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: Сборка программ в Си

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение порядка сборки программ 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.

Из библиотеки stdio.h: функции scanf() и printf() (Ввод и вывод соответственно).

Из библиотеки stdlib.h: функция abs() (Модуль числа).

Операторы: if(){} else{}, for (){}, while(){} , switch(){}.

Любой make-файл состоит из

- списка **целей**
- **зависимостей** этих целей
- **команд**, которые требуется выполнить, чтобы достичь эту цель

цель: зависимости

[tab] команда

Выполнение работы.

- ✓ ***menu.c***:

В функции *main{}* с помощью функции *skanf{}* присваивается целочисленное значение переменной *n*, объявленной ранее. Значение переменной *n* определяет то, к какой функции будет обращение в дальнейшем (или вывод "Данные некорректны"). Затем в теле функции объявляется целочисленный массив *arr* размером *N (N = 100)*, переменная *len*, которая хранит количество элементов массива (изначально *len* = 0), символьная переменная *gap= “ “*. Далее вводится сам массив посредствам цикла *while(){}*, где с помощью функции *skanf{}* вводится каждый целочисленный элемент массива *arr[i]* и идёт счёт элементов этого массива. После этого оператор *switch(n){}* выполняет различные команды, зависящие от значения переменной *n*.

При *n* = 0: происходит обращение к функции *index_first_even(){}*.

При *n* = 1: происходит обращение к функции *index_last_odd(){}*.

При *n* = 2: происходит обращение к функции *sum_between_even_odd (){}*.

При *n* = 3: происходит обращение к функции *sum_before_even_and_after_odd (){}*.

Возвращаясь к функции *main(){}*, точнее оператору *switch(){}* - при вводе определённого значения *n* оператор обращается к соответствующей функции и, с помощью функции *printf()*, выводит возвращённое значение (или сообщении о некорректных данных, при наличии таких), для выхода из оператора реализуется *break*.

Так как функция включает в себя функции *index_first_even(){ ,index_last_odd() {}, sum_between_even_odd (){}, sum_before_even_and_after_odd (arr, len){}* → посредствам *#include* включены заголовочные файлы: *index_first_even.h, index_last_odd.h, sum_between_even_odd.h, sum_before_even_and_after_odd .h*

- ✓ ***index_first_even***

Определение функции: *index_first_even.c*

Объявление функции: *index_first_even.h*

Функция *index_first_even(arr, len){}* получает на вход массив *arr* и значение переменной *len.* Происходит перебор каждого элемента массива посредствам цикла *for(){}* и поиск первого чётного элемента массива (остаток от деления на 2 равен 0). Для того, что бы отрицательные элементы массива так же рассматривались в переборе используется функция *abs().* Функция *index_first_even* возвращает индекс найденного элемента.

- ✓ ***index_last_odd***

Определение функции: *index_last_odd.c*

Объявление функции: *index_last_odd.h*

Функция *index_last_odd(arr, len){}* получает на вход массив *arr* и значение переменной *len*. Аналогично функции *index_first_even* происходит перебор элементов массива, но в данном случае с конца. Поиск последнего нечётного элемента (остаток от деления на 2 равен 1) осуществляется так же с использованием функции *abs().*Функция *index_last_odd* возвращает индекс найденного элемента.

- ✓ ***sum_between_even_odd***

Определение функции: *sum_between_even_odd.c*

Объявление функции: *sum_between_even_odd.h*

Функция *sum_between_even_odd (arr, len){}* получает на вход массив *arr* и значение переменной *len*. Объявляется целочисленная переменная *sum*, далее происходит перебор элементов массива посредствам цикла *for(){} (*Диапазон считается от первого чётного элемента (включая) и до последнего нечетного (не включая)). В переменную *sum* записывается сумма модулей элементов массива *arr* в данном диапазоне. Границы диапазона счёта определяют функции *index_first_even* и *index_last_odd*. Функция *sum_between_even_odd* возвращает значение переменной sum.

Функция использует функции *index_first_even* и *index_last_odd* → при помощи *#include* в файл включены *index_first_even.h , index_last_odd.h*

- ***sum_before_even_and_after_odd***

Определение функции: *sum_before_even_and_after_odd.c*

Объявление функции: *sum_before_even_and_after_odd.h*

Функция *sum_before_even_and_after_odd (arr, len){}* получает на вход массив *arr* и значение переменной *len*. Объявляется целочисленная переменная *summ,* далее происходит перебор элементов массива посредствам цикла *for(){}* (Диапазон захватывает весь массив). В переменную *summ* записывается сумма модулей элементов массива *arr* в данном диапазоне. После этого, из переменной *summ* вычитается значение, возвращённое функцией *sum_between_even_odd*, для получения суммы модулей элементов в диапазоне, данном в задании (От начала до первого чётного элемента (не включая) и от последнего нечётного элемента (включая) до конца массива).Функция *sum_before_even_and_after_odd* возвращает значение переменной *summ*.

Функция использует функцию *sum_between_even_odd* → при помощи *#include* в файл включен *sum_between_even_odd.h*

- **Makefile**

1) Инстукция: *all*

Зависимость: *menu.o index_first_even.o index_last_odd.o sum_between_even_odd.o sum_before_even_and_after_odd.o*

Команда*: gcc menu.o index_first_even.o index_last_odd.o sum_between_even_odd.o sum_before_even_and_afterodd.o -o menu*

Ключ *-o* определяет название исполняемого файла.

