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 языке Си

Студент гр. 3384 _____ Рудаков А.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2023

Цель работы.

Изучить процесс сборки программ из нескольких файлов с исходным кодом на языке Си. Применить их на практике, разработав программу на языке Си и создав Makefile для её сборки.

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".
Ошибкой в данном задании считается дублирование кода!
При выводе результата, не забудьте символ переноса строки

Выполнение работы.

Для организации ввода-вывода была использована стандартная библиотека Си (подключен заголовочный файл <*stdio.h*>).

Основным файлом программы является файл *menu.c*. В нем обозначены подключения функций из заголовочных файлов *"min.h", "max.h", "sum.h", "diff.h", "scan_array.h"*.

Далее была написана основная функция *int main()*. Внутри нее определены 3 переменные:

- *int array[102]* — массив, содержащий 102 элемента типа *int*, в который будут записаны считанные значения.
- *int n* — переменная типа int, в которую будет записано первое считанное число
- *int len = 0* — переменная типа *int*, в которой будет храниться количество считанных значений.
- *int i = 0* — счетчик цикла.

Функция ввода *scanf("%d",&n)* считывает первое поданное на вход число типа *int*, тип определяется благодаря спецификации *%d*, и передает значение переменной *n*, передача происходит благодаря *&*.

Далее вызывается функция *scan_array(array).*

Следующим действием выполняется цикл *for (int i=0;i<101;i++)*, в котором определяется внутренняя переменная *int i=0* типа *int*, имеющая начальное значение 0, условие продолжения цикла *i<101* (цикл будет выполняться до момента, пока условие верно) и действие, которое будет происходить после каждого прохода цикла *i++* - увеличение переменной *i* на 1. Данный цикл нужен для того, чтобы определить количество введенных значений. Внутри цикла находиться условный оператор *if (array[i]==101)*, который проверяет *i*-ый элемент массива *array* на его равенства числу 101. Так как в массиве *array* существует единственный элемент равный 101, который определяется в функции *scan_array*, идущий после всех введенных элементов, его индекс будет обозначать количество поданных на вход чисел. Если *i*-ый элемент равен 101, то выполняются действия внутри условного оператора, а именно присваивание переменной *len* значения, равного *i*, и выход из цикла, реализованный благодаря *break*.

Далее находиться условный оператор *if (array[101]!=102)*, который проверяет неравенство 101 элемента массива *array* значению 102. Если элемент равен данному значению, значит данные некорректны. 102-ой элемент определяется в функции *scan_array*, и становиться равным только в том случае, если введенные данные некорректны. В таком случае выполняется ветка *else*, в которой при помощи функции вывода *printf("Данные некоректны\n")* на экран выводиться сообщение о некорректности данных, *\n* в конце выводимого сообщения обозначает символ переноса строки.

Если же 101-ый элемент массива *array* не равен 102, то выполняется действие внутри условного оператора. Внутри оператора *if* находиться оператор *switch(n)*, зависящий от переменной *n*.

- Если *n=0*, то выполняются действия внутри *case 0*. Инициализируется переменная *maximum* типа *int*, которой присваивается значение, возвращаемое из функции *max(array,len)*, после чего происходит происходит вывод на экран значения переменной *maximum* при помощи *printf("%d\n",maximum)* и выход из условного оператора, реализованный через *break*.
- Если *n=1*, то выполняются действия внутри *case 1*. Инициализируется переменная *minimum* типа *int*, которой присваивается значение, возвращаемое из функции *min(array,len)*, после чего происходит происходит вывод на экран значения переменной *minimum* при помощи *printf("%d\n",minimum)* и выход из условного оператора, реализованный через *break*.
- Если *n=2*, то выполняются действия внутри *case 2*. Инициализируется переменная *difference* типа *int*, которой присваивается значение, возвращаемое из функции *diff(array,len)*, после чего происходит происходит вывод на экран значения переменной *difference* при помощи *printf("%d\n",difference)* и выход из условного оператора, реализованный через *break*.

