**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра Математического обеспечения электронно-вычислительных машин**

# ОТЧЕТ

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема: Сборка программ в Си. Работа с утилитой make.**

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

**Цель работы.**

Отработать на практике принципы сборки программ в си. Научиться работать с утилитой make и создавать Makefile.

**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 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.**

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.**

Название файла: menu.c

1.Подключение заголовочных файлов:

```
#include <stdio.h> //--одна из стандартных библиотек
#include <stdlib.h>//--одна из стандартных бииблиотек
#include "index_first_negative.h" //--заголовочный файл с объявлением функции index_first_negative()
#include "index_last_negative.h" //--заголовочный файл с объявлением функции index_last_negative()
#include "multi_between_negative.h" //--заголовочный файл с объявлением функции multi_between_negative()
#include "multi_before_and_after_negative.h"//--заголовочный файл с объявлением функции multi_before_and_after_negative()
```

1. Инициализируем переменную *int inp=0*, в которой в дальнейшем будет храниться считываемое значение – команда пользователя(целое число от 0 до 3), соответствующая подзадаче, которую должна будет выполнить программа.

2. Инициализируем целочисленный массив из 20 элементов *int arr[20]={0}* и переменную *int arr_size=0*, в которой будет считаться количество элементов массива в зависимости от исходных данных, так как в условии сказано, что элементов в массиве не больше 20.

3. Инициализируем переменную *char sym = ´ ´*, она нам понадобится для того, чтобы в дальнейшем прекратить ввод элементов массива, подающихся на вход строкой(числа разделены пробелом), оканчивающейся символом перевода строки.

4. Считываем с клавиатуры команду пользователя(целое число от 0 до 3), соответствующую подзадаче, которую должна выполнить программа, с помощью функции *scanf("%d",&inp).*

5. Далее будем считывать с клавиатуры элементы массива для дальнейшей работы с ними и одновременно считать количество введенных элементов, используя *arr_size++*. По условию элементы подаются на вход через пробел строкой, оканчивающейся символом перевода строки. Для их считывания воспользуемся циклом с предусловием:

```
while(arr_size<20&&sym==' '){
   scanf("%d%c",&arr[arr_size++],&sym);
}
```

Элементы будут считываться до тех пор, пока количество введенных элементов *arr_size* меньше 20 и следующий за элементом символ – пробел(´ ´). То есть, как только количество введенных элементов превысит лимит или пользователь введет символ перевода строки(´\n´ - в нашем случае, символизирует конец входной строки), программа прекратит ввод.

6. Для того, чтобы в зависимости от введенной пользователем команды(int inp - целое число от 0 до 3), программа переходила к решению определенной подзадачи, воспользуемся оператором множественного выбора *switch(inp).*

7.Опишем каждый операторный блок(case).

Если пользователь первым входным числом введет 0, то программа перейдет к выполнению операторного блока *case 0*, где будет вызвана функция *index_first_negative(arr,arr_size),* которая возвращает значение индекса первого отрицательного элемента массива. Однако, чтобы возвращенное значение вывести на экран, нужно воспользоваться функцией p*rintf("%d\n", index_first_negative(arr,arr_size))*. Также не забудем про оператор *break*, чтобы исключить последовательное выполнение операторных блоков после первого совпадения.

Если пользователь первым входным числом введет 1, то программа перейдет к выполнению операторного блока *case 1*, где будет вызвана функция *index_last_negative(arr,arr_size),* которая возвращает значение индекса последнего отрицательного элемента массива. Однако, чтобы возвращенное значение вывести на экран, нужно воспользоваться функцией *printf("%d\n", index_last_negative(arr,arr_size))*. Также не забудем про оператор *break*, чтобы исключить последовательное выполнение операторных блоков после первого совпадения.

Если пользователь первым входным числом введет 2, то программа перейдет к выполнению операторного блока *case 2*. Для выполнения подзадачи нам понадобятся индексы первого и последнего отрицательного элемента массива, их мы найдем внутри функции, которую вызовем, с помощью уже написанных первых двух функций. Поэтому нам остается вызвать нашу функцию *multi_between_negative(arr, arr_size),* которая возвращает значение произведения элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент). Однако, чтобы возвращенное значение вывести на экран, нужно воспользоваться функцией *printf("%d\n", multi_between_negative(arr, arr_size).* Также не забудем про оператор *break,* чтобы исключить последовательное выполнение операторных блоков после первого совпадения.

