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. 1303		Чубан Д.В.
Преподаватель		Чайка К.В

Санкт-Петербург

2021

Цель работы.

Научиться использовать такие управляющие конструкции языка Си, как

условный оператор if-else, циклы for и while, оператор выбора switch.

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 массиве есть хотя бы один четный и нечетный элемент.

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)

Выполнение работы.

В начале подключается библиотека вывода stdio.h и заголовочный файл stdlib.h

В программе реализованы функции:

- *int get_first_ind(int ar[], int n)*
- *int get_last_ind(int ar[], int n)*
- *get_abssum_between(int ar[], int n)*
- *get_abssum_bef_aft(int ar[], int n)*

Рассмотрим каждую функцию:

Функция *int get_first_ind(int ar[], int n)* используется для нахождения индекса первого по счету четного числа в массиве. На вход функции подаются массив,

который вводится в основной программе, и его размер. С помощью цикла *for* мы начинаем проверять элементы массива на отсутсвие остатка при его делении на 2. Если остаток равен нулю, то функция возвращает значение счетчика *int i* в цикле *for,* который равен индексу текущего проверяемого элемента массива, прерывая цикл.

Функция *int get_last_ind(int ar[], int n)* используется для нахождения индекса последнего по счету нечетного числа в массиве. На вход функции подаются массив, который вводится в основной программе, и его размер. С помощью цикла *for* мы начинаем проверять элементы массива на наличие остатка при его делении на 2. Если остаток не равен нулю, то функция возвращает значение счетчика *int i* в цикле *for,* который равен индексу текущего проверяемого элемента массива, прерывая цикл.

Функция *get_abssum_between(int ar[], int n)* используется для нахождения суммы модулей элементов массива, стоящих после первого четного элемента и до последнего нечетного элемента (включая первый и не включая последний). Эта функция использует две предыдущих функции для нахождения индексов первого четного и последнего нечетного элементов, используя для этого внутренние переменные *int first* и *int last* (для первого чет. эл. и посл. нечет. эл. соответственно), после чего, используя цикл *for* накапливает сумму модулей элементов, используя *abs(ar[i]),* где *ar[i]* - элемент массива.

Функция *get_abssum_bef_aft(int ar[], int n)* используется для нахождения суммы модулей элементов массива, стоящих до первого четного элемента и после последнего нечетного элемента (не включая первый и включая последний). Эта функция использует две первых функции для нахождения индексов первого четного и последнего нечетного элементов, используя для этого внутренние переменные *int first* и *int last* (для первого чет. эл. и посл. нечет. эл. соответственно), после чего, используя цикл *for* накапливает сумму модулей элементов, используя *abs(ar[i]),* где *ar[i]* - элемент массива.

В теле основной функции *int main()* мы осуществляем ввод *int cs,* влияющей на дальнейшие действия программы. В функции существует переменная *int n*. Она используется в качестве счетчика в цикле *while*. Пока значение *n* меньше максимального размера массива, с клавиатуры считываются значения элементов массива и символы пробела (для этого используется переменная *char c*). Если встречается символ переноса строки, то выполнение цикла прекращается.

После ввода массива, используется оператор *switch(cs),* который в зависимости от значения введенного ранее *int cs* выполняет одну из 4 функций, описанных выше и выводит на экран их результат. В случае, если значение *int cs* не входит в допустимые для работы значения (0, 1, 2, 3), то программа выводит строку "Данные некорректны".

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	2 11 -4 -8 -7 -14	12	Ожидаемый результат - 12, результат верный
2	1 3 -13 -11 -10 -13	4	Ожидаемый результат - 4, результат верный
3	0 8 7 1 -10 -11	0	Ожидаемый результат - 0, результат верный
4	1 6 4 -9 4 2	2	Ожидаемый результат - 2, результат верный

Выводы.

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

Разработана программа, выполняющая считывание с клавиатуры исходных данных и заполняющая этими данными массив. В зависимости от первого введенного значения, программа печатает индекс первого четного элемента,

последнего нечетного элемента, сумму модулей элементов между первым четным и последним нечетным элементами и сумму модулей элементов до первого четного и после последнего нечетного элементов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1_finale.c

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

int get_first_ind(int ar[], int n){
    int res;
    for (int i = 0 ; i < n; i++){
        if(ar[i]%2 == 0){
           res = i;
              return res;
        }
    }

}

int get_last_ind(int ar[], int n){
    int res;
    for (int i = n-1; i >= 0; i--){
         if(ar[i]%2 != 0){
           res = i;
            return res;
        }
    }
}

int get_abssum_between(int ar[], int n){
  int first = get_first_ind(ar, n);
  int last = get_last_ind(ar, n);
  int abssum = 0;
  for (int i = first; i < last; i++){
            abssum += abs(ar[i]);
        }
        return abssum;
}

int get_abssum_bef_aft(int ar[], int n){
        int abssum = 0;
        int first = get_first_ind(ar, n);
        int last = get_last_ind(ar, n);
        for (int i = 0; i < first; i++){
            abssum +=  abs(ar[i]);
        }
```

```
        for (int i = last; i < n; i++){
            abssum += abs(ar[i]);
        }
        return abssum;
}

int main()
{
     int ar[100];
     int cs;
     int n = 0;
     char c;
     scanf("%d%c", &cs, &c);
     while(n<100){
            scanf("%d%c",&ar[n],&c);
            n++;
            if(c == '\n'){
                break;
            }
     }

    int res = 0;
    switch(cs){
case 0:
   printf("%d\n", get_first_ind(ar, n));

    break;

case 1:
   printf("%d\n", get_last_ind(ar,n));
   break;

case 2:

    res = get_abssum_between(ar, n);
    printf("%d\n", res);

    break;

case 3:
    res = get_abssum_bef_aft(ar, n);
    printf("%d\n", res);

    break;

default:
    printf("Данные некорректны");

    }
return 0;
}
```