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 Си

Студент гр. 0382 Литягин С.М.

Преподаватель Чайка К.В., Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение сборки программ в Си с помощью утилиты

Make.

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.

В программе использовались следующие управляющие конструкции языка:

Функции библиотеки **stdio.h**:

- **printf()** – функция вывода на консоль;
- **scanf()** – функция ввода данных из консоли. Функция библиотеки **stdlib.h**:
- **abs()** – функция получения модуля числа. Циклы:
- **while(){}** – каждая итерация проверяет, выполняется ли условие в круглых скобках, если оно верно, то выполняется код в фигурных скобках, а если неверно, то происходит выход из цикла;

- **for(){<переменная>; <условие>; <выражение_1>}** – код в теле цикла будет исполнятся до тех пор, пока объявленная в цикле переменная будет удовлетворять условию цикла, выражение_1 каким-либо способом меняет значение этой переменной.

Операторы:

- **if(){} … else{}** – если выполняется условия, указанное в круглых скобках, то выполняется код в фигурных скобках после if, иначе – в фигурных скобках после else (else не является обязательной частью конструкции)
- **switch(<переменная>){case x:… break; … default:…break;}** – от значения переменной в круглых скобках зависит, какой кейс будет выполняться (например, если переменная имеет значение x – выполнится case x). Если же не будет кейса с таким значением, то выполнится код из блока default.

Функции:

- **<тип_функции> имя_функции(<аргумент_1>, … , <argument_n>) {}** – при вызове данной функции в главной(main) функции выполняется код в фигурных скобках, а затем возвращает значение оператором return (если тип функции не void)

Также использовался make-file:

Любой make-файл состоит из

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

цель: зависимости
[tab] команда
Для сборки проекта обычно используется цель all, которая находится самой первой и является целью по умолчанию (фактически, первая цель в файле и является целью по умолчанию).

Выполнение работы.

Решение задачи заключается в считывании данных, их обработке и выводе результата.

Чтобы считать данные, были созданы следующие переменные:

1. Массив **numbers** типа int, в котором хранятся вводимые целые числа;
2. Переменная **value** типа int. В ней хранится значение (0, 1, 2 или 3), от которого зависит способ обработка целых чисел массива;
3. Переменная **index** типа int, которая считает, сколько чисел ввели в массив.

Также для созданных функций были сделаны следующие переменные:

1. Переменная *max_n* типа int для хранения максимального по модулю числа (элемента массива);
2. Переменная *min_n* типа int для хранения минимального по модулю числа (элемента массива);
3. Переменная *different* типа int для хранения разницы между максимального по модулю числа (элемент массива) и минимального по модулю числа (элемент массива).
4. Переменная *summa* типа int для хранения суммы чисел, расположенных после максимального по модулю числа (элемент массива), включая его;

Описание библиотек:

1. **stdio.h** – используется для подключения ввода-вывода (printf(), scanf());
2. **stdlib.h** – используются для доступа к функции abs(), которая позволяет получить модуль числа.

Для реализации программы были созданы следующие функции:

1. Функция *int abs_max(int numbers[], int index).*

Определение функции находится в файле *abs_max.c*, а объявление – в файле *abs_max.h*.

Она принимает целочисленный массив *numbers*, целочисленную переменную *index*.

Цель этой функции – найти максимальный по модулю элемент массива и его индекс. Для этого переменной *max_n* присваивается первый элемент массива *numbers[0]*. Далее в цикле for все элементы массива с индексами от 0 до значения переменной *index* сравниваются оператором if на удовлетворение условию abs(*max_n*) < abs(*numbers[k]*) (*k* – счетчик цикла). Если условие выполняется, то переменная *max_n* принимает значение элемента *numbers[k]*.

В результате мы получаем максимальный по модулю элемент массива и его индекс. Функция возвращает значение переменной *max_n* с помощью оператора return.

2. Функция *int abs_min(int numbers[], int index)*.

Определение функции находится в файле *abs_min.c*, а объявление – в файле *abs_min.h*.

Она принимает целочисленный массив *numbers*, целочисленную переменную *index*.

Цель этой функции – найти минимальный по модулю элемент массива. Для этого переменной *min_n* присваивается первый элемент массива *numbers[0]*. Далее в цикле for все элементы массива с индексами от 0 до значения переменной *index* сравниваются оператором if на удовлетворение условию abs(*min_n*) > abs(*numbers[k]*) (*k* – счетчик цикла). Если условие выполняется, то переменная *min_n* принимает значение элемента *numbers[k]*.

В результате мы получаем минимальный по модулю элемент массива. Функция возвращает значение переменной *min_n* с помощью оператора return.

3. Функция *int diff(int numbers[], int index)*.

Определение функции находится в файле *diff.c*, а объявление – в файле *diff.h*. Также, поскольку здесь используются функции *abs_max()* и *abs_min()*, то в файл *diff.c* включены заголовочные файлы *abs_max.h* и *abs_min.h*.

Она принимает целочисленный массив *numbers*, целочисленную переменную *index*.

Цель этой функции – найти разницу между максимальным по модулю элементом массива и минимальным по модулю элементом массива. Для этого переменной *different* присваивается разность значений, полученных функциями *abs_max()* и *abs_min()*.

