МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Математического Обеспечения и Применения ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Программирование»
Тема: Сборка программ в Си

Студент гр. 0382 _____ Кондратов Ю.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

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

Задание.

Создать проект функции-меню с использованием make-файла.

На вход программе подаётся 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 (функция max);

1: минимальное число в массиве (функция min);

2: разницу между максимальным и минимальным элементом (функция diff);

3: сумму элементов массива, расположенных до первого минимального элемента (функция sum);

иначе необходимо вывести строку «Данные некорректны».

Определение каждой функции должно находиться в отдельном файле.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.

В данной работе были использованы такие конструкции языка Си как:

- Функции стандартной библиотеки ввода-вывода:
  - *printf()* - выводит принимаемые значения на консоль;
  - *scanf()* - считывает входные данные из консоли;
- Операторы:
  - i*f(){}* - если выражение в круглых скобках верно, выполняет блок кода в фигурных скобках;

- *switch(){case x: ; default:}* - в зависимости от значения переменной в круглых скобках, выполняет блок когда, находящий после «*case x:*», где x — значение переменной в круглых скобках. Если x не соответствует ни одному *case*, то выполняет блок кода, находящийся после «*default:*».

- Циклы:
  - *while(){}* - на каждой итерации проверяется выражение в круглых скобках, если оно верно — выполняется блок кода в фигурных скобках, иначе — производится выход из цикла;
  - *for(<переменная>, <выражение 1>, <выражение 2>){}* - первым аргументом является переменная цикла, далее, если верно выражение 1 — выполняется блок кода в фигурных скобках и выражение 2, которое зачастую связано с переменной цикла;
- Пользовательские функции:
  - *<тип_возвращаемого_значения> имя_функции (список_параметров _функции) {return <возвращаемое_значение>;}* - при вызове в функции *main* выполняет блок кода в фигурных скобках, используя переданные параметры, и возвращает значение после оператора *return* (если тип возвращаемого значение не *void*).

Также был использован make-файл, который состоит из:

- списка целей;

- зависимостей этих целей;

- команд, которые требуется выполнить, чтобы достичь эту цель.

Содержимое должно выглядеть следующим образом:

цель: зависимости

[tab] команда

Первая цель в файле является целью по умолчанию. Для сборки проекта обычно используется цель all, которая находится самой первой.

Выполнение работы.

Для решения поставленных задач необходимо считать данные, обработать их и вывести результат на консоль.

Считывание и вывод данных производится в основном файле *menu.c.*

Для считывания входных данных используются переменные:

- *option* типа *int* — в этой переменной хранится значение управляющего символа (0, 1, 2 или 3);
- *array* массив типа i*nt* размера 100 элементов — массив, предназначенный для хранения массива целых чисел, введённых пользователем;
- *index* типа *int* с начальным значением ноль — переменная, хранящая текущее значения индекса нового элемента массива;
- *c* типа *char* — переменная, в которой хранится символ, введённый после числа.

Далее с помощью функции *scanf* в переменную *option* считывается управляющее значение, после чего с помощью цикла *while,* в каждой итерации которого проверяются условия: i*ndex < 100* и *c != ' ',* и функцией *scanf* считывается очередной целочисленный элемент массива и следующий за ним символ, также значение переменной *index* увеличивается на 1 при помощи постфиксного инкремента.

При помощи оператора *switch,* в зависимости от значения переменной *option,* функцией *printf* выводится на консоль:

- значение функции *max* если *option == 0*;
- значение функции *min* если *option == 1*;
- значение функции *diff* если *option == 2*;
- значение функции *sum* если *option == 3*;

- строка «Данные некорректны» если *option* имеет другое значение.

Описание используемых функций:

1. Функция *int max (int ar[], int len).*

Определение функции находится в файле *max.c,* объявление — в файле *max.h.*

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len,* хранящую длину массива. В целочисленную переменную *ans* записывается значение элемента массива с индексом 0 в качестве начального максимума.

