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.

Работа с утилитой make.

Студент гр. 0382		Тюленев Т. В.
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>
#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"
```

1. Инициализируем переменные *int n=0*; *int arr[20]={0}; int arr_size=0; char l = ′ ′*;

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

3. 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&&l==' '){ scanf("%d
   %c",&arr[arr_size++],&l);}
```

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.

4. 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(n)*.

Функции – для определения каждой функции выделен отдельный файл

1.Название файла: index_first_negative.c

Подключение заголовочных файлов:

```
#include <stdio.h>
#include   "index_first_negative.h"
```

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* size*)*

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 size.

Тело функции: Для того, чтобы найти первый отрицательный элемент массива, будем перебирать все элементы массива с помощью цикла *for(int i=0;i<size;i++),* где size – 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>
#include   "index_last_negative.h"
```

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 size)*

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 *size*. Тело

функции: Для того, чтобы найти последний отрицательный элемент массива, будем перебирать с конца все элементы массива с помощью цикла *for(int i=size-1;i>=0;i--),* где *size* – 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>
#include "multi_between_negative.h"
#include "index_first_negative.h"
#include "index_last_negative.h"
```

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 size)*

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 size*.

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 x и int y 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* multi на каждой итерации*(* multi инициализирована в начале тела фунции*).* Цикл for будет выглядеть так: *for(int i=x;i<y;i++)*multi*=*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>
#include "multi_before_and_after_negative.h"
#include   "index_first_negative.h"
#include      "index_last_negative.h"
```

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 size)*

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 size*. 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 x и int y 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 multi*=*arr[i]*(переменную int multi 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 multi*=arr[i] (переменную *int* multi инициализируем в начале тела функции*).* После выполнения двух циклов for, мы получим итоговое значение искомого произведения. Вернем это значение с помощью оператора return.

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	true
2	0 1 -2 3 4 5 6	1	true
3	0 1 2 -3 -4 -5 -6	2	true
4	0 1 2 3 4 5 6		true
5	1 1 2 3 4 -5 6	4	true
6	1 1 2 -3 4 -5 6	4	true
7	1 1 2 3 4 5 6		true
8	2 1 -2 3 4 -5 6	-24	true
9	2 1 -2 -3 -4 -5 6	-24	true
10	2 1 2 3 4 5 6	0	true
11	2 1 -2 3 4 5 6	-2	true
12	3 1 -2 3 4 -5 6	-30	true
13	3 1 -2 -3 -4 -5 6	-30	true
14	3 1 2 3 4 5 6	720	true
15	3 1 -2 3 4 5 6	-720	true
16	13 0 0 0 0 0 0 0	Данные некорректны	true

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
"index_first_negativ
e.h"
#include
"index_last_negative
.h"
#include
"multi_between_negat
ive.h"
#include
"multi_before_and_af
ter_negative.h"
int main()
{
    int n;
    int arr[20]={0};
    int arr_size=0;
    char l = ' ';

    scanf("%d",&n);


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

    switch(n){
        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_negati
ve(arr,arr_size));
            break;
```

```
        }
        case 3: {

printf("%d\n",
multi_before_and_aft
er_negative(arr,arr_
size));
            break;
        }
        default: {

printf("%s\
n","Данные
некорректны");
            break;
        }
    }
}
```

Функции Название файла: index_first_negative.c

```
#include "index_first_negative.h"
int index_first_negative(int arr[], int size){
   int i;
   for(i=0;i<size;i++){
      if(arr[i]<0){
         return i;
      }
   }
}
```

Функции Название файла: index_last_negative.c

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

Функции Название файла: 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 size){
    int multi=1;
    int x=index_first_negative(arr,size);
    int y=index_last_negative(arr,size);
    int i=0;
    for(i;i<x;i++){
    multi*=arr[i];
    }
    for(i=y;i<size;i++){
    multi*=arr[i];
    }
    return multi;
}
```

Функции Название файла: 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 size){
    int multi;
    int x=index_first_negative(arr,size);
    int y=index_last_negative(arr,size);
    int i;
    multi=1;
    for(i=x;i<y;i++){
    multi*=arr[i];
    }
    return multi;
}
```

Makefile:

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

menu.o: menu.c
                                        gcc -c menu.c

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

index_last_negative.o: index_last_negative.c
                                        gcc -c
index_last_negative.c

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

multi_before_and_after_negative.o:
multi_before_and_after_negative.c
                                        gcc -c
multi_before_and_after_negative.c
clean:
                                        rm -rf *.o menu
```