В результате функция возвращает значение переменной *different* с помощью оператора return.

4. Функция *int sum(int numbers[], int index)*.

   Определение функции находится в файле *sum.c*, а объявление – в файле *sum.h*. Также, поскольку здесь используется функция *abs_max()*, то в файл *sum.c* включен заголовочный файлы *abs_max.h*.

   Она принимает целочисленный массив *numbers*, целочисленную переменную *index*.

   Цель этой функции – найти сумму всех элементов массива, расположенных после максимального по модулю элемента массива, включая его. В самом начале функции переменная *numb* получает значение из функции *abs_max()*. Затем с помощью цикла for находится *numbers[j]* (*j* - счетчик), равный *numb* и ключу *key* присваивается значение 1. После этого переменная *summa* возрастает на величину *numbers[j]* каждую последующую итерацию.

   В результате функция выдает сумму всех элементов массива, расположенных после максимального по модулю элемента массива, включая его. Функция возвращает значение переменной *summa* с помощью оператора return.

Также создан *Makefile*. В данном файле прописываются команды для

компиляции программы. С помощью утилиты *make* собирается программа *menu*.

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 -2 3 -6 50 -250\n	-250	Программа работает правильно
2	1 100 -101 50 -5 3\n	3	Программа работает правильно
3	3 250 100 1000 50 349\n	1399	Программа работает правильно
4	2 500 2 3 4 -1\n	501	Программа работает правильно
5	3 -5 20 100\n	100	Программа работает правильно

Выводы.

Был изучен способ сборки программ в Си с помощью утилиты Make. А также разработана программа, выполняющая считывание с клавиатуры исходных данных, которая собирается из нескольких файлов с помощью *Makefile* и утилиты *make*. Сначала программа считывает значение переменной *value*. Далее с помощью цикла while считываются целые числа, которые заполняют массив *numbers*, а также считается количество этих чисел в переменной *index*, до тех пор, пока не будет получен символ перевода строки.

Далее, с помощью оператора switch(){}, определяется дальнейшая обработка чисел, введенных в массив. Это зависит от значения,

присвоенного переменной *value*:

Если *value* = 0, то выводится результат функции abs_max().

Если *value* = 1, то выводится результат функции abs_min().

Если *value* = 2, то выводится результат функции diff().

Если *value* = 3, то выводится результат функции sum().

Если же *value* имеет какое-либо другое значение, программа выведет фразу “Данные некорректны”.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#include <stdio.h>
#include <stdlib.h>
#include "abs_max.h"
#include "abs_min.h"
#include "diff.h"
#include "sum.h"

int main()
{
    int numbers[100];
    int index = 0;
    int value;

    scanf("%d", &value);

    while(getchar()!='\n'){
        scanf("%d", &numbers[index]);
        index++;
    }

    switch(value){
        case 0:
          printf("%d\n", abs_max(numbers, index));
          break;
        case 1:
          printf("%d\n", abs_min(numbers, index));
          break;
        case 2:
          printf("%d\n", diff(numbers, index));
          break;
        case 3:
          printf("%d\n", sum(numbers, index));
          break;
        default:
          printf("Данные некорректны\n");
          break;
    }
   return 0;
}
```

Название файла: abs_max.c

```
#include "abs_max.h"

int abs_max(int numbers[], int index){
  int max_n = numbers[0];
  for(int k=0; k<index;k++){
    if(abs(max_n) < abs(numbers[k])){
      max_n = numbers[k];
    }
  }
  return max_n;
}
```

Название файла: abs_max.h

```
#include <stdlib.h>

int abs_max();
```

Название файла: abs_min.c

```
#include "abs_min.h"

int abs_min(int numbers[], int index){
  int min_n = numbers[0];
  for(int p=0; p<index; p++){
    if(abs(min_n) > abs(numbers[p])){
      min_n = numbers[p];
    }
  }
  return min_n;
}
```

Название файла: abs_min.h

```
#include <stdlib.h>

int abs_min();
```

Название файла: diff.c

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

int diff(int numbers[], int index){
  int different = abs_max(numbers, index) - abs_min(numbers, index);
  return different;
}
```

Название файла: diff.h

```
int diff();
```

Название файла: sum.c

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

int sum(int numbers[], int index){
  int numb = abs_max(numbers, index);
  int summa = 0;
  int key = 0;
  for(int j = 0; j < index; j++){
    if (numbers[j] == numb){
      key = 1;
    }
    if (key==1){
      summa = summa + numbers[j];
    }
  }
  return summa;
}
```

Название файла: sum.h

```
int sum();
```

Название файла: Makefile

```
all: menu.o abs_max.o abs_min.o diff.o sum.o
	gcc menu.o abs_max.o abs_min.o diff.o sum.o -o menu
menu.o: menu.c
	gcc -c menu.c -std=c99
abs_max.o: abs_max.c
	gcc -c abs_max.c -std=c99
abs_min.o: abs_min.c
	gcc -c abs_min.c -std=c99
diff.o: diff.c
	gcc -c diff.c -std=c99
sum.o: sum.c
	gcc -c sum.c -std=c99
clean:
	rm -rf *.o menu
```