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

Студентка гр. 0382 _____ Ситченко К.С.

Преподаватель _____ Жангиров Т.Р

Санкт-Петербург
2020

Цель работы.

Изучить способы сборки программ на языке Си. Научиться работать с утилитой make.

Задание.

Вариант 5.

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.

Сборка проекта – это процесс получения исполняемого файла из исходного кода.

Сборка проекта вручную может стать довольно утомительным занятием, особенно, если исходных файлов больше одного и требуется задавать некоторые параметры компиляции/линковки. Для этого используются **Makefile** - список инструкций для утилиты **make**, которая позволяет собирать проект сразу целиком.

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)

Также, рекомендуется создание цели clean, которая используется для очистки всех результатов сборки проекта

Использование нескольких целей и их зависимостей особенно полезно в больших проектах, так как при изменении одного файла не потребуется пересобирать весь проект целиком. Достаточно пересобрать измененную часть.

Выполнение работы.

Для решения задачи было создано несколько файлов.

1. abs_max.c – программа, содержащая в себе функцию abs_max(int arr[], int n), на вход которой подается массив и его размер, в цикле for находится максимальное по модулю число в массиве, затем возвращается его значение.

2. abs_min.c – программа, содержащая в себе функцию abs_min(int arr[], int n), на вход которой подается массив и его размер, в цикле for находится минимальное по модулю число в массиве, затем возвращается его значение.

3. diff.c – программа, содержащая в себе функцию diff(int arr[], int n), на вход которой подается массив и его размер, функция возвращает разницу между максимальным по модулю и минимальным по модулю элементом.

4. sum.c – программа, содержащая в себе функцию sum(int arr[], int n), на вход которой подается массив и его размер, функция возвращает сумму

элементов массива, расположенных после максимального по модулю элемента (включая этот элемент).

5. menu.c – основная программа, в которой изначально подключены заголовочные файлы abs_max.h, abs_min.h, diff.h, sum.h, которые содержат объявления о функциях abs_max, abs_min, diff, sum соответственно. Первое число, которое подается программе на вход, записывается в переменную action, значение которой в дальнейшем будет проверятся в операторе switch. Оставшиеся числа записываются в массив arr, а его размер, посчитанный в цикле, сохраняется в переменную size. В зависимости от значения переменной action, выполняются следующие функции:

“0”: abs_max(int arr[], int n)

“1”: abs_min(int arr[], int n)

“2”: diff(int arr[], int n)

“3”: sum(int arr[], int n)

Если переменная action не соответствует ни одному из вышеперечисленных значений, программа выведет сообщение об ошибке.

6. Makefile – предназначен для сборки проекта и создания исполняемого файла menu.

Разработанный программный код см. в приложении А.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	0 -28 26 30 22 -13 -28 3 -12 8 10 -19 -26 11 -6 -18 -3 -2 -26 18 8 -19 -17 -11 -12 -23 19 -16 -11 9	30	Программа вывела на экран максимальный по модулю элемент
2	0	Данные некорректны	Программа вывела сообщение об ошибке, так как не был введен массив
3	6 7 4 -6 -56 0	Данные некорректны	Программа вывела сообщение об ошибке, так как значение первого входного числа должно быть от 0 до 3
4	1 4 5 -7 -89 0	0	Программа вывела на экран минимальный по модулю элемент
5	2 24 5 -9 -5 -34 56 76 -76	71	Программа вывела на экран разницу между макс. по модулю и мин. по модулю элементом
6	3 4 5 3 87 -54 -20 45 -3 58	113	Программа вывела на экран сумму элементов массива, начиная с максимального

Выводы.

Были изучены способы сборки программ на языке Си и проведена работа с утилитой make.

Разработана программа, обрабатывающая команду пользователя и массив, а также в зависимости от значения переменной action выводящая результат на экран. Программа собирается с использованием make-файла и утилиты make, результатом работы которой является исполняемый файл menu.

Приложение А

Исходный код программы

1.1. Название файла: abs_max.c

```
#include<stdlib.h>

int abs_max(int arr[], int n) {
  int abs_max=-1; int i;
  for (i=0; i < n; i++)
    if (abs(arr[i])  > abs(abs_max))
      abs_max = arr[i];
  return abs_max;
}
```

1.2. Название файла: abs_max.h

```
int abs_max(int arr[], int n);
```

2.1. Название файла: abs_min.c

```
#include<stdlib.h>

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

2.2. Название файла: abs_min.h

```
int abs_min(int arr[], int n);
```

3.1. Название файла: diff.c

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

int diff(int arr[], int n) {
  return abs_max(arr, n) - abs_min(arr, n);
}
```

3.2. Название файла: diff.h

```
int diff(int arr[], int n);
```

4.1. Название файла: sum.c

```
#include<stdlib.h>

int sum(int arr[], int n){
  int sum=0; int i, index=0;
  for (i=1; i<n; i++)
```

```
    if (abs(arr[i]) > abs(arr[index]))
      index = i;
  for (i=index; i < n; i++)
    sum += arr[i];
return sum;
}
```

4.2. Название файла: sum.h

```
int sum(int arr[], int n);
```

5. Название файла: menu.c

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

int main(void){
  int action, size=0;
  char s;
  int arr[100];
  scanf("%d%c", &action, &s);
  while (size <= 100 && s == ' '){
      scanf("%d%c", &arr[size], &s);
      size++;
  }

  if (size == 0)
    printf("Данные некорректны\n");
  else{
     switch (action){
      case 0:
       printf("%d\n", abs_max(arr, size));
       break;
      case 1:
       printf("%d\n", abs_min(arr, size));
       break;
      case 2:
       printf("%d\n", diff(arr, size));
       break;
      case 3:
       printf("%d\n", sum(arr, size));
       break;
      default:
       printf("Данные некорректны\n");
       break;
     }
  }
  return 0;
}
```

6. Название файла: Makefile

```
all: menu


menu: 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

abs_max.o: abs_max.c
    gcc -c abs_max.c

abs_min.o: abs_min.c
    gcc -c abs_min.c

diff.o: diff.c
    gcc -c diff.c

sum.o: sum.c
    gcc -c sum.c

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