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 _____ Поршнев Р.А.
Преподаватель _____ Чайка К.В.

Санкт-Петербург
2021

Цель работы.

Изучить основные команды для работы с Makefile.

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 : 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".

Вариант 2.

Основные теоретические положения.

В данной лабораторной работе использовалась библиотека *stdio.h*, сборка программы осуществлялась с помощью Makefile.

Выполнение работы.

В данной лабораторной работе не имеет смысла подробно

рассматривать принцип работы кода, ведь данная работа практически полностью копирует прошлую лабораторную работу за исключение того, что теперь каждая функция находится в отдельном одноимённом файле. О том, как каждую функцию поместили в отдельный файл, пойдёт речь далее.

Проект состоит из следующих файлов: *menu.c*, *max.c*, *max.h*, *min.c*, *min.h*, *diff.c*, *diff.h*, *sum.c*, *sum.h*, *readarr.c*, *readarr.h*.

В файле *menu.c* подключается библиотека <*stdio.h*>, а также заголовочные файлы *max.h*, *min.h*, *diff.h*, *sum.h*, *readarr.h*. В функции main объявляется массив *arr[N]* типа *int* и переменная *n* типа *int*, которой присваивается значение функции *readarr(arr)*, которое является числом элементов массива *arr*. Далее в программе находится оператор *switch*, с помощью которого, в зависимости от значения *arr[0]*, вызываются те или иные функции либо выводится сообщение о некорректности данных. В конце файла menu.c находится оператор *return*, возвращающий значение 0.

```
#include <stdio.h>
#include "max.h"
#include "min.h"
#include "diff.h"
#include "sum.h"
#include "readarr.h"

#define N 100

int main() {

      int arr[N];
      int n;
      n = readarr(arr);
      switch(arr[0]) {
            case 0:
              printf("%d\n", max(arr, n));
             break;
            case 1:
              printf("%d\n", min(arr, n));
             break;
            case 2:
               printf("%d\n", diff(arr, n));
               break;
            case 3:
               printf("%d\n", sum(arr, n));
               break;
            default:
               printf("Данные некорректны\n")
```

```
	}
	return 0;

}
```

В файле *readarr.h* находится объявление функции *readarr(int *arr).*

```
#define N 100

int readarr(int *arr, int n);
```

В файле *readarr.c* находится подключение заголовочного файла *readarr.h*, объявление функции *readarr(int *arr)* и её тело. Данная функция предназначена для ввода массива *arr*, а её результатом является длина введённого массива.

```
#include "readarr.h"

int readarr(int *arr) {

	int i, n = 0;
	char c;
	for(i = 0; i < N; i++) {
		n = n + 1;
		scanf("%d%c", &arr[i], &c);
		if (c == '\n') {
			break;
		}
	}
	return n;
}
```

В файле *max.h* находится объявление функции *max(int *arr, int n).*

```
int max(int *arr, int n);
```

В файле *max.c* находится подключение заголовочного файла *max.h*, объявление функции *max(int *arr, int n)* и её тело. Данная функция предназначена для нахождения максимального элемента в массиве *arr*.

```
#include "max.h"

int max(int *arr, int n) {

int i;
int maxinarr = arr[1];
for(i = 1; i < n; i++) {
```

```
         if (arr[i] >= maxinarr) {
            maxinarr = arr[i];
         }
      }
      return maxinarr;
}
```

В файле *min.h* находится объявление функции *min(int *arr, int n)*.

```
int max(int *arr, int n);
```

В файле *min.c* находится подключение заголовочного файла *min.h*, объявление функции *min(int *arr, int n)* и её тело. Данная функция предназначена для нахождения минимального элемента в массиве *arr*.

```
#include "min.h"

int min(int *arr, int n) {

   int i;
   int mininarr = arr[1];
   for(i = 1; i < n; i++) {
      if (arr[i] <= mininarr) {
         mininarr = arr[i];
      }
    }
return mininarr;

}
```

В файле *diff.h* находится объявление функции *diff(int *arr, int n)*.

```
int diff(int *arr, int n);
```