- Если *n=3*, то выполняются действия внутри *case 3*. Инициализируется переменная *summ* типа *int*, которой присваивается значение, возвращаемое из функции *sum(array,len)*, после чего происходит происходит вывод на экран значения переменной *summ* при помощи *printf("%d\n",summ)* и выход из условного оператора, реализованный через *break*.
- Если *n* не равно ни одному из предыдущих значений, то выполняются действия внутри *default*. Функция вывода *printf("Данные некорректны\n")* выводит на экран сообщение о некорректности данных и происходит выход из условного оператора, реализованный через *break*. .

Далее была реализована функция *void scan_array(int arr[]),* предназначенная для ввода массива элементов. Она реализована при помощи типа функции *void*, невозвращающей значения. В ее аргументах содержится *int arr[]* - ссылка на массив *int array[]* в функции *int main().*

В данной функции определяются 3 переменные:

- *char symbol* — переменная типа *char*, которая в последующем будет хранить символ, стоящий в строке после очередного числа из введенного массива.
- *int i = 0* — переменная типа *int*, отвечающая за индекс считываемого элемента. Имеет начальное значение, равное 0.
- *int proverka = 0* — переменная типа *int*, благодаря которой определяется корректность введенных данных. Имеет начальное значение, равное 0.

Далее идет цикл *do { } while (symbol!='\n' && proverka==0)* с постусловием, который повторяется, пока верны оба условия. Первое условие *symbol!='\n'* проверяет наличие символа переноса строки после считанного значения. Второе условие *proverka==0* проверяет введенные данные на корректность. Пока значение равно 0, данные верны. Внутри цикла первым стоит условный оператор *if (scanf("%d%c",&arr[i],&symbol)==0),* внутри которого находиться функция ввода *scanf*, которая отвечает за считывание числа типа *int* и последующего символа. Спецификация *%d* — указывает на считывание числа типа *int* и передает значение в переменную *array[i]* через *arr[i] (*ссылку на него внутри

функции), которая принимает значение благодаря *&*. Спецификация *%c* — указывает на считывание символа типа *char* и передает значение в переменную *symbol*, которая принимает значение, благодаря *&*. В условном операторе проверяется корректность введенных значений, если введенные данные не корректны, то выполняются действия внутри *if*. Первым идет присваивание переменной *proverka* значения 1, вторым действием элементу массива *array* с индексом *i* через *arr[i] (*ссылку на него внутри функции) присваивается значение 102. Далее в цикле идет второй условный оператор *if*, имеющий условие *(arr[i]>100)*, который проверяет корректность введенных данных, а конкретно их непревосходство 100. Если введенное число больше 100, то данные некорректны, значит выполняются действия внутри условного оператора, идентичные действиям, внутри предыдущего условного оператора.

Последним действием внутри функции выполняется присваивание переменной *array[i]* значения 101 через *arr[i] (*ссылку на него внутри функции).

Во всех объектных файлах были использованы конструкции *#ifndef #define #endif,* защищающие от множественности подключения файла.

В файле *max.h* был описан прототип функции *int max(int arr[], int len)*, которая принимает на вход ссылку на массив и длину данного массива, а также возвращает значения типа *int.*

В файле *min.h* был описан прототип функции *int min(int arr[], int len)*, которая принимает на вход ссылку на массив и длину данного массива, а также возвращает значения типа *int.*

В файле *diff.h* был описан прототип функции *int diff(int arr[], int len)*, которая принимает на вход ссылку на массив и длину данного массива, а также возвращает значения типа *int.*

В файле *sum.h* был описан прототип функции *int sum(int arr[], int len)*, которая принимает на вход ссылку на массив и длину данного массива, а также возвращает значения типа *int.*

В файле *max.c* было представлено описание функции *int max(int arr[], int len)*. Первоначально в ней инициализируются переменные *maximum* типа *int*,

которой присваивается значение первого элемента массива *arr*, и *i* типа *int*. Далее идет цикл *for(i=0;i<len;i++),* проходящий по всему массива и ищущий максимальный элемент в нем. Это реализовано благодаря условному оператору *if (maximum<arr[i])*, который выполняется, если *maximum* меньше текущего элемента массива, в таком случае выполняется присвоение значению переменной *maximum* значения *arr[i]*-ого. После цикла выполняется возврат значения переменной *maximum*, выполненный благодаря оператору *return*.

