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 _____ Гудов Н.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

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

Задание.

3 Вариант

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".

Основные теоретические положения.

Препроцессор - это программа, которая подготавливает код программы для передачи ее компилятору.

Команды препроцессора называются директивами и имеют следующий формат:

#ключевое_слово параметры
Основные действия, выполняемые препроцессором:

- Удаление комментариев
- Включение содержимого файлов (*#include*)

- Макроподстановка (*#define*)
- Условная компиляция (*#if, #ifdef, #elif, #else, #endif*)

Компиляция - процесс преобразования программы с исходного языка высокого уровня в эквивалентную программу на языке более низкого уровня (в частности, машинном языке).

Компилятор - программа, которая осуществляет компиляцию.

Линковщик (компоновщик) - он принимает на вход один или несколько объектных файлов и собирает по ним исполняемый модуль.

Работа компоновщика заключается в том, чтобы в каждом модуле определить и связать ссылки на неопределённые имена.

Сборка проекта - это процесс получения исполняемого файла из исходного кода.

Если исходных файлов больше одного и требуется задавать некоторые параметры компиляции/линковки. Для этого используются Makefile - список инструкций для утилиты make, которая позволяет собирать проект сразу целиком.

Если запустить утилиту make, то она попытается найти файл с именем Makefile в текущей директории и выполнить из него инструкции.

Выполнение работы.

Используемые переменные:

space-тип char, для определения пробела

inp-тип int, для определения сценария работы программы

a-целочисленный массив для хранения последующего ряда цисел

Пользовательские функции:

index_first_zero и Index_last_zero для нахождения очереди ввода первого и последнего нуля соответственно.

sum_between и sum_before_and_after для нахождения сумм чисел стоящих относительно первого и последнего нулей.

Все функции возвращают целочисленное значение.

Файлы проекта:

menu.c – основной файл проекта, содержит функцию main

index_first_zero.c – файл, содержащий функцию для нахождения первого нулевого элемента.

index_first_zero.h – заголовочный файл, содержащий объявление функции index_first_zero.

index_last_zero.c – файл, содержащий функцию для нахождения последнего нулевого элемента.

index_last_zero.h – заголовочный файл, содержащий объявление функции index_last_zero.

sum_between.c - файл, содержащий функцию для нахождения суммы между первым и последним нулевыми элементами.

sum_between.h - заголовочный файл, содержащий объявление функции sum_between.

sum_before_and_after.c - файл, содержащий функцию для нахождения суммы до и после первого и последнего нулевых элементов.

sum_before_and_after.h - заголовочный файл, содержащий объявление функции sum_before_and_after.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2 3 0 4	2	Ошибок не обнаружено
2.	2 3 0 5 6 0 7	11	Ошибок не обнаружено
3.	0 0 46 -5	0	Ошибок не обнаружено
4.	3 -3 4 2 0 2 0 -2	11	Ошибок не обнаружено

Выводы.

Разработан проект из нескольких файлов, собирающийся при помощи Make-файла в исполняемый.

Изучен процесс сборки программ на языке Си при помощи утилиты Make.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#include <stdio.h>
#include <stdlib.h>
#include "index_first_zero.h"
#include "index_last_zero.h"
#include "sum_between.h"
#include "sum_before_and_after.h"
int main()
{
    char space =' ';
    int a[100];
    int inp, size = 0, ot;
    scanf("%d", &inp);
    while (size<100)
    {
        scanf("%d%c", &a[size++], &space);
        if(space !=' '){break;}
    }
    switch (inp)
    {
        case 0:
            ot = index_first_zero(a, size);
            printf("%d\n", ot);
            break;
        case 1:
            ot = index_last_zero(a, size);
            printf("%d\n", ot);
```

```
            break;
        case 2:
            ot = sum_between(a, size);
            printf("%d\n", ot);
            break;
        case 3:
            ot = sum_before_and_after(a, size);
            printf("%d\n", ot);
            break;

        default:
            printf("Данные некорректны\n");
            break;
    }
    return 0;
}
```

Название файла: sum_before_and_after.c

```
#include "sum_before_and_after.h"
#include "index_first_zero.h"
#include "index_last_zero.h"
#include <stdlib.h>
int sum_before_and_after(int a[], int size)
{
    int s = 0;
    int c = index_first_zero(a, size);
    int b = index_last_zero(a, size);
    for(int i = 0; i < c; i++)
    {
```

```
        s = s + abs(a[i]);
    }
    for(int i = b; i < size; i++)
    {
        s = s + abs(a[i]);
    }
    return s;
}
```

Название файла: sum_between.c

```
#include "sum_between.h"
#include "index_first_zero.h"
#include "index_last_zero.h"
#include <stdlib.h>
int sum_between(int a[], int size)
{
    int s = 0;
    int c = index_first_zero(a, size);
    int b = index_last_zero(a, size)+1;
    for ( int i = c; i < b; i++)
    {
        s = s + abs(a[i]);
    }
    return s;
}
```

Название файла: index_last_zero.c

```
#include "index_last_zero.h"
```

```
int index_last_zero(int a[], int size)
{
    int i = 0;
    int lz;


    for (i = 0; i < size; i++)
    {
        if (a[i] == 0)
        {
            lz = i;
        }
    }
    return lz;
}
```

Название файла: index_first_zero.c

```
#include "index_first_zero.h"
int index_first_zero(int a[], int size)
{
    int i, fz;

    for (i = 0; i < size; i++)
    {
        if (a[i] == 0)
        {
            fz = i;
            break;
        }
```

```
        }
        return fz;
    }
```

Название файла: sum_before_and_after.h

```
int sum_before_and_after();
```

Название файла: index_first_zero.h

```
int index_first_zero();
```

Название файла: index_last_zero.h

```
int index_last_zero();
```

Название файла: sum_between.h

```
int sum_between();
```

Название файла: Makefile

```
    all:  menu.o  index_first_zero.o  index_last_zero.o
sum_between.o sum_before_and_after.o
        gcc  menu.o  index_first_zero.o  index_last_zero.o
sum_between.o sum_before_and_after.o -o menu
    menu.o:  menu.c  index_first_zero.c  index_last_zero.c
sum_between.c sum_before_and_after.c
        gcc -c menu.c -std=c99
```

```
    index_first_zero.o: index_first_zero.c
        gcc -c index_first_zero.c -std=c99
    index_last_zero.o: index_last_zero.c
        gcc -c index_last_zero.c -std=c99
    sum_between.o: index_first_zero.c index_last_zero.c
        gcc -c sum_between.c -std=c99
    sum_before_and_after.o:            index_first_zero.c
index_last_zero.c
        gcc -c sum_before_and_after.c -std=c99
```