

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

Осинкин Е.А.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

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

Задание.

Вариант 2.

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:
 - `if(){} - 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, находящийся 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 значения 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:

- *command* типа *int* — в этой переменной хранится значение управляющего символа (0, 1, 2 или 3);
- *array* массив типа *int* размера 100 элементов — массив, предназначенный для хранения массива целых чисел, введенных пользователем;
- *size* типа *int* с начальным значением ноль — переменная, хранящая текущее значения индекса нового элемента массива;

Далее с помощью функции *scanf* в переменную *command* считывается управляющее значение, после чего с помощью цикла *while*, в каждой итерации которого проверяется условие: *getchar() != '\n'*, и функцией *scanf* считывается очередной целочисленный элемент массива и следующий за ним символ, также значение переменной *size* увеличивается на 1 при помощи постфиксного инкремента.

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

- значение функции *max* если *command == 0*;
- значение функции *min* если *command == 1*;
- значение функции *diff* если *command == 2*;
- значение функции *sum* если *command == 3*;
- строка «Данные некорректны» если *command* имеет другое значение.

Описание используемых функций:

1. Функция *int max(int array[], int size)*.

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

В качестве аргументов принимает целочисленный массив *array* и целочисленную переменную *size*, хранящую длину массива. Если длина массива отрицательная или 0, то возвращаем 0 с помощью оператора *return*. В целочисленную переменную *max* 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 *max < array[i]*. Если условие верно, то значение предыдущего максимума, записанное в переменной *max* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден максимальный элемент массива.

С помощью оператора *return* возвращаем значение элемента *max*.

2. Функция *int min(int array[], int size)*.

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

В качестве аргументов принимает целочисленный массив *array* и целочисленную переменную *size*, хранящую длину массива. Если длина массива отрицательная или 0, то возвращаем 0 с помощью оператора *return*. В целочисленную переменную *min* 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 *min > array[i]*. Если условие верно, то значение предыдущего минимума,

записанное в переменной *min* меняется на значение текущего элемента массива. Таким образом, после всех итераций будет найден минимальный элемент массива.

С помощью оператора *return* возвращаем значение элемента *min*.

3. Функция *int diff(int array[], int size)*.

В качестве аргументов принимает целочисленный массив *array* и целочисленную переменную *size*, хранящую длину массива. Если длина массива отрицательная или 0, то возвращаем 0 с помощью оператора *return*.

Далее в переменную *result*, с помощью функции *max* и *min*, записываем разницу между максимальным и минимальным элементом массива.

С помощью оператора *return* возвращаем значение элемента *result*.

4. Функция *int sum (int array[], int size)*.

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 *array* и целочисленную переменную *size*, хранящую длину массива. Если длина массива отрицательная или 0, то возвращаем 0 с помощью оператора *return*. В целочисленную переменную *minimum* присваиваем минимальный элемент массива с помощью функции *min*. В целочисленную переменную *sum* записывается значение 0 в качестве начального значения суммы.

Далее с помощью цикла *for* все элементы массива с индексами от 0 до значения длины массива проверяются оператором *if* на соответствие условию *array[i] == minimum*. Если условие верно, то прекращаем данный цикл оператором *break*. После этого условия в теле цикла к *sum* прибавляем *array[i]*. Таким образом, после всех итераций будет найдена сумма элементов до первого минимального элемента массива.

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*.

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.	0 1 2 3 4 5 6 7\n	7	Программа работает правильно
2.	1 1 2 3 4 5 6 7\n	1	Программа работает правильно
3.	2 1 2 3 4 5 6 7\n	6	Программа работает правильно
4.	3 1 2 3 4 5 6 7\n	0	Программа работает правильно
5.	4 1 2 3 4 5 6 7\n	Данные некорректны	Программа работает правильно
6.	3 5462 1234 367 2346\n	6696	Программа работает правильно
7.	2 5489 2123 534 785\n	4955	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 *command* и массив *array[100]*, написаны функции для обработки входных результатов, подробное описание которых приведено в разделе «выполнение работы», с помощью оператора `switch(){}` и функции `printf()` реализован вывод результата определённой функции в зависимости от входного управляющего значения *command*:

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

Если значение *command* 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 "min.h"
#include "max.h"
#include "diff.h"
#include "sum.h"

int main() {
    int command;
    int array[100];
    int size = 0;
    scanf("%d", &command);
    while (getchar() != '\n') {
        scanf("%d", &array[size]);
        size++;
    }
    switch (command) {
        case 0:
            printf("%d\n", max(array, size));
            break;
        case 1:
            printf("%d\n", min(array, size));
            break;
        case 2:
            printf("%d\n", diff(array, size));
            break;
        case 3:
            printf("%d\n", sum(array, size));
            break;
        default:
            printf("Данные некорректны\n");
            break;
    }
    return 0;
}
```

2. Название файла: max.c

```
#include "max.h"

int max(int array[], int size) {
    if (size <= 0) {
        return 0;
    }
    int max = array[0];
    for (int i = 1; i < size; i++) {
        if (max < array[i]) {
            max = array[i];
        }
    }
    return max;
}
```

3. Название файла: min.c

```
#include "min.h"

int min(int array[], int size) {
    if (size <= 0) {
        return 0;
    }
    int min = array[0];
    for (int i = 1; i < size; i++) {
        if (min > array[i]) {
            min = array[i];
        }
    }
    return min;
}
```

4. Название файла: diff.c

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

int diff(int array[], int size) {
    if (size <= 0) {
        return 0;
    }
    int result;
    result = max(array, size) - min(array, size);
    return result;
}
```

5. Название файла: sum.c

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

int sum(int array[], int size) {
    if (size <= 0) {
        return 0;
    }
    int minimum = min(array, size);
    int sum = 0;
    for (int i = 0; i < size; i++) {
        if (array[i] == minimum) {
            break;
        }
        sum = sum + array[i];
    }
    return sum;
}
```

6. Название файла: max.h

```
int max(int array[], int size);
```

7. Название файла: min.h

```
int min(int array[], int size);
```

8. Название файла: diff.h

```
int diff(int array[], int size);
```

9. Название файла: sum.h

```
int sum(int array[], int size);
```

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 *.
```