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. 1304 _____ Шаврин А.П

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2021

Цель работы.

Изучить способ сборки программ на языке Си с помощью утилиты make, а также процессы прекомпиляции, компиляции, линковки.

Задание.

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 нулевого элемента. (index_first_zero.c)
1 : индекс последнего нулевого элемента. (index_last_zero.c)
2 : Найти сумму модулей элементов массива, расположенных от первого нулевого элемента и до последнего. (sum_between.c)
3 : Найти сумму модулей элементов массива, расположенных до первого нулевого элемента и после последнего. (sum_before_and_after.c)

иначе необходимо вывести строку "Данные некорректны".
Ошибкой в данном задании считается дублирование кода!
Подсказка: функция нахождения модуля числа находится в заголовочном файле stdlib.h стандартной библиотеки языка Си.
При выводе результата, не забудьте символ переноса строки

Основные теоретические положения.

- Заголовочный файл задается с расширением .h
- Объектные файлы являются результатом компиляции и создаются с расширением .o
- Makefile – файл для быстрой и удобной сборки программы. Он состоит из цели, зависимости и исполнения:

  Структура мэйкфайла:

  цель: зависимости

  [tab] команда

Выполнение работы.

Все функции были выполнены в соответствии с заданием, описанным в лабораторной работе №1. Каждая функция размещена в отдельном файле с

названием соответствующим названию функции. Для каждого файла с функцией был создан заголовочный файл с таким же названием и расширением *.h*. С помощью директив препроцессора *#ifndef, #define, #endif* в заголовочных файлах была осуществлена защита от повторного включения.

Был создан Makefile для автоматизации сборки программы. Добавлена возможность удаления всех объектных файлов и исполнительного с помощью цели *clean*. В сборке программы последовательно реализованы все цели, начиная с главной *menu* и заканчивая функциями.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 -3 3 0 4 -5 4 0 7 -9 0 4 0 4 -4 -5	2	Функция по поиску индекса первого нуля в массиве выполнилась корректно.
2.	1 -3 3 0 4 -5 4 0 7 -9 0 4 0 4 -4 -5	11	Функция по поиску индекса последнего нуля в массиве выполнилась корректно.
3.	2 -3 3 0 4 -5 4 0 7 -9 0 4 0 4 -4 -5	33	Функция по вычислению суммы значений в массиве от первого нуля до последнего выполнилась корректно.
4.	3 -3 3 0 4 -5 4 0 7 -9 0 4 0 4 -4 -5	19	Функция по вычислению суммы значений в массиве до первого нуля и после последнего выполнилась корректно.
5.	4 -3 3 0 4 -5 4 0 7 -9 0 4 0 4 -4 -5	Данные некорректны	Значение пользователя было не корректно и выполнилось действие по умолчанию.

Выводы.

Были исследованы: способ сборки программ на языке Си с помощью утилиты make, а также процессы прекомпиляции, компиляции, линковки.

Была разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Все функции реализованы в файлах, с названиями соответствующих названиях функций, с расширением *.c*. Для каждой функции создан заголовочный файл с расширением *.h*. Создан корректно работающий Makefile, собирающий программу.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *index_first_zero.c*

```
#include "index_first_zero.h"
int index_first_zero(int arr[], int len){
	int n=0;
	 while(n < len){
		if(arr[n] == 0)
			return n;
		n++;
	}
	return -1;
}
```

Название файла: *index_first_zero.h*

```
#ifndef __index_first_zero__
	#define __index_first_zero__
	int index_first_zero(int arr[], int len);
#endif
```

Название файла: *index_last_zero.c*

```
#include "index_last_zero.h"
int index_last_zero(int arr[], int len){
	for(int i=len-1; i>=0; i--){
		if(arr[i] == 0)
			return i;
	}
	return -1;
}
```

Название файла: *index_last_zero.h*

```
#ifndef __index_last_zero__
    #define __index_last_zero__
    int index_last_zero(int arr[], int len);
#endif
```

