МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Программированию»
Тема: Сборка программ в Си

Студент гр. 0382 _____ Мукатанов А.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение процесса сборки программ на языке си при помощи утилиты Make.

Задание.

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.

Операторный блок - несколько операторов, сгруппированные в единый блок с помощью фигурных скобок { [<оператор 1>...<оператор N>] }

Условный оператор: if (<выражение>) <оператор 1> [else <оператор 2>]

Если выражение интерпретируется как истина, то оператор1 выполняется. Может иметь необязательную ветку else, путь выполнения программы пойдет в случае если выражение ложно. В языке С любое ненулевое выражение расценивается как истина. Оператор множественного выбора switch (<выражение>)

{ case <константное выражение 1>: <операторы 1>

...

case <константное выражение N>: <операторы N>

[default: <операторы>]

}

Выполняет поочередное сравнение выражения со списком константных выражений. При совпадении, выполнение программы начинается с соответствующего оператора. В случае, если совпадений не было, выполняется необязательная ветка default. Важно помнить, что операторы после первого совпадения будут выполняться далее один за другим. Чтобы этого избежать, следует использовать оператор break Цикл со счетчиком

for ([<начальное выражение>]; [<условное выражение>]; [<выражение приращения>])

<оператор>

Условием продолжения цикла, как и в цикле с предусловием, является некоторое выражение, однако в цикле со счетчиком есть еще 2 блока — начальное выражение, выполняемое один раз перед первым началом цикла и выражение приращения, выполняемое после каждой итерации цикла.

Любая из трех частей оператора for может быть опущена.

Оператор break — досрочно прерывает выполнение цикла.

Выполнение работы.

Используется стандартная библеотека языка си.

Переменная command определяет ,какая из следующих функций будет использоваться. Вводится массив и считывается его длина в переменную a_size.Далее с помощью оператора switch ,в котором описывается 5 случаев (4 из которых заданные функции , а 5 выводит :” Данные некорректны”), и переменной command выполняется одна из следующих операций:

(0)Первая функция - int index_first_negative(ищет индекс первого отрицательного числа) , в ней я ввожу переменную fn (first negative) и передаю в нее индекс первого отрицательного э-та . Далее создаю цикл for ,который перестает читать массив после встречи первого отрицательного числа.

(1)Вторая функция - int index_last_negative(ищет индекс последнего отрицательного числа) .Ввожу переменную ln (last negative) и передаю ей индекс последнего отрицательного э-та. В ней я делаю все то же самое ,что и в первой функции ,но цикл for читает массив с конца.

(2)Третья функция - int multi_between_negative(ищет произведение чисел между первым (включая) и последним (не включая) отрицательным элементом). Чтобы посчитать произведение , я вызываю первую и вторую функции . Далее идет цикл for ,где первым индексом у меня является fn , а последним ln. Потом вводится переменная res (считает результат произведения).

(3)Четвертая функция - int multi_before_and_after_negative(ищет произведение чисел до первого (не включая) и после последнего (включая) отрицательного э-та).Делаю все то же ,что и в 3 функции , но ввожу 2 цикла for (в первом считается от эл-та с индексом 0 до ln , во втором от эл-та с индексом ln до a_size)

Переменные (которые не указал выше):

Int a[] - массив ;

Char sym - пробел при вводе чисел;

Res1,res2- результат произведения до первого отрицательного и после последнего отрицательного эл-та.

M_res - главный результат (объединяет res1 и res2)

Makefile

В make-файле предназначен для сборки проекта и создания исполняемого файла *menu.* В нём содержатся следующие инструкции:

Инструкция all.

Имеет следующие зависимости: *max.o, min.o, diff.o, sum.o, menu.o.*

Команда: gcc max.o min.o diff.o sum.o menu.o -o menu.

Выполнение данной инструкции приводит к сборке проекта из необходимых объектных файлов.