Если пользователь первым входным числом введет 3, то программа перейдет к выполнению операторного блока *case 3*. Для выполнения подзадачи нам понадобятся индексы первого и последнего отрицательного элемента массива, их мы найдем внутри функции, которую вызовем, с помощью уже написанных первых двух функций. Поэтому нам остается вызвать нашу функцию *multi_before_and_after_negative(arr, arr_size),* которая возвращает значение произведения элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). Однако, чтобы возвращенное значение вывести на экран, нужно воспользоваться функцией *printf("%d\n", multi_before_and_after_negative(arr,arr_size).* Также не забудем про оператор *break,* чтобы исключить последовательное выполнение операторных блоков после первого совпадения.

Если пользователь первым входным числом ввел число, отличное от целых чисел 0, 1, 2, 3, то программа перейдет к операторному блоку *default*, в котором будут лишь инструкция вывести на экран “Данные некорректны”( с помощью функции *printf("%s\n", "Данные некорректны") )*, а также оператор *break.*

Функции – для определения каждой функции выделен отдельный файл

1.Название файла: index_first_negative.c

Подключение заголовочных файлов:

```
#include <stdio.h> //--одна из стандартных библиотек
#include "index_first_negative.h"//--нужен, так как в нем содержится объявление соответствующей функции
```

Функция нужна для решения первой подзадачи: найти индекс первого отрицательного элемента массива.

Тип возвращаемого функцией значения: *int*(индекс элемента массива – целое число)

Имя функции: *index_first_negative*(дано по условию)

Аргументы функции: *(int arr[], int n)*

Функция будет принимать массив *int arr[]*, который нужно обработать. В квадратных скобках размерность массива не указываем, функции в Си не умеют самостоятельно определять размерность переданного им массива, поэтому нам нужно отдельным параметром передать его размер. В нашей функции мы передаем размер массива с помощью переменной int n.

Тело функции: Для того, чтобы найти первый отрицательный элемент массива, будем перебирать все элементы массива с помощью цикла *for(int i=0;i<n;i++),* где n – размер массива, и проверять элемент на отрицательность условием *if(arr[i]<0).* Как только найдется элемент, удовлетворяющий условию, вернем значение индекса этого элемента(он окажется первым отрицательным элементом в массиве) с помощью оператора return.

2.Название файла: index_last_negative.c

Подключение заголовочных файлов:

```
#include <stdio.h>//--одна из стандартных библиотек
#include "index_last_negative.h"//--нужен, так как в нем содержится объявление соответствующей функции
```

Функция нужна для решения второй подзадачи: найти индекс последнего отрицательного элемента массива.

Тип возвращаемого функцией значения: *int*(индекс элемента массива – целое число)

Имя функции: *index_last_negative*(дано по условию)

Аргументы функции: *(int arr[], int n)*

Функция будет принимать массив *int arr[]*, который нужно обработать. В квадратных скобках размерность массива не указываем, функции в Си не умеют самостоятельно определять размерность переданного им массива,

поэтому нам нужно отдельным параметром передать его размер. В нашей функции мы передаем размер массива с помощью переменной int n.

Тело функции: Для того, чтобы найти последний отрицательный элемент массива, будем перебирать с конца все элементы массива с помощью цикла *for(int i=n-1;i>=0;i--),* где n – размер массива, и проверять элемент на отрицательность условием *if(arr[i]<0).* Как только найдется первый с конца отрицательный элемент, возвращаем значение переменной i(искомый индекс последнего отрицательного элемента массива), с помощью оператора return .

3.Название файла: multi_between_negative.c

Подключаемые заголовочные файлы:

```
#include <stdio.h>//--стандартная библиотека
#include "multi_between_negative.h"//для объявления функции
#include "index_first_negative.h"//--объявление функции, определяющей первый отриц.элемент массива
#include "index_last_negative.h"//--объявление функции, определяющей последний отриц.элемент массива
```

Функция нужна для решения третьей подзадачи: найти произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент)

Тип возвращаемого функцией значения: *int*(элементы массива – целые числа, значит, произведение любых его элементов есть целое число)

Имя функции: multi*_between_negative*(дано по условию)

Аргументы функции: *(int arr[], int n)*

Функция будет принимать массив *int arr[]*, который нужно обработать. В квадратных скобках размерность массива не указываем, функции в Си не умеют самостоятельно определять размерность переданного им массива, поэтому нам нужно отдельным параметром передать его размер. В нашей функции мы передаем размер массива с помощью переменной *int n*.

