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 Довченко М.К .

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

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

Задание.

Создать 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".

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(){}.
Также был использован make-файл, который состоит из: списка целей, зависимости этих целей, команд, которые требуется выполнить, чтобы достичь эту цель.

Выполнение работы.

Каждая функция данной программы находится в отдельном файле с идентичным названием. В файле menu находится главная функция

main. Также используются заголовочные файлы для объявления функций с расширением *.h.

Список файлов, содержащих одноименные функции и заголовочных файлов:

- menu.c
- index_first_even.c
- index_first_even.h
- index_last_odd.c
- index_last_odd.h
- sum_before_even_and_after_odd.c
- sum_before_even_and_after_odd.h
- sum_between_even_odd.c
- sum_between_even_odd.h

Описание работы функций:

В функции *main{}* объявляется целочисленная переменная val, которой с помощью функции *scanf()* присваивается целочисленное значение. Далее объявляется целочисленный массив *array* размером 100 и целочисленная переменная *size* равная 0, которая показывает количество элементов в массиве. В следующей строчке объявляется символьная переменная spaceb = *‘ ‘*. Далее в теле цикла *while (size <100 && spaceb==’ ’){}* применяется функция *scanf()*, с помощью которой вводится целый элемент массива *array[]* с индексом *size++* и сим вол spaceb. Далее применяется оператор *switch(val){}*, который в зависимости о

т значения val, будет выполнять различные команды.

Если val равняется 0, то с помощью функции *printf()* печатается значение функции *index_first_even(array, size)*. Функция *index_first_even(int [], int)* получает на вход целочисленный массив array и целое число size, затем в функции создаётся локальная целочисленная переменная counter0, равная 0. Используя цикл for(counter0 = 0; counter0 <= size; counter0++), в теле которого counter0 увеличивается на 1 за итерацию, удаётся найти индекс первого чётного элемента(фукнция *abs()* используется, так как если

array[counter0] будет отрицательным, то в случае нечётности элемента значение array*[counter0]%2* будет равно -1 и цикл завершится. Функция возвращает значение counter0. Для выхода из оператора *switch* используется *break.*

Если val равняется 1, то с помощью функции *printf()* печатается значение функции *index_last_odd(array, size)*. Функция *index_last_odd(int [], int){}* получает на вход целочисленный массив array и целое число size. Затем, используя цикл for(counter1 = size-1; counter1 >= 0; counter1--), в теле которого counter1 уменьшается на 1 за итерацию, удаётся найти индекс последнего нечётного элемента в массиве. Функция возвращает значение counter1. Для выхода из оператора *switch* используется *break.*

Если val равняется 2, то с помощью функции *printf()* печатается значение функции *sum_between_even_odd(array, size)*. Функция *sum_between_even_odd(int [], int){}* получает на вход целочисленный массив array и целое число size. Объявляется 2 локальные целочисленные переменные: counter 2, sbed = 0, где counter2 присваевается индекс первого чётного элемента массива(находится с помощью ранее описанной функции), sbed – искомая сумма. Далее с помощью цикла *for(counter2 = index_first_even(array, size); counter2 < index_last_odd(array , size); counter2++){}* с телом *sbed += abs(array[counter2])* , находится сумма ч ленов массива от первого чётного(включая) до последнего нечетного(исключая). Функция возвращает значение *summ*. Для выхода из оператора switch используется *break.*

Если val равняется 3, то с помощью функции *printf()* печатается значение

функции *sum_before_even_and_after_odd(array, size)*. Функция *sum_before_even_and_after_odd (int [], int){}* получает на вход целочисленный массив и целое число. Объявляются три целочисленные локальные переменн ые *sbeaao=0*, где *sbeaao* –искомая сумма, *counter3 и counter4*. Далее с помощью 2х циклов находится сумма всех элементов массива. После чего функция возвращает значение *sbeaao* . Для выхода из оператора switch(){}используется *break*.

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

Описание Makefile

Файл Makefile предназначен для сборки самого проекта и создания исполняемого файла menu. В Makefile содержатся следующие инструкции:

Инструкция all.

Имеет следующие зависимости: *index_first_even.o, index_last_odd.o, sum_between_even_odd.o, sum_before_even_and_after_odd.o, menu.o.*

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

Выполнение данной инструкции приводит к сборке проекта из необходимых объектных файлов.

С помощью ключа «-o» сообщается название получаемого после выполнения исполняемого файла — *menu.*

Инструкция *menu.o.*

Имеет следующие зависимости: *menu.c, index_first_even.h,*

index_last_odd.h, sum_between_even_odd.h,

sum_before_even_and_after_odd.h.

Команда: *gcc -c menu.c*

Выполнение данной инструкции приводит к созданию объектного файла *menu.o.*

Инструкция *index_first_even.o.*

Имеет следующие зависимости: *index_first_even.c, index_first_even.h.*

Команда: *gcc -c index_first_even.c*

Выполнение данной инструкции приводит к созданию объектного файла *index_first_even.o.*

Инструкция *index_last_odd.o*.