В файле *min.c* было представлено описание функции *int min(int arr[], int len)*. Первоначально в ней инициализируются переменные *minimum* типа *int*, которой присваивается значение первого элемента массива *arr*, и *i* типа *int*. Далее идет цикл *for(i=0;i<len;i++),* проходящий по всему массива и ищущий минимальный элемент в нем. Это реализовано благодаря условному оператору *if (minimum>arr[i])*, который выполняется, если *minimum б*ольше текущего элемента массива, в таком случае выполняется присвоение значению переменной *minimum* значения *arr[i]*-ого. После цикла выполняется возврат значения переменной *minimum*, выполненный благодаря оператору *return*.

В файле *diff.c* было представлено описание функции *int diff(int arr[], int len)*. Первоначально в ней инициализируются переменные *maximum* типа *int*, которой присваивается значение, возвращаемое функцией *max* с аргументами *arr, len и minimum* типа *int*, которой присваивается значение, возвращаемое функцией *min* с аргументами *arr, len.* Далее инициализирована переменная diff типа int, которой присваивается значение равное разности значений переменных maximum и minimum. После выполняется возврат значения переменной *diff*, выполненный благодаря оператору *return*.

В файле *sum.c* было представлено описание функции *int sum(int arr[], int len)*. Первоначально в ней инициализируются переменные *minimum* типа *int*, которой присваивается значение первого элемента массива *arr*, переменная *sum*, которой присваивается значение 0 и *i* типа *int*. Далее идет цикл *while (arr[i]!=minimum)*, выполняющийся до момента, пока текущий элемент массива не станет равным минимальному элементу массива. Внутри цикла выполняется

прибавление к значению переменной *sum* значения *arr[i]* и инкремент *i*. После цикла выполняется возврат значения переменной *sum*, выполненный благодаря оператору *return*.

Далее был написан *Makefile*, отвечающий за сборку программы. Основной задачей является all, которая зависит от подзадач (файлов) *menu.o, max.o, min.o, diff.o, sum.o.* Внутри данной задачи выполняется линковка этих файлов в исполняемый (бинарный) файл, с названием *menu.*

Далее подзадача *max.o*, зависящая от файлов *max.c* и *max.h,* в которой выполняются такие процессы как препроцессинг, компиляция и ассамблирование данных файлов в объектный файл *max.o.*

Далее подзадача *min.o*, зависящая от файлов *min.c* и *min.h,* в которой выполняются такие процессы как препроцессинг, компиляция и ассамблирование данных файлов в объектный файл *min.o.*

Далее подзадача *diff.o*, зависящая от файлов *diff.c, diff.h, min.h, max.h,* в которой выполняются такие процессы как препроцессинг, компиляция и ассамблирование данных файлов в объектный файл *diff.o.*

Далее подзадача *sum.o*, зависящая от файлов *sum.c, sum.h* и *min.h,* в которой выполняются такие процессы как препроцессинг, компиляция и ассамблирование данных файлов в объектный файл *sum.o.*

Далее подзадача *menu.o*, зависящая от файлов *menu.c, scan_array.h, min.h, max.h, diff.h* и *sum.h,* в которой выполняются такие процессы как препроцессинг, компиляция и ассамблирование данных файлов в объектный файл *menu.o.*

Последней подзадачей является *clean*, которую нужно вызывать отдельно при помощи команды *make clean*. Она удаляет из директории все объектные файлы.

Собрать данную программу можно при помощи команды *make* и запустить командой *./menu*.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	93	Верно
2.	1 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	0	Верно
3.	2 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	93	Верно
4	3 6 92 70 59 63 90 38 57 29 0 47 81 5 48 93 29 69 93 56 23 59 25 33 91 77 71 34 85 62 41 84 66 8 41 25 89 13 68	504	Верно

Выводы.