В файле *diff.c* находится подключение заголовочных файлов *diff.h, max.h, min.h*, объявление функции *diff(int *arr, int n)* и её тело. Данная функция предназначена для нахождения разности между максимальным и минимальным элементами в массиве *arr*. Именно поэтому требовалось подключать заголовочные файлы *max.h* и *min.h*, ведь функции *max(int *arr, int n)* и *min(int *arr, int n)* предназначены для поиска максимального и минимального элемента в массиве соответственно.

```
#include "diff.h"
#include "max.h"
```

```
#include "min.h"

int diff(int *arr, int n) {

      int maxinarr, mininarr,x;
      maxinarr = max(arr, n);
      mininarr = min(arr, n);
      x = maxinarr - mininarr;
      return x;

}
```

В файле *sum.h* находится объявление функции *sum(int *arr, int n).*

```
int sum(int *arr, int n);
```

В файле *sum.c* находится подключение заголовочных файлов *sum.h* и *min.h*, объявление функции *sum(int *arr, int n)* и её тело. Данная функция предназначена для нахождения суммы до первого минимального элемента в массиве *arr*. Именно поэтому требовалось подключать заголовочный файл *min.h*, ведь функция *min(int *arr, int n)* предназначена для поиска минимального элемента в массиве.

```
#include "sum.h"
#include "min.h"

int sum(int *arr, int n) {

      int i, x, summa;
      x = min(arr, n);
      summa = 0;
      for(i = 1; i < n; i++) {
            if (arr[i] == x) {
            break;
            }
      summa = summa + arr[i];
      }
return summa;

}
```

Далее нужно собрать программу с помощью Makefile. Нужно создать файл с названием Makefile. В начале заменяется *all* на *lb2* для большего удобства и общего понимания. Далее определяется цель *lb2* и зависимости для неё:

```
lb2: menu.o max.o min.o diff.o sum.o readarr.o
```

Далее каждая зависимость становится целью. Теперь для новой цели появляются новые зависимости и команды для них. К примеру, для того, чтобы получить выходной файл, нужно собрать объектные файлы каждой из функции, из которых состоит проект и скомпилировать их с ключом *-o*. Для того, чтобы получить объектные файлы, нужно скомпилировать файлы с функциями с расширением *.c* с ключом *-c*. Допустим, нужно получить объектный файл *max.o*. Для этого нужно взять файл *max.c* и скомпилировать его с ключом *-c*. То есть, требуется прописать команду *gcc -c max.c*. Такую процедуру нужно проделать с каждым файлом, входящим в проект. Позже все полученные объектные файлы компилируются с ключом *-o* для получения выходного файла. Однако стоит отметить, что для цели *diff.o* зависимостями являются файлы *diff.c*, *max.h* и *min.h*. Похожая ситуация с целью sum.o, которой соответствуют следующие зависимости: sum.c, min.h.

Так же можно добавить цель *clean* и зависимость rm -rf *.o lb2 для неё. Данная цель нужна для очистки текущей папки от файлов с расширением .o. Для очистки нужно в терминале написать команду *make -f Makefile clean*.

```
all: lb2
lb2: menu.o max.o min.o diff.o sum.o readarr.o
	gcc menu.o max.o min.o diff.o sum.o readarr.o -o menu
menu.o: menu.c max.h min.h diff.h sum.h readarr.h
	gcc -c menu.c
max.o: max.c
	gcc -c max.c
min.o: min.c
	gcc -c min.c
diff.o: diff.c max.h min.h
	gcc -c diff.c
sum.o: sum.c min.h
	gcc -c sum.c
readarr.o: readarr.c
	gcc -c readarr.c
clean:
	rm -rf *.o lb2
```

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 3 1 4 6 10 7 -1 2 -100	10	Ответ правильный
2.	0 10 4 7 89 100 2 1 4 3	100	Ответ правильный
3.	1 10 89 67 89 999 1000000	10	Ответ правильный
4.	1 13 89 67 89 1 12300 900	1	Ответ правильный
5.	2 3 1 4 2 6 4 8 6 10 11 12 20	19	Ответ правильный
6.	2 100 89 67 65 -100	200	Ответ правильный
7.	3 3 1 4 6 10 7 -1 2 -100	32	Ответ правильный
8.	3 1 2 3 4 5 6 7 8 9 10 11 12	0	Ответ правильный
7.	10 3 1 4 6 10 7 -1 2 -100	Данные некорректны	Ответ правильный
8.	-66 174 477 2 848 184	Данные некорректны	Ответ правильный

Выводы.

Я изучил сборку программ в языке Си и основные команды для работы с Makefile