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. 0383 Шквиря Е.В.

Преподаватели: Чайка К.В.
Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучить процесс сборки программ на языке Си.

Задание.

Вариант 2.

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 : максимальное число в массиве. (max.c)
1 : минимальное число в массиве. (min.c)
2 : разницу между максимальным и минимальным элементом. (diff.c)
3 : сумму элементов массива, расположенных до минимального элемента. (sum.c)

иначе необходимо вывести строку "Данные некорректны".
```

Ошибкой в данном задании считается дублирование кода!

При выводе результата, не забудьте символ переноса строки

Выполнение работы.

Порядок выполнения поставленной задачи.

1. За основу программы была взята реализация решения задачи из лабораторной работы №1.

2. Для каждой функции из решения был создан файл с исходным кодом, в котором она описана, и заголовочный файл, где содержится её объявление. Для основного интерфейса управления программой также был создан файл с исходным кодом (menu.c).

3. С помощью директивы #include в файлы с исходным кодом и некоторые заголовочные добавлялись объявления функций, которые нужны были в дальнейшей реализации.

4. Сборка программы производиться с помощью специального файла Makefile, в котором прописаны цели для сборки проекта и дальнейшей очистки директории от объектных файлов после их использования. Цель *all* последовательно вызывает зависимые цели для создания объектных файлов, исполняемого файла и очистки директории от объектных файлов.

Используемые цели в Makefile:

all — сборка проекта.

build_proj — создание из объектных файлов исполняемый файл.

max.o — создание объектного файла из max.c.

min.o — создание объектного файла из min.c.

diff.o — создание объектного файла из diff.c.

sum.o — создание объектного файла из sum.c.

menu.o — создание объектного файла из menu.c.

clear — очистка директории от объектных файлов, в которых больше нет нужды.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования запуска Makefile в терминале

№ п/п	Входные данные	Выходные данные	Комментарии
1.	make	gcc -c max.c gcc -c min.c gcc -c diff.c gcc -c sum.c gcc -c menu.c gcc max.o min.o diff.o sum.o menu.o -o menu rm *.o	Программа запускается с помощью файла menu и полностью функционирует.

Выводы.

Был изучен процесс сборки программ на языке Си.

Разработана программа, считывающая с клавиатуры массив чисел и позволяющая найти в нём минимальный и максимальный элемент, разность между ними и сумму элементов до первого минимального элемента. В отдельные файлы с исходным кодом были вынесены все описания функций обработки массива, а их объявления - в заголовочные файлы. Также для сборки программы был создан специальный Makefile, который выполняет прописанные в нём цели для создания исполняемого файла и очистки директории.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

max.h:

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

max.c:

```
#include "max.h"

int max(int array[], int size)
{
	int maxElement = array[0];
	int i = 0;
	for (i = 1; i < size; ++i)
	{
		if (array[i] > maxElement)
			maxElement = array[i];
	}
	return maxElement;
}
```

min.h:

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

min.c:

```
#include "min.h"

int min(int array[], int size)
{
	int minElement = array[0];
	int i = 0;
	for (i = 1; i < size; ++i)
```

```
        {
                if (array[i] < minElement)
                        minElement = array[i];
        }
        return minElement;
}
```

diff.h:

```
#include "min.h"
#include "max.h"
int diff(int array[], int size);
```

diff.c:

```
#include "diff.h"

int diff(int array[], int size)
{
        return max(array, size) - min(array, size);
}
```

sum.h:

```
#include "min.h"
long sum(int array[], int size);
```

sum.c:

```
#include "sum.h"

long sum(int array[], int size)
{
        int minElement = min(array, size);
        long sumAnswer = 0;
        int i = 0;
```

```
        for (i = 0; i < size; ++i)
        {
                if (array[i] != minElement)
                        sumAnswer += array[i];
                else
                        return sumAnswer;
        }
}
```

menu.c:

```
#include <stdio.h>
#include "max.h"
#include "min.h"
#include "diff.h"
#include "sum.h"
const int MAX_SIZE = 100;
const int SIZE_ERROR = -1;
const char END_CHAR = '\n';

int main()
{
        int command = 0;
        scanf("%d", &command);
        int array[MAX_SIZE];
        int size = 0;
        int inputVar = 0;
        char tempChar = '$';

        while((tempChar = getchar()) != END_CHAR)
        {
                scanf("%d", &inputVar);
                array[size++] = inputVar;
```

```
	}

	//проверка для пустого массива
	if(size == 0)
		command = SIZE_ERROR;

	switch (command)
	{
		case 0://Максимальное число
			printf("%d\n", max(array, size));
			break;

		case 1://Минимальное число
			printf("%d\n", min(array, size));
			break;

		case 2://Разность между максимальным и минимальным
значением
			printf("%d\n", diff(array, size));
			break;

		case 3://Сумма чисел до первого минимального элемента
			printf("%ld\n", sum(array, size));
			break;

		default://Некорректные данные
			printf("Данные некорректны\n");
	}

	return 0;
}
```

Makefile:

```
all: build_proj clear

build_proj: max.o min.o diff.o sum.o menu.o
	gcc max.o min.o diff.o sum.o menu.o -o menu
max.o: max.c max.h
	gcc -c max.c
min.o: min.c min.h
	gcc -c min.c
diff.o: diff.c diff.h
	gcc -c diff.c
sum.o: sum.c sum.h
	gcc -c sum.c
menu.o: menu.c
	gcc -c menu.c
clear:
	rm *.o
```