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 _____ Куликов М.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение сборки программ в Си с использованием make-файла.

Задание.

Реализовать функцию-меню с использованием make-файла, 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. Определение каждой функции должно быть расположено в отдельном файле, главная цель должна приводить к сборке проекта. Файл, который реализует главную функцию, должен называться menu.c; исполняемый файл - menu.

В зависимости от **значения**, функция должна выводить следующее:

0 : максимальное по модулю число в массиве. (abs_max.c)

1 : минимальное по модулю число в массиве. (abs_min.c)

2 : разницу между максимальным по модулю и минимальным по модулю элементом. (diff.c)

3 : сумму элементов массива, расположенных после максимального по модулю элемента (включая этот элемент). (sum.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*)

#include

Препроцессор обрабатывает содержимое указанного файла и включает содержимое на место директивы. Включаемые таким образом файлы называются заголовочными и обычно содержат объявления функций, глобальных переменных, определения типов данных и другое.
Директива может иметь вид #include "...." либо #include <...>. Для <...> поиск файла осуществляется среди файлов стандартной библиотеки, а для "..." - в текущей директории.

#define

Позволяет определить макросы или макроопределения. Имена их принято писать в верхнем регистре через нижние подчеркивания, если это требуется.
Следует обратить особое внимание, что define выполняет просто подстановку идентификатора (без каких-то дополнительных преобразований), что иногда может приводить к ошибкам, которые трудно найти.

#if, #ifdef, #elif, #else, #endif

Директивы условной компиляции допускают возможность выборочной компиляции кода. Это может быть использовано для настройки кода под определенную платформу, внедрения отладочного кода или проверки на повторное включение файла.

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

Сборка проекта вручную может стать довольно утомительным занятием, особенно, 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.

Любой make-файл состоит из

-списка целей

-зависимостей этих целей

-команд, которые требуется выполнить, чтобы достичь эту цель

Для сборки проекта обычно используется цель all, которая находится самой первой и является целью по умолчанию. (фактически, первая цель в файле и является целью по-умолчанию)

Также, рекомендуется создание цели clean, которая используется для очистки всех результатов сборки проекта

Использование нескольких целей и их зависимостей особенно полезно в больших проектах, так как при изменении одного файла не потребуется пересобирать весь проект целиком. Достаточно пересобрать измененную часть.

Выполнение работы.

Программа, созданная в ходе 1 лабораторной работе, была разделена на отдельные функции. Для этих файлов с функциями были созданы заголовочные файлы, объявляющие эти функции и был создан Makefile с главной целью all и дополнительной целью clean,с помощью которой удалялись объектные файлы из директории. Весь код предоставлен в Приложении А.

Описание функций:

1) abs_max() - первым аргументом данной функции является массив, вторым — количество элементов в нем. С помощью цикла for мы перебираем

этот массив и ищем элемент в максимальным значением по модулю и его индекс. Функция возвращает индекс максимального по модулю числа в массиве.

2) abs_min() - эта функция является копией прошлой функции за исключением того, что она ищет минимальный по модулю элемент и его индекс в массиве.

3) diff() - первым аргументом данной функции является массив, вторым — количество элементов в нем. С помощью функций abs_max и abs_min мы ищем разность между максимальным и минимальным по модулю элементом. Функция возвращает значение этой разности.

4) sum() - первым аргументом данной функции является массив, вторым — количество элементов в нем. С помощью функции abs_max и цикла for мы складываем значение всех элементов массива, которые идут после максимального, включая максимальный элемент. Функция возвращает сумму этих элементов.

Описание функции main:

Далее идет описание функции main:

Мы объявляем некоторые переменные:

int choice- выбор опции пользователем, которая будет использована в операторе switch

int arr[] - объявляем массив, в котором будут храниться введенные пользователем числа

int arr_size — количество заполненных ячеек массива

char sym — изначально этой переменной присваивается «пробел». Делается это для того, чтобы при вводе массива в массив вводилось только число.

Пользователь вводит цифру, опцию которой он хочет выбрать.
После этого мы начинаем заполнение массива:

Если количество элементов массива не превышает максимальный и переменная sym == « », тогда считывается сначала число массив, а потом символ, который должен являться пробелом.

