**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 |
|  | 2 -90 16 77 2 34 | -92 | Найдена разница между максимальным по модулю и минимальным по модулю элементом. |
|  | 3 0 0 -2 0 | -2 | Сумма всех элементов, начиная с максимального по модулю. |
|  | 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 С99 языка С.

# 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