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

Студент гр. 0382 _____ Сергеев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение процесса сборки программ на языке Си при помощи утилиты Make.

Задание.

Создать проект функции-меню с использованием make-файла.

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. (функция index_first_even)

1 : индекс последнего нечётного элемента. (функция index_last_odd)

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

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

иначе необходимо вывести строку "Данные некорректны".

Определение каждой функции должно быть расположено в отдельном файле.

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.

- Файл menu.c:

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*)

Определение функции находиться в файле index_first_even.c, а объявление в файле *index_first_even.h*.

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*.

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)*

Определение функции находиться в файле *index_last_odd.c*, а объявление в файле *index_last_odd.h*.

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*.

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)*

Определение функции находиться в файле *sum_between_even_odd.c*, а объявление в файле *sum_between_even_odd.h*. Так как в функции *sum_between_even_odd* используются значения функций *index_first_even* и *index_last_odd*, то в файл *sum_between_even_odd.c* с помощью директивы *#include* подключены заголовочные файлы *index_first_even.h* и *index_last_odd.h*. Также, так как в функции используется функция *abs()*, то подключается стандартная библиотека *stdlib.h*

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*.

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)*

Определение функции находиться в файле *sum_before_even_and_after_odd.c*, а объявление в файле *sum_before_even_and_after_odd.h*. Так как в функции *sum_before_even_and_after_odd* используются значения функций *index_first_even*, *index_last_odd и sum_between_even_odd*, то в файл *sum_before_even_and_after_odd.c* с помощью директивы *#include* подключены заголовочные файлы *index_first_even.h, index_last_odd.h и sum_between_even_odd.h*. . Также, так как в функции используется функция *abs()*, то подключается стандартная библиотека *stdlib.h*

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*.

- Файл menu.c:

При значении *k*, отличном от 0,1,2 или 3, с помощью функции *printf*() печатается строка “Данные некорректны”.

- Makefile:
  1. Инструкция all:
     1) Зависимости: *menu.o, index_first_even.o, index_last_odd.o, sum_between_even_odd.o, sum_before_even_and_after_odd.o*
     2) Команда: *gcc menu.o index_first_even.o index_last_odd.o sum_between_even_odd.o sum_before_even_and_after_odd.o -o menu*
     3) Что происходит: Выполнение данной инструкции приводит к сборке исполняемого файла с именем *menu* из объектных

2. Инструкция *menu.o*
    1) Зависимости: *menu.c, index_first_even.h, index_last_odd.h, sum_between_even_odd.h, sum_before_even_and_after_odd.h*
    2) Команда*: gcc -c menu.c*
    3) Что происходит: Создаётся объектный файл *menu.o*
3. Инструкция *index_first_even.o*
    1) Зависимости: *index_first_even.c, index_first_even.h*
    2) Команда*: gcc –c index_first_even.c*
    3) Что происходит: Создаётся объектный файл *index_first_even.o*
4. Инструкция *index_last_odd.o*
    1) Зависимости: *index_last_odd.c, index_last_odd.h*
    2) Команда: *gcc –c index_last_odd.c*
    3) Что происходит: Создаётся объектный файл *index_last_odd.o*
5. Инструкция *sum_between_even_odd.o*
    1) Зависимости: *sum_between_even_odd.c, index_last_odd.h, index_first_even.h sum_between_even_odd.h*
    2) Команда: *gcc –c sum_between_even_odd.c*
    3) Что происходит: Создаётся объектный файл *sum_between_even_odd.o*
6. Инструкция *sum_before_even_and_after_odd.o*
    1) Зависимости: *sum_before_even_and_after_odd.c, index_first_even.c, index_first_even.h, sum_between_even_odd.h, sum_before_even_and_after_odd.h*
    2) Команда: *gcc –c sum_before_even_and_after_odd.o*
    3) Что происходит: Создаётся объектный файл *sum_before_even_and_after_odd.o*

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 был изучен процесс сборки программы на языке Си при помощи утилиты Make.

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*.

Все функции хранятся в отдельных файлах. Для каждой функции создан файл с расширением *.c и *.h, в которых соответственно хранятся определение функции и её объявление. Основная функция main находится в файле menu.c. Разработан make-файл, в котором расположены инструкции по сборке программы, указаны зависимости этих инструкций и команды, которые необходимо выполнить. Результатом работы утилиты Make является исполняемый файл menu.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1.Название файла: menu.c

```
#include <stdio.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 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;
}
```

2.Название файла: index_first_even.c

```
#include "index_first_even.h"

int index_first_even(int A[], int b)

{

    int ind = 0;

    while ((A[ind] % 2 != 0) && (ind != b))

    {

        ind++;

    }

    return ind;

}
```

3.Название файла: index_first_even.h

```
int index_first_even();
```

4.Название файла: index_last_odd.c

```
#include "index_last_odd.h"
int index_last_odd(int A[], int b)
{
    b = b - 1;
    while ((A[b] % 2 == 0) && (b >= 0))
    {
        b--;
    }
    return b;
}
```

5.Название файла: index_last_odd.h

```
int index_last_odd();
```

6.Название файла: sum_between_even_odd.c

```
#include "sum_between_even_odd.h"
#include "index_first_even.h"
#include "index_last_odd.h"
#include <stdlib.h>
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);
}
```

7.Название файла: sum_between_even_odd.h

```
int sum_between_even_odd();
```

8.Название файла: sum_before_even_and_after_odd.c

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

```
#include "sum_between_even_odd.h"
#include "sum_before_even_and_after_odd.h"
#include <stdlib.h>
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;
}
```

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_be-
tween_even_odd.o sum_before_even_and_after_odd.o -o menu
menu.o: menu.c index_first_even.h index_last_odd.h sum_between_even_odd.h
sum_before_even_and_after_odd.h
        gcc -c menu.c
index_first_even.o: index_first_even.c index_first_even.h
        gcc -c index_first_even.c
index_last_odd.o: index_last_odd.c index_last_odd.h
        gcc -c index_last_odd.c
sum_between_even_odd.o: sum_between_even_odd.c index_first_even.h in-
dex_last_odd.h sum_between_even_odd.h
        gcc -c sum_between_even_odd.c
sum_before_even_and_after_odd.o: sum_before_even_and_after_odd.c in-
dex_first_even.h index_last_odd.h  sum_between_even_odd.h sum_be-
fore_even_and_after_odd.h
        gcc -c sum_before_even_and_after_odd.c
```