, int arr[]) и index_l_zero(int size, int arr[]), возвращаемые значения которых записываются в переменные index_first_zero и index_last_zero соответственно. При корректности полученных данных вызывается функция sum_between(arr[], first, last), возвращаемое значение которой выводится на экран. Иначе выводится «Данные некорректны».

При par = 3, поочередно вызываются функции index_f_zero(int size, int arr[]) и index_l_zero(int size, int arr[]), возвращаемые значения которых записываются в переменные index_first_zero и index_last_zero соответственно. При корректности полученных данных вызывается функция sum_before_and_after(arr[], size, first, last), возвращаемое значение которой выводится на экран. Иначе выводится «Данные некорректны».

При любых других par выводится «Данные некорректны».

2. **index_f_zero(int size, int arr[])** — функция, принимающая размер массива в size и указатель на сам массив. Возвращаемое значение равно индексу *первого* элемента массива, равного нулю.

3. **index_l_zero(int size, int arr[])** — функция, подобная index_f_zero, но возвращающая индекс *последнего* элемента массива, равного нулю.

4. **sum_between(int arr[], int first, int last)** — функция принимает указатель на массив и индексы первого и последнего элементов массива со значениями, равными нулю. Возвращаемое значение является модулем суммы элементов массива, находящихся между первым и последним «нулевыми» элементами массива.

5. **sum_before_and_after(int arr[], int size, int first, int last) —** функция принимает указатель на массив, его размер и индексы первого и последнего элементов массива со значениями, равными нулю. Возвращаемое значение равно модулю суммы элементов массива, находящихся до первого «нулевого» и после последнего «нулевого» элемента массива.

Тестирование.

Результаты тестирования представлены в табл. 1.

№ п/п	Входные данные	Выходные данные	Комментарий
1.	0 3 2 4 5 2 6 3 4 3 1 0 23 -41 4	10	Верно
2.	5 -23 5 13 0 23 65 0 5 4 8 9 620	Данные некорректны	Верно
3.	2 3 -42 0 -152 54 0 -4	206	Верно
4.	1 0 3 4 -2351 15645 3546 0 2246 354 354	6	Верно
5.	3 424 -15 0 1421 515 -215 0 123 634	1196	Верно

Вывод.

В ходе данной лабораторной работы было изучено раздление задачи на подзадачи, обработка и работа с массивами, а также работа с оператором ветвления switch. Результатом работы стала программа, решающая поставленную задачу посредством использования инструментов языка С.

Приложение

Исходный код программы на языке С

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

int index_f_zero(int size, int arr[]); // Поиск первого нуля в массиве чисел
int index_l_zero(int size, int arr[]); // Поиск последнего нуля в массиве чисел
int sum_between(int arr[], int first, int last); // Сумма чисел между первым и последним нулями
int sum_before_and_after(int arr[], int size, int first, int last); // Сумма чисел до первого и после последнего нолей

int main(){
	int par; // Передаваемое значение - параметр
	int index_first_zero, index_last_zero; // Индексы первого и последнего нулей
	char n; // Проверка пробела и enter
	int lenght = 0; // Количество чисел
	int numbers[N]; // Массив передаваемых чисел

	scanf("%d%c", &par, &n); // Ввод параметра
	while (n != '\n' &&  lenght < N){ // Ввод чисел в массив
		scanf("%d%c", &numbers[lenght], &n);
		if (numbers[lenght] < 0) // Модуль отрицательного числа
			numbers[lenght] *= -1;
		lenght++;
	};
	switch (par){
		case 0:
```

```
                index_first_zero = index_f_zero(lenght, numbers);
                if (index_first_zero == lenght)
                        printf("Данные некорректны");
                else printf ("%d", index_first_zero);
                break;
        case 1:
                index_last_zero = index_l_zero(lenght, numbers);
                if (index_last_zero == lenght)
                        printf("Данные некорректны");
                else printf ("%d", index_last_zero);
                break;
        case 2: index_first_zero = index_f_zero(lenght, numbers);
                index_last_zero = index_l_zero(lenght, numbers);
                if (index_first_zero == lenght || index_last_zero == lenght ||
index_first_zero == index_last_zero)
                        printf("Данные некорректны");
                else printf("%d", sum_between(numbers, index_first_zero,
index_last_zero));
                break;
        case 3: index_first_zero = index_f_zero(lenght, numbers);
                index_last_zero = index_l_zero(lenght, numbers);
                if (index_first_zero == lenght && index_last_zero == lenght)
                        printf("Данные некорректны");
                else printf("%d", sum_before_and_after(numbers, lenght,
index_first_zero, index_last_zero));
                break;
        default:
                printf("Данные некорректны");
                break;
    };
```

```
        return 0;
}

int index_f_zero(int size, int arr[]){
        int index = 0;
        for (index; index < size; index++){
                if (arr[index] == 0){
                        return index;
                };
        }
        return size;
}
int index_l_zero(int size, int arr[]){
        int index = size - 1;
        for (index; index >= 0; index--){
                if (arr[index] == 0){
                        return index;
                };
        };
        return size;
}
int sum_between(int arr[], int first, int last){
        int sum = 0;
        for (int index = first+1; index < last; index++)
                sum += arr[index];
        return sum;
}
int sum_before_and_after(int arr[], int size, int first, int last){
        int sum = 0;
        if (first != size)
```

```
            for (int index = 0; index < first; index++)
                sum += arr[index];
        if (last != size)
            for (int index = last + 1; index < size; index++)
                sum += arr[index];
        return sum;
}
```