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. 0304 Докучаев Р.А.

Преподаватель Чайка К.В.

Санкт-Петербург
2020

Цель работы.

Изучить процесс сборки программ на языке С, научиться создавать заголовочные файлы, разбивать проект на файлы и собирать его с помощью утилиты *make*.

Задание.

Вариант №1.

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 20**. Числа разделены пробелами. Строка заканчивается символом перевода строки.

В зависимости от значения, функция должна выводить следующее:

0: индекс первого отрицательного элемента. (*index_first_negative.c*)

1: индекс последнего отрицательного элемента. (*index_last_negative.c*)

2: Найти произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). (*multi_between_negative.c*)

3: Найти произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). (*multi_before_and_after_negative.c*)

иначе необходимо вывести строку "Данные некорректны".

Основные теоретические положения.

Были использованы заголовочные файлы стандартных библиотек: *stdio* и *stdlib* (из данных библиотек были задействованы функции *int scanf(const char*, ...)* и *int getchar(void))*. Также были использованы управляющие конструкции языка Си: циклы с счетчиком for и с предусловием *while*, условный оператор *if* и оператор множественного выбора (ветвления) *switch*. Также для разбиения исходного кода на несколько файлов использовались расширения *.c*, созданы заголовочные файлы *.h*, использована утилита make и создан специальный *Makefile*, главная цель которого собирает программу с помощью утилиты *gcc*.

Выполнение работы.

1. Создание исходного кода.
   1.1. Подключение стандартных библиотек *stdio* и *stdlib*.
   1.2. При помощи директивы процессора *#define* задается символическая константа *N*, которая обозначает максимальное количество элементов массива.
   1.3. Создаются прототипы функций, на которые в дальнейшем будет ссылаться оператор множественного выбора *switch*:

```
int index_first_negative (arr, n)
int index_last_negative (arr, n)
int multi_between_negative (arr, n)
int multi_before_and_after_negative(arr, n)
```

В качестве аргументов в этих функциях используются массив *arr []* и его длина *n*.

   1.4. В функции *int main ()* объявляется переменная *key*, которая отвечает за выбор используемой функции, создается масив *arr[]* типа *int* с размерностью *N* и количеством считанных элементов (длиной) *n*.
   1.5. При помощи функции *scanf* вызывается переменная *key*, выбранная пользователем.
   1.6. При помощи цикла с предусловием *while* и функции *scanf* происходит ввод и считывание элементов целочисленного массива. При этом в цикле *while* указано условие, когда считывание данных прекращается: превышение размерности массив ($n<N$) и символ переноса срока (*getchar() != '\n'* – если введенный символ совпадает с символом переноса строки, то считывание символов прекращается). Также после выполнения считывания символа происходит инкрементирование *n* – счётчика массива.

1.7. Далее при помощи оператора множественного выбора switch происходит вызов функции, соответствующей выбору пользователя. Результат выполнения функции выводится на экран.

1.8. Функции, используемые в данной работе, работают следующим образом:

```
index_first_negative (int arr [], int n)
```

1) объявляется переменная *i* типа *int*
2) при помощи цикла *for* переменная *i* инкрементируется с 0 и до момента нахождения первого отрицательного элемента последовательности
3) функция возвращает значение переменной *i*

```
index_last_negative (int arr [], int n)
```

1) объявляется переменная *i* типа *int*
2) при помощи цикла for переменная i декрементируется от (n-1) до момента нахождения первого с конца отрицательного элемента последовательности
3) функция возвращает значение *i*

```
multi_between_negative (arr [], int n)
```

1) вводятся символические переменные *a* и *b* (порядковые номера первого и последнего отрицательных чисел соответственно), переменная *i*, которая отвечает за порядковый номер элемента, и переменная *um_between*, которая отвечает за произведение элементов от первого отрицательного до числа, предшествующего последнему отрицательному
2) при помощи цикла *for* производится перемножение элементов массива, начиная с элемента с порядковым номером *a* и заканчивая элементом, расположенного перед элементом с порядковым номером *b*
3) функция возвращает значение *um_between*

```
multi_before_and_after_negative (arr [], int n)
```

1) вводятся символические переменные *a* и *b* (порядковые номера первого и последнего отрицательных чисел соответственно), переменная *i*, которая отвечает за порядковый номер элемента, и переменная *um_before_and_after*, которая отвечает за произведение элементов от 0 до элемента, предшествующего первому отрицательному элементу, и от последнего

отрицательного элемента до последнего элемента массива

2) при помощи двух циклов *for* производится подсчет произведений. Первый цикл *for* производит перемножение элементов массива от *arr [0]* до *arr [a-1]*, а второй цикл *for* производит перемножение элементов от *arr [b]* до *arr [n]*

3) функция возвращает значение *um_before_and_after*

2. Создание заголовочных файлов, которые содержат прототипы функций:

index_first_negative.h:
```
int index_first_negative (arr [], int n);
```
index_last_negative.h:
```
int index_last_negative (arr [], int n);
```
multi_between_negative.h:
```
int multi_between_negative (arr [], int n);
```
multi_before_and_after_negative.h:
```
int multi_before_and_after_negative (arr [], int n);
```

3. Создание файла *menu.c*, в котором заключается функция *int main()* с подключением заголовочных файлов из пункта 2:

```
#include “index_first_negative”
#include “index_last_negative”
#include “multi_between_negative”
#include “multi_before_and_after_negative”
```

4. Создание файлов *index_first_negative.c*, *index_last_negative.c*, *multi_between_negative.c* и *multi_before_and_after_negative.c* , которые соответствуют исходным функциям
5. Создание *Makefile* с целью *all*, которая представляет собой вызов *gcc* с параметрами в виде всех выше указанных файлов с расширением *.c*. Параметр *-o menu* задаёт имя *menu* итоговому исполняемому файлу.