Далее с помощью цикла *for* все элементы массива с индексами от 1 до значения длины массива проверяются оператором if на соответствие условию *ar[i] > ans.* Если условие верно, то значение предыдущего максимума, записанное в переменной *ans* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден максимальный элемент массива.

С помощью оператора *return* функцией *main* будет возвращено значение элемента *ans.*

2. Функция *int min (int ar[], int len).*

Определение функции находится в файле *min.c,* объявление — в файле *min.h.*

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len,* хранящую длину массива. В целочисленную переменную *ans* записывается значение элемента массива с индексом 0 в качестве начального минимума.

Далее с помощью цикла *for* все элементы массива с индексами от 1 до значения длины массива проверяются оператором if на соответствие условию

ar[i] < *ans*. Если условие верно, то значение предыдущего минимума, записанное в переменной *ans* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден минимальный элемент массива.

С помощью оператора *return* функцией *main* будет возвращено значение элемента *ans*.

3. Функция *int diff (int ar[], int len)*.

Определение функции находится в файле *diff.c,* объявление — в файле *diff.h.* Также, в связи с тем, что функция *diff* использует функции *max* и *min*, в файл *diff.c* при помощи *#include* включены заголовочные файлы *max.h* и *min.h*

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len,* хранящую длину массива.

Далее с помощью функции *max* находится максимальное значение элемента массива, а с помощью функции *min* — минимальное.

Функция возвращает разность значения *max* и значения *min.*

4. Функция *int sum (int ar[], int len)*.

Определение функции находится в файле *sum.c,* объявление — в файле sum.*h.* Также, в связи с тем, что функция *sum* использует функцию *min*, в файл *sum.c* при помощи *#include* включен заголовочный файл *min.h*

В качестве аргументов принимает целочисленный массив *ar* и целочисленную переменную *len,* хранящую длину массива. В целочисленную переменную *ans* записывается значение 0 в качестве начального значения суммы.

Далее с помощью цикла *for*, в каждой итерации которого к целочисленной переменной i, начальное значение которой равно 0, прибавляется единица, и, пока выполнено условие *ar[i] > min(ar, len)* к переменной *ans* прибавляется значение i-го элемента массива, находится

сумма всех элементов массива до первого минимального и записывается в переменную *ans,* значение которой возвращает функция.

5. Makefile

В make-файле предназначен для сборки проекта и создания исполняемого файла *menu.* В нём содержатся следующие инструкции:

5.1 Инструкция all.

Имеет следующие зависимости: *max.o, min.o, diff.o, sum.o, menu.o.*

Команда: gcc max.o min.o diff.o sum.o menu.o -o menu.

Выполнение данной инструкции приводит к сборке проекта из необходимых объектных файлов.

С помощью ключа «-o» сообщается название получаемого после выполнения исполняемого файла — *menu.*

5.2 Инструкция *menu.o.*

Имеет следующие зависимости: *menu.c, max.h, min.h, diff.h, sum.h.*

Команда: *gcc -c menu.c -std=c99.*

Выполнение данной инструкции приводит к созданию объектного файла *menu.o.*

Ключ *-std=c99* здесь и далее используется для сообщения компилятору стандарта, по которому написан код, во избежание ошибок компиляции.

5.3 Инструкция *max.o.*

Имеет следующие зависимости: *max.c, max.h.*

Команда: *gcc -c max.c -std=c99.*

Выполнение данной инструкции приводит к созданию объектного файла *max.o.*

5.4 Инструкция *min.o.*

Имеет следующие зависимости: *min.c, min.h.*

Команда: *gcc -c min.c -std=c99.*

Выполнение данной инструкции приводит к созданию объектного файла *min.o.*

5.5 Инструкция diff.*o.*

Имеет следующие зависимости: *diff.c diff.h, max.h, min.h.*

Команда: *gcc -c diff.c -std=c99.*

Выполнение данной инструкции приводит к созданию объектного файла *diff.o.*

5.6 Инструкция *sum.o.*

Имеет следующие зависимости: sum.*c, sum.h, min.h.*

Команда: *gcc -c sum.c -std=c99.*

Выполнение данной инструкции приводит к созданию объектного файла *sum.o.*

5.7 Инструкция *clean.*

Используется для удаления всех объектных файлов из текущей директории.