Название файла: *sum_between.c*

```
#include <stdlib.h>
#include "index_first_zero.h"
#include "index_last_zero.h"
int sum_between(int arr[], int len){
    int sum=0;
    for(int i=index_first_zero(arr, len)+1; i != index_last_zero(arr, len); i++)
{
        sum += abs(arr[i]);
    }
    return sum;
}
```

Название файла: *sum_between.h*

```
#ifndef __sum_between__
    #define __sum_between__
    int sum_between(int arr[], int len);
#endif
```

Название файла: *sum_before_and_after.c*

```
#include <stdlib.h>
#include "index_first_zero.h"
#include "index_last_zero.h"
int sum_before_and_after(int arr[], int len){
    int sum=0;
    for(int i=0; i<len; i++){
```

```
            if((i >= index_first_zero(arr, len)) && (i <= index_last_zero(arr, len)))
                continue;
            sum += abs(arr[i]);
        }
        return sum;
    }
```

Название файла: *sum_before_and_after.h*

```
#ifndef __sum_before_and_after__
    #define __sum_before_and_after__
    int sum_before_and_after(int arr[], int len);
#endif
```

Название файла: *read_arr.c*

```
#include <stdio.h>
int read_arr(int arr[], int len){
    char c;
    int n=0;
    while(n<len){
        scanf("%d%c", &arr[n], &c);
        n++;
        if(c == '\n')
            break;
    }
    return n;
}
```

Название файла: *read_arr.h*

```
#ifndef __read_arr__
    #define __read_arr__
```

```
    int read_arr(int arr[], int len);
#endif
```

Название файла: *user_choice.c*

```
#include <stdio.h>
#include "index_first_zero.h"
#include "index_last_zero.h"
#include "sum_between.h"
#include "sum_before_and_after.h"
#define ERROR printf("Данные некорректны\n")
void user_choice(int x, int arr[], int len){
    switch(x){
        case 0:
            printf("%d\n", index_first_zero(arr, len));
            break;
        case 1:
            printf("%d\n", index_last_zero(arr, len));
            break;
        case 2:
            printf("%d\n", sum_between(arr, len));
            break;
        case 3:
             printf("%d\n", sum_before_and_after(arr, len));
            break;
        default:
            ERROR;
    }
}
```

Название файла: *user_choice.h*

```
#ifndef __user_choice__
```

```
#define __user_choice__
void user_choice(int x, int arr[], int len);
#endif
```

Название файла: *menu.c*

```
#include <stdio.h>
#include "read_arr.h"
#include "user_choice.h"
#define N 100
int main(){
    int arr[N];
    int x;
    int n;
    scanf("%d", &x);
    n = read_arr(arr, N);
    user_choice(x, arr, n);
    return 0;
}
```

Название файла: *Makefile*

```
CC = gcc

all: menu

menu: menu.o user_choice.o sum_before_and_after.o sum_between.o index_last_zero.o index_first_zero.o read_arr.o
        $(CC) menu.o user_choice.o sum_before_and_after.o sum_between.o index_last_zero.o index_first_zero.o read_arr.o -o menu

menu.o: menu.c menu.h user_choice.h read_arr.h
    $(CC) -c menu.c
```

user_choice.o: user_choice.c user_choice.h sum_before_and_after.h sum_between.h index_last_zero.h index_first_zero.h

$(CC) -c user_choice.c

sum_before_and_after.o: sum_before_and_after.c sum_before_and_after.h index_last_zero.h index_first_zero.h

$(CC) -c sum_before_and_after.c

sum_between.o: sum_between.c sum_between.h index_last_zero.h index_first_zero.h

$(CC) -c sum_between.c

index_last_zero.o: index_last_zero.c index_last_zero.h

$(CC) -c index_last_zero.c

index_first_zero.o: index_first_zero.c index_first_zero.h

$(CC) -c index_first_zero.c

read_arr.o: read_arr.c read_arr.h

$(CC) -c read_arr.c

clean:

*rm -rf *.o menu*