МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Условия, циклы, оператор switch

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

Санкт-Петербург

2021

Цель работы.

Узнать, как работает препроцессор, как протсходят процессы компиляции и линковки. Научиться писать Make-файл, который упрощает работу с проектами, состоящими из нескольких файлов. Написать программу на языке Си, которая будет обрабатывать информацию по заданному алгоритму и

выводить результат.

Задание.

Вариант 4

В текущей директории создайте проект с make-файлом. Главная цель должна приводить к сборке проекта. Файл, который **реализует главную функцию**, должен называться menu.c; **исполняемый файл** - menu. Определение каждой функции должно быть расположено в **отдельном файле,** название файлов указано в скобках около описания каждой функции.

Реализуйте функцию-меню, 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.c)
1 : индекс последнего нечётного элемента. (index_last_odd.c)
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.c)
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.c)

иначе необходимо вывести строку "Данные некорректны".

Выполнение работы.

В начале подключается библиотека вывода stdio.h и заголовочный файл

stdlib.h

В программе реализованы функции:

- *int index_first_even(int ar[], int n)*
- *int index_last_odd(int ar[], int n)*
- *int sum_between_even_odd(int ar[], int n)*
- *int sum_before_even_and_after_odd(int ar[], int n)*

Каждая из этих функций вынесена в отдельный файл с расширением .c, также для каждой из функций создается одноименный заголовочный файл с расширением .h

Также создается Makefile, в котором описаны зависимости и указания компилятору. При запуске утилиты *make* будут созданы и залинкованы объектные файлы для каждого файла с кодом.

Рассмотрим каждую функцию:

Функция *int index_first_even(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 index_last_odd(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 *int sum_between_even_odd(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 две первые функции, после чего, используя цикл *for* накапливает сумму модулей элементов, используя *abs(ar[i])*, где *ar[i]* - элемент массива.

Функция *int sum_before_even_and_after_odd(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, после чего, используя цикл *for* накапливает сумму модулей элементов, используя *abs(ar[i])*, где *ar[i]* - элемент массива.

В теле основной функции *int main()* в файле *menu.c* 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 работа компилятора, процессы компиляции и линковки. Написан make-файл, с помощью которого собирается проект.

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: menu.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "index_first_even.h"
#include "index_last_odd.h"
#include "sum_between_even_odd.h"
#include "sum_before_even_and_after_odd.h"


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", index_first_even(ar, n));

    break;

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

case 2:


    printf("%d\n", sum_between_even_odd(ar, n));

    break;

case 3:
    printf("%d\n", sum_before_even_and_after_odd(ar, n));

    break;

default:
    printf("Данные некорректны");

    }
return 0;
```

Makefile:

```
all: menu

menu: sum_before_even_and_after_odd.o sum_between_even_odd.o index_first_even.o index_last_odd.o menu.o
    gcc menu.o sum_before_even_and_after_odd.o sum_between_even_odd.o index_first_even.o index_last_odd.o -o menu

menu.o: menu.c sum_before_even_and_after_odd.h sum_between_even_odd.h index_first_even.h index_last_odd.h
    gcc -c -std=c99 menu.c

sum_before_even_and_after_odd.o: sum_before_even_and_after_odd.c sum_before_even_and_after_odd.h
    gcc -c -std=c99 sum_before_even_and_after_odd.c

sum_between_even_odd.o: sum_between_even_odd.c sum_between_even_odd.h
    gcc -c -std=c99 sum_between_even_odd.c

index_first_even.o: index_first_even.c index_first_even.h
    gcc -c -std=c99 index_first_even.c

index_last_odd.o: index_last_odd.c index_last_odd.h
    gcc -c -std=c99 index_last_odd.c


clean:
    rm -rf *.o menu
```

index_first_even.c:

```c
#include "index_first_even.h"

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

}


```

index_first_even.h:

```c
#include <stdio.h>

int index_first_even(int ar[], int n);
```

index_last_odd.c:

```c
#include "index_last_odd.h"

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

index_last_odd.h:

```c
#include <stdio.h>

int index_last_odd(int ar[], int n);
```

sum_before_even_and_after_odd.c:

```c
#include "sum_before_even_and_after_odd.h"
#include "index_last_odd.h"
#include "index_first_even.h"

int sum_before_even_and_after_odd(int ar[], int n){
        int abssum = 0;
        int first = index_first_even(ar, n);
        int last = index_last_odd(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;
```

sum_before_even_and_after_odd.h:

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

int sum_before_even_and_after_odd(int ar[], int n);
```

sum_between_even_odd.c:

```c
#include "sum_between_even_odd.h"
#include "index_last_odd.h"
#include "index_first_even.h"

int sum_between_even_odd(int ar[], int n){
  int abssum = 0;
  int first = index_first_even(ar, n);
  int last = index_last_odd(ar, n);
  for (int i = first; i < last; i++){
            abssum += abs(ar[i]);
        }
        return abssum;
}
```

sum_between_even_odd.h:

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

int sum_between_even_odd(int ar[], int n);
```