2) Инстукция: *menu.o*

Зависимость: *menu.c*

Команда: *gcc -c menu.c -std=c99*

↓

Создает объектный файл *menu.o*

3) Инстукция: *index_first_even.o*

Зависимость: *index_first_even.c index_first_even.h*

Команда: *gcc -c index_first_even.c -std=c99*

↓

Создает объектный файл *index_first_even.o*

4) Инстукция: *index_last_odd.o*

Зависимость: *index_last_odd.c index_last_odd.h*

Команда: *gcc -c index_last_odd.c -std=c99*

↓

Создает объектный файл *index_last_odd.o*

5) Инстукция: *sum_between_even_odd.o*

Зависимость: *sum_between_even_odd.c sum_between_even_odd.h*

Команда: *gcc -c sum_between_even_odd.c -std=c99*

↓

Создает объектный файл *sum_between_even_odd.o*

6) Инстукция: *sum_before_even_and_after_odd.o*

Зависимость: *sum_before_even_and_after_odd.c*

sum_before_even_and_after_odd.h

Команда: *gcc -c sum_before_even_and_after_odd.c -std=c99*

↓

Создает объектный файл *sum_before_even_and_after_odd.o*

(Ключ *-std=c99* сообщает компилятору стандарт языка Си, необходимый для работы программы)

Разработанный программный код см. в приложении А.

Тестирование.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 1 2 3 4 5 6 7 8 9 10\n	0	Ответ верный.
2.	1 1 2 3 4 5 6 7 8 9 10\n	8	Ответ верный.
3.	2 1 2 3 4 5 6 7 8 9 10\n	35	Ответ верный.
4.	3 1 2 3 4 5 6 7 8 9 10\n	10	Ответ верный.
5.	8 1 2 3 4 5 6 7 8 9 10\n	данные некорректны	Ответ верный.
6.	1 -1 -1 -1 -1 -1 -1 -3 -4\n	6	Ответ верный.
7.	3 -1 -2 -3 -4 -5 -1 -2 -3\n	4	Ответ верный.
8.	2 -34 5 78 -9 0 -32 1 4 15 -3 -3 8 9 34\n	192	Ответ верный.
9.	3 5 7 9 33 4 77 121 67 -86 -15 76 0 32 -78 36 42 17 26 -17 -90 -9 16 28 14\n	121	Ответ верный.
10.	1 0 0 0 0 0 0 0 0 0 -2 -4 -6 7 0 0 0\n	12	Ответ верный.

Выводы.

Был изучены порядок сборки программ в Си.

Разработана программа, выполняющая считывание с клавиатуры исходных данных с помощью функции *scanf()* и цикла *while(){}*, для обработки команд пользователя использовались: условный оператор *if*, циклы *for(){}* и *while(){}* , оператор множественного выбора *switch(){}*, функция *abs()*, функции, обрабатывающие входные данные, функция *printf()*.

Создан Makefile, содержащий инструкции по сборке программы, зависимости и необходимые команды. Все функции находятся в отдельных файлах, созданы заголовочные файлы, в которых хранятся объявления функций.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1) Название файла: menu.c

```
#include <stdio.h>
#include <stdlib.h>
#define N 100
#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 n;
    scanf("%d", &n);
    int arr[N];
    int len=0;
    int i=0;
    char gap= ' ';

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

2) Название файла: index_first_even.c

```c
#include <stdlib.h>
int index_first_even(int arr[], int len){
    for (int i=0; i<len; i++){
        if((abs(arr[i]))%2==0){
            return i;
        }
    }
}
```

3) Название файла: index_first_even.h

```c
int index_first_even();
```

4) Название файла: index_last_odd.c

```c
#include <stdlib.h>
int index_last_odd(int arr[], int len ){
    len = len -1;
    for (int i= len; i>=0; i--){
        if((abs(arr[i]))%2==1){
            return i;
        }
    }
}
```

5) Название файла: index_last_odd.h

```c
int index_last_odd();
```

6) Название файла: sum_between_even_odd.c

```c
#include <stdlib.h>
#include "index_first_even.h"
#include "index_last_odd.h"
int sum_between_even_odd(int arr[], int len){
    int sum=0;
    for (int i=index_first_even(arr, len); i<index_last_odd(arr, len); i++){
        sum=sum+abs(arr[i]);
    }
    return sum;
}
```

7) Название файла: sum_between_even_odd.h

```c
int sum_between_even_odd();
```

8) Название файла: sum_before_even_and_after_odd.c

```
#include <stdlib.h>
#include "index_first_even.h"
#include "index_last_odd.h"
#include "sum_between_even_odd.h"
int sum_before_even_and_after_odd(int arr[], int len){
   int summ=0;
   for (int i = 0; i < len; i++){
   summ=summ+abs(arr[i]);
   }
   summ=summ-sum_between_even_odd(arr, len);
   return summ;
}
```

9) Название файла: sum_before_even_and_after_odd.h

```
int sum_before_even_and_after_odd();
```

10)Название файла: Makefile

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