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: Создание make-файла

Студентка гр. 9304 _____ Каменская Е.К.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2019

Цель работы.

Разделить код программы на несколько файлов в соответствии с количеством созданных функций, создать соответствующие заголовочные файлы и make-файл для сборки исполняемого файла.

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 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.

make — утилита, автоматизирующая процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки.

Утилита использует специальные make-файлы, в которых указаны зависимости файлов друг от друга и правила для их удовлетворения. На основе информации о времени последнего изменения каждого файла make определяет и запускает необходимые программы. Запуск утилиты осуществляется командой *make*. По умолчанию утилита выполнит файл с именем Makefile

(makefile), но можно выполнить и конкретный файл указав *make –f <имя_make-файла>*.

Любой make-файл состоит из

- списка целей
- зависимостей этих целей
- команд, которые требуется выполнить, чтобы достичь эту цель

Записанных в форме

цель: зависимости

[tab] команда

Целью может являться имя генерируемого файла (например *main.o*) или название действия (например *clean*). Для сборки проекта обычно используется цель all, которая находится самой первой и является целью по умолчанию.

Также в make-файле можно создавать переменные:

<имя_переменной> = <значение>

Для вызова переменной требуется указать $(<имя_переменной>).

Выполнение работы.

Переменные:

- *arr[]* – массив вводимых целых чисел. Начальный размер *arr[]* равен 100 (максимальное возможное количество элементов по условию).
- *real_n* – целочисленная переменная для хранения действительного количества введенных элементов.
- *option* – целочисленная переменная для хранения значения, от которого зависит вывод программы.
- *c* – символьная переменная, считывающая символ, следующий за введенным.

Функции:

- *int abs_max(int *arr, int real_n)* принимает на вход указатель на первый элемент массива *arr[]* и целочисленное количество элементов в нем.

Внутри функции создается целочисленная переменная *a_max*, инициализируемая как модуль первого элемента массива *arr[]*, для записи максимального модуля и целочисленная переменная *elem* для записи искомого элемента.

С помощью цикла *for* и оператора *if* функция сравнивает модуль каждого элемента с *a_max* и присваивает *elem* значение элемента с максимальным модулем. В результате возвращается значение *elem*.

- *int abs_min(int *arr, int real_n)* принимает на вход указатель на первый элемент массива *arr[]* и целочисленное количество элементов в нем.

  Внутри функции создается целочисленная переменная *a_min*, инициализируемая как модуль первого элемента массива *arr[]*, для записи наименьшего модуля и целочисленная переменная *elem* для записи искомого элемента.

  С помощью цикла *for* и оператора *if* функция сравнивает модуль каждого элемента с *a_min* и присваивает *elem* значение элемента с минимальным модулем. В результате возвращается значение *elem*.
- *int diff(int *arr, int real_n)* принимает на вход указатель на первый элемент массива *arr[]* и целочисленное количество элементов в нем.

  Внутри функции создается целочисленная переменная *e_max*, инициализируемая как результат работы функции *abs_max*, для записи числа с максимальным модулем и целочисленная переменная *e_min*, инициализируемая как результат работы функции *abs_min*, для записи числа с минимальным модулем.

  На выходе возвращается значение разности *e_max* и *e_min*.
- *int sum(int *arr, int real_n)* принимает на вход указатель на первый элемент массива *arr[]* и целочисленное количество элементов в нем.

Внутри функции создается целочисленная переменная *e_max*, инициализируемая как результат работы функции *abs_max*, для записи числа с максимальным модулем и целочисленная переменная *answer*, инициализируемая нулем, для записи суммы.

