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 Сергеев Д.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение базовых управляющих конструкций языки Си.

Задание.

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 работе была использована функция abs() из библиотеки stdlib.h для нахождения модуля числа. Также были использованы функции scanf() и printf() для ввода и вывода из библиотеки stdio.h. Кроме этого были использованы операторы if(){} else{}, for (){}, while(){} ,switch(){}

Выполнение работы.

Разработанный программный код см. в приложении А.

В функции *main{}* объявляется целочисленная переменная *k*, которой с помощью функции *scanf()* присваивается целочисленное значение. Далее объявляется целочисленный массив *arr* размером 100 и целочисленная переменная *arr_size* равная 0, которая показывает количество элементов в массиве. В следующей строчке объявляется символьная переменная *sym = ' '*. Далее в теле цикла *while (arr_size <100 && sym==' '){}* применяется функция *scanf()*, с помощью которой вводится целый элемент массива *arr[]* с индексом *arr_size++* и символ *sym*. Далее применяется оператор *switch(k){}*, который в зависимости от значения *k*, будет выполнять различные команды.

Если *k* равняется 0, то с помощью функции *printf()* печатается значение функции *index_first_even(arr, arr_size)*. Функция *index_first_even(int A[], int b)* получает на вход целочисленный массив *A* и целое числа *b*, затем в функции создаётся локальная целочисленная переменная *ind*, равная 0. Используя цикл *while ((abs(A[ind]%2==1) && (ind !=b))*, в теле которого *ind* увеличивается на 1 за итерацию, удаётся найти индекс первого чётного элемента(фукнция *abs()* используется, так как если *A[ind]* будет отрицательным, то в случае нечётности элемента значение *A[ind]%2* будет равно -1 и цикл завершится. Функция возвращает значение *ind*. Для выхода из оператора *switch* используется *break*.

Если *k* равняется 1, то с помощью функции *printf()* печатается значение функции *index_last_odd(arr, arr_size)*. Функция *index_last_odd(int A[], int b){}* получает на вход целочисленный массив *A* и целое числа *b*, затем *b* уменьшается на 1, так как он будет использоваться как индекс массива. Затем, используя цикл *while ((A[b]%2==0) && (b>=0)){}*, в теле которого *b* уменьшается на 1 за итерацию, удаётся найти индекс последнего нечётного элемента в массиве. Функция возвращает значение *b*. Для выхода из оператора *switch* используется *break*.

Если *k* равняется 2, то с помощью функции *printf()* печатается значение функции *sum_between_even_odd(arr, arr_size)*. Функция *sum_between_even_odd (int A[], int b){}* получает на вход целочисленный массив *A* и целое числа *b*. Объявляется четыре локальные целочисленные переменные: *f=index_first_even(A,b), l=index_last_odd(A,b), summ=0* и *i*, где *f* – индекс первого чётного элемента массива(находится с помощью ранее описанной функции), *l* – индекс последнего нечётного элемента массива(находится с помощью ранее описанной функции), *summ* – искомая сумма. Далее с помощью цикла *for(i=f;i<l;i++){}* с телом *summ=summ+abs(A[i])*, находится сумма членов массива от первого чётного(включая) до последнего нечетного(исключая). Функция возвращает значение *summ*. Для выхода из оператора switch используется *break*.

Если *k* равняется 3, то с помощью функции *printf()* печатается значение функции *sum_before_even_and_after_odd(arr, arr_size)*. Функция *sum_before_even_and_after_odd (int A[], int b){}* получает на вход целочисленный массив *A* и целое числа *b*. Объявляются две целочисленные локальные переменные *summ=0*, где *summ* – искомая сумма, и *i*. Далее с помощью цикла *for(i=0;i<b;i++){}* с телом *summ=summ+abs(A[i])*, находится сумма всех элементов массива. После чего *summ* присваивается значение *summ-sum_between_even_odd(A,b)*, в результате чего в переменной *summ* хранится сумма всех элементов массива до первого чётного элемента(исключая) и после последнего нечетного элемента массива(включая). Функция возвращает значение *summ*. Для выхода из оператора switch(){} используется *break*.

При значении *k*, отличном от 0,1,2 или 3, с помощью функции *printf*() печатается строка “Данные некорректны”. Для выхода из оператора *switch* (){}используется *break*.

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\n	0	Программа работает правильно.
2.	1 -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\n	25	Программа работает правильно.
3.	2 -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\n	426	Программа работает правильно.
4	3 -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\n	5	Программа работает правильно.
5	0 1 1 1 3 3 6 5 3 3 2 1\n	5	Программа работает правильно.
6	1 -2 -2 -2 -2 -3 -5 -6 -7 -8 -23 -2 -14 -16 -18 -20\n	9	Программа работает правильно.
7	2 1 2 -3 -5 -1 -4 -7 8 10 12\n	15	Программа работает правильно.
8	3 -1 3 5 -2 14 15 17 19 3 2 2 2\n	18	Программа работает правильно.

Выводы.

В ходе работы были изучены основные управляющие конструкции языка Си.

Разработана программа, выполняющая считывание с клавиатуры исходных данных с помощью функции *scanf()* и цикла *while(){}* и команды пользователя, для обработки команд пользователя использовалась символьная переменная *sym*, хранящая код символа ’ ‘, написаны функции, обрабатывающие входные данные, описание функций приведено в блоке “Выполнение работы”. С помощью оператора *switch(){}* и функции *printf()* реализован вывод значения определенной функции в зависимости от значения переменной *k*.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

int index_first_even(int A[], int b)
{
    int ind = 0;
    while ((abs(A[ind]) % 2 == 1) && (ind != b))
    {
        ind++;
    }
    return (ind);
}

int index_last_odd(int A[], int b)
{
    b = b - 1;
    while ((A[b] % 2 == 0) && (b >= 0))
    {
        b--;
    }
    return (b);
}

int sum_between_even_odd(int A[], int b)
{
    int f, l, summ = 0;
    f = index_first_even(A, b);
    l = index_last_odd(A, b);
    int i;
    for (i = f; i < l; i++)
    {
        summ = summ + abs(A[i]);
    }
    return (summ);
}

int sum_before_even_and_after_odd(int A[], int b)
{
    int summ = 0;
    int i;
    for (i = 0; i < b; i++)
    {
        summ = summ + abs(A[i]);
    }
    summ = summ - sum_between_even_odd(A, b);
    return summ;
}

int main()
{
    int k = 0;
    scanf("%d", &k);
```

```
    int arr[100];
    int arr_size = 0;
    char sym = ' ';
    while (arr_size < 100 && sym == ' ')
    {
        scanf("%i%c", &arr[arr_size++], &sym);
    }
    switch (k)
    {
    case 0:
        printf("%d\n", index_first_even(arr, arr_size));
        break;
    case 1:
        printf("%d\n", index_last_odd(arr, arr_size));
        break;
    case 2:
        printf("%d\n", sum_between_even_odd(arr, arr_size));
        break;
    case 3:
        printf("%d\n", sum_before_even_and_after_odd(arr, arr_size));
        break;
    default: printf("Данные некорректны\n"); break;
    }
    return 0;
}
```