Тело функции:

Индексы первого и последнего отрицательных элементов массива найдем с помощью уже написанных первых двух функций

index_first_negative и index_last_negative. Присвоим переменным int a и int b возвращенные этими функциями значения соответственно.

Для того, чтобы найти произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент), буду перебирать элементы массива, начиная с первого отрицательного элемента, до последнего отрицательного(не включая его) с помощью цикла for и значения элементов буду перемножать, обновляя значение переменной *int p* на каждой итерации*(*p инициализирована в начале тела фунции*)*. Цикл for будет выглядеть так: *for(int i=a;i<b;i++)p*=arr[i]*. После выполнения всех итераций, получим итоговое значение искомого произведения. Вернем это значение с помощью оператора return.

4. Название файла: multi_before_and_after_negative.c

Подключаемые заголовочные файлы:

```
#include <stdio.h> // --стандартная библиотека
#include "multi_before_and_after_negative.h"//--в нем содержится объявление искомой функции
#include "index_first_negative.h"//-- объявление функции, определяющей первый отриц.элемент массива
#include "index_last_negative.h"//--объявление функции, определяющей последний отриц.элемент массива
```

Функция нужна для решения четвертой подзадачи: найти произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент).

Тип возвращаемого функцией значения: *int*(элементы массива – целые числа, значит, произведение любых его элементов есть целое число)

Имя функции: multi*_before_and_after_negative*(дано по условию)

Аргументы функции: *(int arr[], int n)*

Функция будет принимать массив *int arr[]*, который нужно обработать. В квадратных скобках размерность массива не указываем, функции в Си не умеют самостоятельно определять размерность переданного им массива,

поэтому нам нужно отдельным параметром передать его размер. В нашей функции мы передаем размер массива с помощью переменной *int n*.

Тело функции: Индексы первого и последнего отрицательных элементов массива найдем с помощью уже написанных первых двух функций index_first_negative и index_last_negative. Присвоим переменным int a и int b возвращенные этими функциями значения соответственно.

Для того, чтобы найти произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент), сначала переберем с помощью цикла for элементы массива, начиная с нулевого, заканчивая первым отрицательным(не включая элемент), и на каждой итерации будем обновлять значение *p*=arr[i]*(переменную int p инициализируем в начале тела функции). И делаем второй перебор с помощью цикла for: начиная с последнего отрицательного элемента(включая его), заканчивая последним элементом массива(включая). На каждой итерации цикла будем обновлять значение p*=arr[i] (переменную *int p* инициализируем в начале тела функции*)*. После выполнения двух циклов for, мы получим итоговое значение искомого произведения. Вернем это значение с помощью оператора return.

Makefile:

Название файла: Makefile

Первая цель all – по умолчанию, для сборки всего проекта, собирает все объектные файлы (index_first_negative.o, index_last_negative.o, multi_between_negative.o, multi_before_and_after_negative.o, menu.o)

Так как объектные файлы – зависимости первой цели, чтобы ее достичь, нужно получить эти объектные файлы, поэтому делаем каждый из них очередной целью. Таким образом, цель *all*, ее зависимости и команды будут выглядеть так:

```
all: index_first_negative.o index_last_negative.o multi_between_negative.o multi_before_and_after_negative.o menu.o
	gcc index_first_negative.c index_last_negative.c multi_between_negative.c multi_before_and_after_negative.c menu.c -o menu
```

Чтобы получить *index_first_negative.o,* мы должны скомпилировать файл *index_first_negative.c*. Но надо не забыть указать в зависимостях подключаемые заголовочные файлы, чтобы в случае изменений в заголовочных файлах, мы учли изменения. Так, цель, зависимости и команды для *index_first_negative.o* будут выглядеть так:

```
index_first_negative.o: index_first_negative.c index_first_negative.h
	gcc -c index_first_negative.c
```

Аналогично будут выглядеть цели *index_last_negative.o, multi_between_negative.o, multi_before_and_after_negative.o, menu.o*