С помощью цикла *for* и оператора *if* начиная с элемента, равного *e_max*, все элементы прибавляются к *answer*:

```
for(int i=0; i<real_n; i++){
          if((arr[i]==e_max)||(e_max==0)){
               answer += arr[i];
               e_max = 0;
          }
}
```

В итоге возвращается переменная *answer*.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	2 -90 16 77 2 34	-92	Найдена разница между максимальным по модулю и минимальным по модулю элементом.
2.	3 0 0 -2 0	-2	Сумма всех элементов, начиная с максимального по модулю.
3.	1 999 -1600 -3400 -2001	999	Минимальный по модулю элемент

Выводы.

Код программы был разделен по файлам в соответствии с реализованными функциями, были написаны заголовочные файлы и make-файл для сборки программы. В заголовочных файлах была использована директива

#pragma once для избегания повторного подключения на этапе работы препроцессора. В make-файле были созданы переменные *CC* и *FLAG* для облегчения задания параметров компиляции файлов, в частности переменная *FLAG* используется для задания стандарта C99 языка C.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#include <stdio.h>
#define N 100
#include "abs_max.h"
#include "abs_min.h"
#include "diff.h"
#include "sum.h"

int main(){
	int arr[N];
	int count = 0;
	int option;
	char c;

	scanf("%d", &option);
	for(int i=0; i<N; i++){
		scanf("%d%c", &arr[i], &c);
		count++;
		if(c=='\n'){
			break;
		}
	}
	switch (option){
		case 0:
			printf("%d\n", abs_max(arr, count));
		break;
		case 1:
			printf("%d\n", abs_min(arr, count));
		break;
		case 2:
			printf("%d\n", diff(arr, count));
		break;
		case 3:
			printf("%d\n", sum(arr, count));
		break;
		default:
```

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

Название файла: abs_max.c

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

int abs_max(int *arr, int count){
        int a_max = abs(arr[0]);
        int elem = arr[0];
        for(int i=0; i<count; i++){
                if(abs(arr[i])>a_max){
                        a_max = abs(arr[i]);
                        elem = arr[i];
                }
        }
        return elem;
}
```

Название файла: abs_max.h

```
#pragma once
int abs_max(int *arr, int count);
```

Название файла: abs_min.c

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

int abs_min(int *arr, int count){
        int a_min = abs(arr[0]);
        int elem = arr[0];
        for(int i=0; i<count; i++){
                if(abs(arr[i])<a_min){
                        a_min = abs(arr[i]);
                        elem = arr[i];
                }
        }
        return elem;
}
```

Название файла: abs_min.h

```
#pragma once

int abs_min(int *arr, int count);
```

Название файла: sum.c

```
#include "sum.h"
#include "abs_max.h"

int sum(int *arr, int count){
	int e_max = abs_max(arr, count);
	int answer = 0;
	for(int i=0;i<count;i++){
		if((arr[i]==e_max)||(e_max==0)){
			answer += arr[i];
			e_max = 0;
		}
	}
	return answer;
}
```

Название файла: sum.h

```
#pragma once

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

Название файла: diff.c

```
#include "diff.h"
#include "abs_max.h"
#include "abs_min.h"

int diff(int *arr, int count){
	int e_max = abs_max(arr, count);
	int e_min = abs_min(arr, count);
	return e_max-e_min;
}
```

Название файла: diff.h

```
#pragma once

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

Название файла: Makefile

```
CC=gcc
FLAG=-std=c99

all: menu

menu: menu.o abs_max.o abs_min.o sum.o diff.o
	$(CC) $(FLAG) menu.o abs_max.o abs_min.o sum.o diff.o -o menu

menu.o: menu.c abs_max.h abs_min.h sum.h diff.h
	$(CC) $(FLAG) -c menu.c

abs_max.o: abs_max.c abs_max.h
	$(CC) $(FLAG) -c abs_max.c

abs_min.o: abs_min.c abs_min.h
	$(CC) $(FLAG) -c abs_min.c

sum.o: sum.c sum.h abs_max.h
	$(CC) $(FLAG) -c sum.c

diff.o: diff.c abs_max.h abs_min.h
	$(CC) $(FLAG) -c diff.c

clean:
	rm -f *.o menu
```