Затем мы вызываем оператор switch:

Если пользователь ввел 0 — на экран выводится элемент массива с индексом abs_max.

Если пользователь ввел 1 -на экран выводится элемент массива с индексом abs_min.

Если пользователь ввел 2 — на экран выводится разница между максимальным и минимальным по модулю элементом массива.

Если пользователь ввел 3 — на экран выводится сумма элементов массива после максимального по модулю, включая максимальный

Если переменная choice имеет другое значение — на экран выводится сообщение «Данные некорректны»
Выполнение функции main заканчивается, возвращается 0.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 3 5 6 848 -938 -5 87 -4	-938	abs_max
2.	1 4564 0 93857 -875 -948 784	0	abs_min
3.	2 1 785 87 -45 876	875	diff
4	3 2 3 4 5 1 1 1 1	9	sum
5	5 6 5 7 3 5	Данные некорректны	error

Выводы.

В ходе работы был изучен просецц сборки программ с помощью make-файлов и была собрана программа, состоящая из главной функции и подфункций,а также их заголовочных файлов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#include <stdio.h>
#include "abs_max.h"
#include "abs_min.h"
#include "sum.h"
#include "diff.h"
#define max_amount 100
int main(){

    int arr[max_amount], arr_size = 0, choice;
    char sym = ' ';
    scanf("%d", &choice);
    while (arr_size < max_amount && sym == ' '){
        scanf("%d%c", &arr[arr_size++], &sym);
    }
    switch(choice){
        case 0:
            printf("%d\n", arr[abs_max(arr,arr_size)]);
            break;
        case 1:
            printf("%d\n", arr[abs_min(arr,arr_size)]);
            break;
        case 2:
            printf("%d\n",diff (arr,arr_size));
            break;
        case 3:
```

```
                printf("%d\n",sum (arr,arr_size));
                break;

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

Название файла: abs_max.c

```
#include "abs_max.h"
#include <stdlib.h>

int abs_max(int numbers[], int quantity){
    int maxnum = abs(numbers[0]);
    int maxindex = 0;
    for (int i = 1; i < quantity; i++){
        if( abs(numbers[i]) > abs(maxnum)){
            maxnum = abs(numbers[i]);
            maxindex = i;
        }
    }

    return maxindex;
}
```

Название файла: abs_min.c

```
#include "abs_min.h"
#include <stdlib.h>

int abs_min(int numbers[], int quantity){
    int minindex = 0;
    int minnum = abs(numbers[0]);
    for (int i = 1; i < quantity; i++){
        if( abs(numbers[i]) < minnum){
            minnum = abs(numbers[i]);
            minindex = i;
        }
    }

    return minindex;
}
```

Название файла: diff.c

```
#include <stdlib.h>
#include <stdlib.h>
#include "abs_max.h"
#include "abs_min.h"

int diff (int numbers[], int quantity){
      int diff = ( numbers[abs_max(numbers, quantity)] - numbers[abs_min(numbers, quantity)]);
    return diff;
}
```

Название файла: sum.c

```
#include <stdlib.h>
#include "sum.h"
#include "abs_max.h"

int sum (int numbers[], int quantity){
    int sum = 0;
    for(int i = abs_max(numbers, quantity); i < quantity; i++){
        sum += numbers[i];
    }
    return sum;
}
```

Название файла: abs_max.h

```
int abs_max(int numbers[], int quantity);
```

Название файла: abs_min.h

```
int abs_min(int numbers[], int quantity);
```

Название файла:

```
int diff(int numbers[], int quantity);
```

Название файла: sum.h

```
int sum(int numbers[], int quantity);
```

Название файла: Makefile

```
all: menu.o abs_min.o abs_max.o diff.o sum.o
	gcc menu.o abs_min.o abs_max.o diff.o sum.o -o menu
menu.o: abs_min.o abs_max.o diff.o sum.o
	gcc -c menu.c -std=c99
abs_min.o: abs_min.c abs_min.h
	gcc -c abs_min.c -std=c99
abs_max.o: abs_max.c abs_max.h
	gcc -c abs_max.c -std=c99
diff.o: diff.c diff.h
	gcc -c diff.c -std=c99
sum.o: sum.c sum.h
	gcc -c sum.c -std=c99
clean:
	rm  *.o
```