Имеет следующие зависимости: *index_last_odd.c, index_last_odd.h.*

Команда: *gcc -c index_last_odd.c*

Выполнение данной инструкции приводит к созданию объектного файла *index_last_odd.o.*

Инструкция sum_between_even_odd.*o.*

Имеет следующие зависимости: sum_between_even_odd.*c* sum_between_even_odd.*h, index_first_even.h, index_last_odd.h.*

Команда: *gcc -c* sum_between_even_odd.*c* .

Выполнение данной инструкции приводит к созданию объектного файла sum_between_even_odd.*o.*

Инструкция *sum_before_even_and_after_odd.o.*

Имеет следующие зависимости: *sum_before_even_and_after_odd.c, sum_before_even_and_after_odd.h, index_first_even.h, index_last_odd.h.*

Команда: *gcc -c sum_before_even_and_after_odd.c*

Выполнение данной инструкции приводит к созданию объектного файла *sum_before_even_and_after_odd.o.*

Инструкция *clean.*

Используется для удаления всех объектных файлов из текущей директории.

Команда: *rm *.o*

Исходный код всех файлов см. в приложении А.

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 8 -23 -30 -11 -28 15	0	Программа работает правильно.
6	1 8 -23 -30 -11 -28 15	5	Программа работает правильно.

7	2 8 -23 -30 -11 -28 15	100	Программа работает правильно.
8	3 8 -23 -30 -11 -28 15	15	Программа работает правильно.

Вывод.

В ходе работы был изучен процесс сборки программ на языке Си при помощи утилиты Make.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ФАЙЛОВ ПРОЕКТА

Файл menu.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 index_first_even(int[], int);
int index_last_odd(int[], int);
int sum_between_even_odd(int[], int);
int sum_before_even_and_after_odd(int[], int);

int main(){
    int size = 0, array[100], val;
    char spaceb = ' ';

    scanf("%d", &val);

    while(size < 100 && spaceb == ' '){
        scanf("%d%c",&array[size++], &spaceb);
    }

    switch (val){
        case 0:
            printf("%d\n", index_first_even(array, size));
```

```
            break;
        case 1:
            printf("%d\n", index_last_odd(array, size));
            break;
        case 2:
            printf("%d\n", sum_between_even_odd(array, size));
            break;
        case 3:
            printf("%d\n", sum_before_even_and_after_odd(array, size));
            break;
        default:
            printf("Данные некорректны\n");
            break;
    }
    return 0;
}
```

Файл index_first_even.c

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

int index_first_even(int array[], int size){
    int counter0;
    for(counter0 = 0; counter0 <= size; counter0++)
        if(abs(array[counter0])%2 == 0)
            return counter0;
}
```

Файл index_first_even.h

```
int index_first_even(int array[], int size);
```

Файл index_last_odd.c

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

int index_last_odd(int array[], int size){
    int counter1;
    for(counter1 = size-1; counter1 >= 0; counter1--)
        if(abs(array[counter1])%2 == 1)
            return counter1;
}
```

Файл index_last_odd.h

```c
int index_last_odd(int array[], int size);
```

Файл sum_before_even_and_after_odd.c

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

int sum_before_even_and_after_odd(int array[], int size){
    int counter3, counter4, sbeaao = 0;
    for(counter3 = index_first_even(array, size) - 1;  counter3 >= 0;  counter3--)
        sbeaao += abs(array[counter3]);
    for(counter4 = index_last_odd(array, size); counter4 < size; counter4++)
        sbeaao += abs(array[counter4]);
    return sbeaao;
}
```

Файл sum_before_even_and_after_odd.h

```c
int sum_before_even_and_after_odd(int array[], int size);
```

Файл sum_between_even_odd.c

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

int sum_between_even_odd(int array[], int size){
    int counter2, sbed = 0;
    for(counter2 = index_first_even(array, size); counter2 < index_last_odd(array, size); counter2++)
        sbed += abs(array[counter2]);
    return sbed;
}
```

Файл sum_between_even_odd.h

```c
int sum_between_even_odd(int array[], int size);
```

Файл Makefile

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

```
3. menu.o: menu.c index_first_even.h index_last_odd.h sum_between_even_odd.h
   sum_before_even_and_after_odd.h
4.     gcc -c menu.c
5. index_first_even.o: index_first_even.c index_first_even.h
6.     gcc -c index_first_even.c
7. index_last_odd.o: index_last_odd.c index_last_odd.h
8.     gcc -c index_last_odd.c
9. sum_between_even_odd.o: sum_between_even_odd.c sum_between_even_odd.h inde
   x_first_even.h index_last_odd.h
10.    gcc -c sum_between_even_odd.c
11.sum_before_even_and_after_odd.o: sum_before_even_and_after_odd.c sum_befor
   e_even_and_after_odd.h index_first_even.h index_last_odd.h
12.    gcc -c sum_before_even_and_after_odd.c
13.clean:
14.    rm *.o
15.
```