

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. 0382

Тихонов С.В.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Научиться работать с базовыми конструкциями в Си

Задание.

Вариант-4.

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 чётного элемента. (index_first_even)

1 : индекс последнего нечётного элемента. (index_last_odd)

2 : Найти сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний. (sum_between_even_odd)

3 : Найти сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент). (sum_before_even_and_after_odd)

иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения.

В данной лабораторной работе были использованы функции ввода и вывода scanf() и printf() из библиотеки stdio.h. Также была использована функция abs() из библиотеки stdlib.h для нахождения модуля числа. Кроме того я использовал такие операторы как if(){}, switch (){} и циклы while(){} и for(){}.

Выполнение работы.

В моей программе использует 5 функций. Первые 4 функции на вход получают два аргумента, массив и количество в нём

1. index_first_even() при помощи for(){} и if(){} ищет первый четный элемент массива. При помощи цикла for(){} мы перебираем массив с начала и ищем первый четный элемент. Функция возвращает его номер в массиве.
2. index_last_odd() используя то же for(){} и if(){} ищет последний нечетный элемент массива. При помощи цикла for(){} мы перебираем массив с конца

и ищем последний нечетный элемент. Функция возвращает его номер в массиве.

3. `sum_between_even_odd()` с помощью функций `index_first_even()` и `index_last_odd()` ищет сумму модулей элементов массива, расположенных от первого чётного элемента и до последнего нечётного, включая первый и не включая последний.
4. `sum_before_even_and_after_odd()` с помощью функций `index_first_even()` и `index_last_odd()` ищет сумму модулей элементов массива, расположенных до первого чётного элемента (не включая элемент) и после последнего нечётного (включая элемент).
5. `main()`.

С начала я объявляю переменные:

`int a[]`- массив в котором будут храниться значения ведёные пользователем.

`int a_size` – здесь будет храниться количество заполненных ячеек массива

`char sym` - изначально присваиваю «пробел», для того, чтобы при вводе массива в массив вводились только числа.

`int shto_delat` здесь будет храниться выбор опции пользователя, которая будет использована в операторе `switch`.

Пользователь вводит цифру, опцию которой он хочет выбрать. После этого мы начинаем заполнение массива: Если количество элементов массива не превышает максимальный и переменная `sum == « »`, тогда считывается сначала число массив, а потом символ, который должен являться пробелом.

Затем мы вызываем оператор `switch`:

Если пользователь ввел 0 — на экран выводится значение функции `index_first_even`.

Если пользователь ввел 1 -на экран выводится значение функции `index_last_odd`.

Если пользователь ввел 2 — на экран выводится значение функции `sum_between_even_odd`.

Если пользователь ввел 3 — на экран выводится значение функции `sum_before_even_and_after_odd`.

Если переменная `choice` имеет другое значение — на экран выводится сообщение «Данные некорректны».

Выполнение функции `main` заканчивается, возвращается 0.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 -8 -23 -30 -11 -28 15 -20 - 24 -27 5 -13 5 21 -5 16 30 - 12 15 -14 -28 -27 -11 -5 4 29 -5	0	<code>index_first_even</code>
2.	1 1 2 3 4 5 6 7	6	<code>index_last_odd</code>
3.	2 1 2 3 4 5 6 7	27	<code>sum_between_even_odd</code>
4.	3 1 2 3 4 5 6 7	8	<code>sum_before_even_and_aft er_odd</code>

Выводы.

Были изучены базовые конструкции в Си (условные операторы, и циклы) и использование отдельных функций для предотвращения дублирования кода.

Были изучены основные управляющие конструкции языка...

Разработана программа, выполняющая считывание с клавиатуры массива чисел и номера операции с помощью цикла `while` и оператора `scanf`, определяющая, какую операцию необходимо выполнить через оператора `switch`, выполняющая расчёт результата в отдельной функции, принимающей в качестве аргументов исходный массив и количество введённых чисел, и выводящая этот результат пользователю. При некорректном выборе опции на экран выводилась строка «Данные некорректны».

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1.c

```
#include <stdio.h>
#include <stdlib.h>
int index_first_even(int a[100],int a_size){
for (int x=0; x<a_size; x++){
    if(a[x]%2==0){
        return x;
    }
}
}
int index_last_odd (int a[100], int a_size){
for (int x=(a_size-1); x>=0; x--){
    if (a[x]%2!=0){
        return x;
    }
}
}
int sum_between_even_odd(int a[100], int a_size){
    int sum=0;
    int first, last, x;
    first = index_first_even(a, a_size);
    last=index_last_odd (a, a_size);
    for (x=first; x<last; x++){
        sum=sum+abs(a[x]);
    }
    printf ("%d\n", sum);
}
int sum_before_even_and_after_odd(int a[100], int a_size){
    int sum=0;
    int first, last, x;
    first=index_first_even(a, a_size);
    last=index_last_odd (a, a_size);
    for (x=0; x<first; x++){
        sum=sum+abs(a[x]);
    }
    for (x=last; x<a_size; x++){
        sum=sum+abs(a[x]);
    }
    printf ("%d\n", sum);
}
int main (){
    int x=0;
    int a[100];
    int a_size = 0;
    char sym = ' ';
    int zero,one;
    int shto_delat = getchar() - '0';
    while(a_size < 100 && sym == ' '){scanf("%d%c",&a[a_size++], &sym);}
    switch (shto_delat){
        case 0:
            zero=index_first_even(a, a_size);
            printf ("%d\n", zero);
            break;
```

```
    case 1:
        one=index_last_odd (a, a_size);
        printf ("%d\n", one);
        break;
    case 2:
        sum_between_even_odd(a, a_size);
        break;
    case 3:
        sum_before_even_and_after_odd(a, a_size);
        break;
    default:
        printf ("Данные некорректны");
}
return 0;
}
```