Исходный код приведён в пункте 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 6 7 -6 7	3	OK
2.	1 2 4 5 65 76 878 99 -556 56 -65 889 322 -342 566 -454 34	14	OK
3.	2 1 2 3 4 -5 6 7 8 9 -10 11 12 13 -14 15 16 17	259459200	OK
4.	3 1 2 3 4 -5 6 7 8 9 -10 11 12 13 -14 15 16 17 -18 19 20	-164160	OK
5.	4 1 24 6596 -3 5868 583 -21 54 35 -54 56	Данные некорректны	OK

Выводы.

Был изучен процесс сборки программ в C с помощью утилиты make.

Разработана программа, выполняющая считывание с клавиатуры исходных данных со стандартного потока вывода и выбор пользователя, которая затем выводит полученную в результат, получаемый при выполнении программы. После этого все пронумерованные операции были внесены в отдельные файлы *index_first_negative.c*, *index_last_negative.c*, *multi_between_negative.c*, *multi_before_and_after_negative.c*. Также были созданы заголовочные файлы, эквивалентные данным программам, а общий вызов необходимой операции производился с помощью файла *menu.c*. Сборка производилась при помощи утилиты *make*.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

index_first_negative.c:

```
#include "index_first_negative.h"

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

index_first_negative.h:

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

index_last_negative.c:

```
#include "index_last_negative.h"

int index_last_negative(int arr[], int n){
     int last_negative;
     int i = 0;
     for (i = (n-1); i >= 0; i--){
      if(arr[i] < 0)
            break;
     }
     return i;
}
```

index_last_negative.h:

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

multi_between_negative.c:

```
#include "multi_between_negative.h"
#include "index_first_negative.h"
#include "index_last_negative.h"

int multi_between_negative(int arr[], int n){
     int i = 0;
     int um_between = 1;
     for(i = index_first_negative (arr, n); i <
     index_last_negative (arr, n); i++){
      um_between *= arr[i];
     }
     return um_between;
}
```

multi_between_negative.h:

```
                int multi_between_negative(int arr[], int n);
```
multi_before_and_after_negative.c:
```
                #include "multi_before_and_after_negative.h"
                #include "index_first_negative.h"
                #include "index_last_negative.h"

                int multi_before_and_after_negative(int arr[],
                int n){
                    int i = 0;
                    int um_before_and_after = 1;
                    for (i = 0; i < index_first_negative (arr,
                    n); i++) um_before_and_after *= arr[i];
                    for (i = index_last_negative (arr, n); i < n;
                    i++) um_before_and_after *= arr[i];
                    return um_before_and_after;
                }
```
multi_before_and_after_negative.h:
```
                int multi_before_and_after_negative(int arr[],
                int n);
```
menu.c:
```
                #include <stdio.h>
                #include <stdlib.h>

                #include "index_first_negative.h"
                #include "index_last_negative.h"
                #include "multi_between_negative.h"
                #include "multi_before_and_after_negative.h"

                #define N 20

                int main(){
                    int n = 0, key;
                    int arr[N];
                    scanf("%d" , &key);
                    while (getchar() != '\n' && n < N){
                          scanf("%d", &arr[n]);
                          n ++;
                    }

                    switch(key){
                          case 0:
                               printf("%d\n", index_first_nega-
                        tive(arr, n));
                               break;
                          case 1:
                               printf("%d\n", index_last_nega-
                        tive(arr, n));
                               break;
```

```
                        case 2:
                              printf("%d\n", multi_between_neg-
                        ative(arr, n));
                              break;
                        case 3:
                              printf("%d\n", multi_be-
                        fore_and_after_negative(arr, n));
                              break;
                        default:
                              puts("Данные некорректны");
                              break;
                  }
                  return 0;
            }
```

Makefile:

```
all: menu.o index_first_negative.o index_last_negative.o
multi_between_negative.o multi_before_and_after_nega-
tive.o
    gcc menu.o index_first_negative.o index_last_nega-
    tive.o multi_between_negative.o multi_before_and_af-
    ter_negative.o -o menu


menu.o: menu.c index_first_negative.h index_last_nega-
tive.h multi_between_negative.h multi_before_and_af-
ter_negative.h
    gcc -c menu.c


index_first_negative.o: index_first_negative.c in-
dex_first_negative.h
    gcc -c index_first_negative.c


index_last_negative.o: index_last_negative.c in-
dex_last_negative.h
    gcc -c index_last_negative.c


multi_between_negative.o: multi_between_negative.c
multi_between_negative.h
    gcc -c multi_between_negative.c


multi_before_and_after_negative.o: multi_before_and_af-
ter_negative.c multi_before_and_after_negative.h
    gcc -c multi_before_and_after_negative.c
```