Не забываем про цель *clean,* которая поможет быстро пересобрать программу в случае изменений и удалить все объектные и исполняемый файл из директории.

```
clean:
	rm *.o 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 -5 -3 -5 -8 3 -9 -3 | 0 | Верно, так как первый отрицательный элемент массива - нулевой |
| 2. | 1 4 -1 3 2 -2 1 -4 5 | 6 | Верно, так как последний отрицательный элемент массива имеет индекс 6 |
| 3. | 2 5 -2 3 -1 2 -4 8 | 12 | Верно, так как произведение элементов |

| | | | |
|---|---|---|---|
| | | | массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент) – равно 12 |
| 4. | 3 2 3 -1 2 -2 3 -4 7 1 | -168 | Верно, так как произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент) – равно -168 |

**Выводы.**

Были отработаны принципы сборки программ в си, а также основные принципы работы с утилитой Make и создания Makefile.

Разработана функция-меню, выполняющая считывание с клавиатуры команды пользователя и исходного целочисленного массива. Для обработки команды пользователя использовался оператор множественного выбора *switch*. В зависимости от команды пользователя для выполнения соответствующей подзадачи вызывалась функция, которая обрабатывала массив и возвращала искомое значение. Программа выводит это значение на экран. Вся программа собирается с использованием Makefile.

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: 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"
int main()
{
    int inp=0;
    int arr[20]={0};
    int arr_size=0;
    char sym = ' ';

    scanf("%d",&inp);

    while(arr_size<20&&sym==' '){
        scanf("%d%c",&arr[arr_size++],&sym);
    }

    switch(inp){
        case 0: {
            printf("%d\n", index_first_negative(arr,arr_size));
            break;
        }
        case 1: {
            printf("%d\n", index_last_negative(arr,arr_size));
            break;
        }
        case 2: {
            printf("%d\n", multi_between_negative(arr,arr_size));
            break;
        }
        case 3: {
            printf("%d\n",
multi_before_and_after_negative(arr,arr_size));
            break;
        }
        default: {
            printf("%s\n","Данные некорректны");
            break;
        }
    }
}
```

Другие файлы см. на следующих страницах

Функции:

Название файла: index_first_negative.c

```
#include <stdio.h>
#include "index_first_negative.h"
int index_first_negative(int arr[], int n){ // n - размер массива
    int i=0;
    for(i;i<n;i++){
        if(arr[i]<0){
            return i;
        }
    }
}
```

Название файла: index_last_negative.c

```
#include <stdio.h>
#include "index_last_negative.h"
int index_last_negative(int arr[], int n){
    int i=0;
    for(i=n-1;i>=0;i--){
        if(arr[i]<0){
            return i;
        }
    }
}
```

Название файла: multi_between_negative.c

```
#include <stdio.h>
#include "multi_between_negative.h"
#include "index_first_negative.h"
#include "index_last_negative.h"
int multi_between_negative(int arr[],int n){
    int p=1;
    int a=index_first_negative(arr,n);
    int b=index_last_negative(arr,n);
    int i=0;
    for(i=a;i<b;i++)p*=arr[i];
    return p;
}
```

Другие файлы см. на следующих страницах

Название файла: multi_before_and_after_negative.c

```
#include <stdio.h>
#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 p=1;
    int a=index_first_negative(arr,n);
    int b=index_last_negative(arr,n);
    int i=0;
    for(i;i<a;i++)p*=arr[i];
    for(i=b;i<n;i++)p*=arr[i];
    return p;
}
```

Заголовочные файлы:

Название файла: index_first_negative.c

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

Название файла: index_last_negative.c

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

Название файла: multi_between_negative.c

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

Название файла: multi_before_and_after_negative.c

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

Makefile см.на след.стр.

Makefile:

Название файла: Makefile

```
all:            index_first_negative.o            index_last_negative.o
multi_between_negative.o multi_before_and_after_negative.o menu.o
        gcc        index_first_negative.c        index_last_negative.c
multi_between_negative.c  multi_before_and_after_negative.c  menu.c  -o
menu

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

index_last_negative.o: index_last_negative.c index_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_after_negative.c
multi_before_and_after_negative.h
        gcc -c multi_before_and_after_negative.c

menu.o:     menu.c     index_first_negative.h     index_last_negative.h
multi_between_negative.h multi_before_and_after_negative.h
        gcc -c menu.c

clean:
        rm *.o menu
```