Был изучен процесс сборки программ, состоящий из нескольких файлов, с исходным кодом на языке Си. Была разработана программа и Makefile для её сборки.

Каждая функция была определена в отдельном файле, так же каждый прототип функции хранится в отдельном заголовочном файле.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

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

int main(){
	int array[102];
	int n;
	scanf("%d",&n);
	scan_array(array);
	int len=0;
	int i;
	for (i=0;i<101;i++){
		if (array[i]==101){
			len=i;
			break;
		}
	}
	if (array[101]!=102){
		switch(n){
			case 0: ;
				int maximum;
				maximum=max(array, len);
				printf("%d\n",maximum);
				break;
			case 1: ;
				int minimum;
				minimum=min(array, len);
				printf("%d\n",minimum);
				break;
			case 2: ;
				int difference;
```

```
                        difference=diff(array, len);
                        printf("%d\n",difference);
                        break;
                case 3: ;
                        int summ;
                        summ=sum(array,len);
                        printf("%d\n",summ);
                        break;
                default:
                        printf("Данные некорректны\n");
            }
        }else{
            printf("Данные некоректны\n");
        }
}
```

Название файла: max.c

```
#include <stdio.h>
#include "max.h"

int max(int arr[], int len){
        int maximum=arr[0];
        int i;
        for(i=0;i<len;i++){
            if (maximum<arr[i]){
                maximum=arr[i];
            }
        }
    return maximum;
}
```

Название файла: min.c

```
#include <stdio.h>
#include "min.h"

int min(int arr[], int len){
        int minimum=arr[0];
        int i;
```

```
        for (i=0;i<len;i++){
                if (minimum>arr[i]){
                        minimum=arr[i];
                }
        }
        return minimum;
}
```

Название файла: diff.c

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

int diff(int arr[], int len){
        int maximum=max(arr,len);
        int minimum=min(arr,len);
        int diff=maximum-minimum;
        return diff;
}
```

Название файла: sum.c

```
#include <stdio.h>
#include "sum.h"
#include "min.h"

int sum(int arr[],int len){
        int minimum=min(arr,len);
        int sum=0;
        int i=0;
        while (arr[i]!=minimum){
                sum+=arr[i];
                i++;
        }
        return sum;
}
```

Название файла: max.h

```
#ifndef MAX_H
#define MAX_H

#include <stdio.h>

int max(int arr[], int len);

#endif
```

Название файла: min.h

```
#ifndef MIN_H
#define MIN_H

#include <stdio.h>

int min(int arr[], int len);

#endif
```

Название файла: diff.h

```
#ifndef DIFF_H
#define DIFF_H

#include <stdio.h>

int diff(int arr[], int len);

#endif
```

Название файла: sum.h

```
#ifndef SUM_H
#define SUM_H

#include <stdio.h>

int sum(int arr[],int len);
```

```
#endif
```

Название файла: scan_array.h

```
#ifndef SCAN_ARRAY_H
#define SCAN_ARRAY_H

#include <stdio.h>

void scan_array(int arr[]){
	char symbol;
	int i=0;
	int proverka=0;
	do{
		if (scanf("%d%c",&arr[i],&symbol)==0){
			proverka=1;
			arr[101]=102;
		}
		if (arr[i]>100){
			proverka=1;
			arr[101]=102;
		}
		i++;
	}while(symbol!='\n' && proverka==0);
	arr[i]=101;
}

#endif
```

Название файла: Makefile

```
all: menu.o max.o min.o diff.o sum.o
	gcc menu.o max.o min.o diff.o sum.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 max.h min.h
	gcc -c diff.c
```

```
sum.o: sum.c sum.h min.h
	gcc -c sum.c
menu.o: menu.c scan_array.h min.h max.h diff.h sum.h
	gcc -c menu.c
clean:
	rm *.o
```