Команда: *rm *.o.*

Исходный код всех файлов см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 34 56 23 56 21 23 100\n	100	Программа работает правильно
2.	0 5 5 5 5 5 5\n	5	Программа работает правильно
3.	1 34 56 73 45 73 -100 100\n	-100	Программа работает правильно
4.	1 5 5 5 5 5\n	5	Программа работает правильно
5.	2 -100 2 3 4 5 6 100\n	200	Программа работает правильно
6.	2 5 5 5 5 5\n	0	Программа работает правильно
7.	3 1 1 1 1 0 1 1 0\n	4	Программа работает правильно
8.	3 1 1 1 1 1 1 1 1\n	0	Программа работает правильно

Выводы.

В ходе работы был изучен процесс сборки программ на языке Си при помощи утилиты Make.

Разработана программа, выполняющая считывание исходных с помощью функции scanf() и цикла *while(){}* в переменную *option* и массив *array[100],* условием которого было равенство переменной *c*, хранящей код символа между числами, коду символа пробела, написаны функции для обработки входных результатов, подробное описание которых приведено в разделе «выполнение работы», с помощью оператора *switch(){}* и функции

printf() реализован вывод реузльтата определённой функции в зависимости от входного управляющего значения *option*:

- если *option = 0* — выводится результат функции *int max()*;
- если *option = 1* — выводится результат функции *int min()*;
- если *option = 2* — выводится результат функции *int diff()*;
- если *option = 3* — выводится результат функции *int sum()*;

Если значение *option* не соответствует ни одному из перечисленных — выводится строка «Данные некорректны».

Все функции хранятся в отдельных файлах. Для каждой функции создан файл с расширением *.c, в котором хранится определение функции и заголовочный файл, в котором находится объявление функции.

Основная функция *main* находится в файле *menu.c.*

Разработан make-файл, в котором расположены инструкции по сборке программы, указаны зависимости этих инструкций и команды, которые необходимо выполнить. Результатом работы утилиты Make является исполняемый файл *menu.*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ФАЙЛОВ ПРОЕКТА

1. Название файла: menu.c

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

int main(){

    int array[100], index = 0, option;
    char c = ' ';

    scanf("%d", &option);

    while(index < 100 && c == ' '){
        scanf("%d%c", &array[index++], &c);
    }

     switch (option){
        case 0:
            printf("%d\n", max(array, index));
            break;
        case 1:
            printf("%d\n", min(array, index));
            break;
        case 2:
            printf("%d\n", diff(array, index));
            break;
        case 3:
            printf("%d\n", sum(array, index));
            break;
        default:
            printf("Данные некорректны\n");
    }

    return 0;
}
```

2. Название файла: max.c

```
#include "max.h"
int max(int ar[], int len){
    int ans = ar[0];
    for (int i = 1; i < len; i++){
        if (ar[i] > ans){
           ans = ar[i];
        }
    }
    return ans;
}
```

3. Название файла: min.c

```
#include "min.h"
int min(int ar[], int len){
    int res = ar[0];
    for (int j = 1; j < len; j++){
        if (ar[j] < res){
           res = ar[j];
        }
    }
    return res;
}
```

4. Название файла: diff.c

```
#include "diff.h"
#include "max.h"
#include "min.h"
int diff(int ar[], int len){
    int ans = max(ar, len) - min(ar, len);
    return ans;
}
```

5. Название файла: sum.c

```
#include "sum.h"
#include "min.h"
int sum(int ar[], int len){
    int ans = 0;
    for(int i = 0; ar[i] != min(ar, len); i++){
            ans += ar[i];
    }
    return ans;
}
```

6. Название файла: max.h

```
int max();
```

7. Название файла: min.h

```
int min();
```

8. Название файла: diff.h

```
int diff();
```

9. Название файла: sum.h

```
int sum();
```

10. Название файла: Makefile

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