С помощью ключа «-o» сообщается название получаемого после выполнения исполняемого файла — *menu.*

Инструкция *menu.o.*

Имеет следующие зависимости: *menu.c, index_first_negative.h, index_last_negative.h, multi_between_negative.h, multi_before_and_after_negative.h.* Команда: *gcc -c menu.c*

Выполнение данной инструкции приводит к созданию объектного файла *menu.o.*

Инструкция *index_first_negative.o.*

Имеет следующие зависимости: *index_first_negative.c, index_first_negative.h.*

Команда: *gcc -c index_first_negative.c.*

Выполнение данной инструкции приводит к созданию объектного файла *index_first_negative.o.*

Инструкция *index_last_negative.o.*

Имеет следующие зависимости: *index_last_negative.c, index_last_negative.h.*

Команда: *gcc -c index_last_negative.c*

Выполнение данной инструкции приводит к созданию объектного файла *index_last_negative.o.*

Инструкция multi_between_negative.*o.*

Имеет следующие зависимости: multi_between_negative.*c* multi_between_negative.*h, index_first_negative.h, index_last_negative.h.*

Команда: *gcc -c* multi_between_negative.*c*

Выполнение данной инструкции приводит к созданию объектного файла multi_between_negative.*o.*

Инструкция *multi_before_and_after_negative.o.*

Имеет следующие зависимости: *multi_before_and_after_negative.c, multi_before_and_after_negative.h, index_first_negative.h, index_last_negative.h*

Команда: *gcc -c multi_before_and_after_negative.c*

Выполнение данной инструкции приводит к созданию объектного файла *multi_before_and_after_negative.o*

Инструкция *clean.*

Используется для удаления всех объектных файлов из текущей директории.

Команда: *rm *.o.*

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	Программа работает правильно
2.	1 -5 -3 -5 -8 3 -9 -3	6	Программа работает правильно
3.	2 -5 -3 -5 -8 3 -9 -3	-16200	Программа работает правильно

Выводы.

В ходе работы был изучен процесс сборки программ на языке Си при помощи утилиты Make.

Были изучены основы языка С, типизация переменных, получение данных, а также их передача.

Были изучены основные управляющие конструкции языка: условия, циклы, оператор switch.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для обработки команд пользователя использовались оператор множественного выбора switch. Для обработки команд пользователя также использовались условные операторы if,for.Для отлавливания некорректных данных был отведён раздел множественного выбора default.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла : menu.c

```
#include <stdio.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 command ;
    scanf("%d\n", &command);
    int a[20];
    int a_size = 0;
    char sym = ' ';
    while(a_size < 20 && sym == ' '){
        scanf("%d%c",&a[a_size++], &sym);
    }
switch(command){
    case 0:
        printf("%d\n", index_first_negative(a,a_size));
        break;
    case 1:
        printf("%d\n", index_last_negative(a,a_size));
        break;
    case 2:
        printf("%d\n", multi_between_negative(a,a_size));
        break;
    case 3:
        printf("%d\n", multi_before_and_after_negative(a,a_size));
        break;
```

```
    default:
        printf("Данные некорректны\n");
        break;
    }
return 0;
}
```

Название файла : index_first_negative.c

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

Название файла : index_first_negative.h

```
int index_first_negative(int a[], int a_size);
```

Название файла : index_last_negative.c

```
#include "index_last_negative.h"
int index_last_negative(int a[], int a_size){
    int ln;
    int i;
    for(i = a_size - 1; i >= 0; i--){
```

```
        if(a[i] < 0){
        ln = i;
        break;
    }
}
return ln;
}
```

Название файла : index_last_negative.h

```
int index_last_negative(int a[], int a_size);
```

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 a[], int a_size){
    int fn = index_first_negative(a, a_size);
    int ln = index_last_negative(a, a_size);
    int res = 1;
    int i;
    for (i = fn; i < ln; i++)
        res *= a[i];
return res;
}
```

Название файла: multi_between_negative.h

```
int multi_between_negative(int a[], int a_size);
```

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 a[], int a_size){
   int fn = index_first_negative(a, a_size);
   int ln = index_last_negative(a, a_size);
   int res1 = 1, res2 = 1, m_res = 0;
   int i;
   for(i = 0; i < fn; i++)
      res1 *= a[i];
   for(i = ln; i < a_size; i++){
      res2 *=a[i];
   }
   m_res = res1 * res2;
return m_res;
}
```

Название файла: multi_before_and_after_negative.h

```
int multi_before_and_after_negative(int a[], int a_size);
```

Название файла: Makefile

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

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: index_first_negative.h index_last_negative.h multi_between_negative.h multi_between_negative.c
	gcc -c multi_between_negative.c

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

clean:
	rm *.o
```