

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Программирование»
Тема: Условия, циклы, оператор switch

Студент гр. 7304

Давыдов А.А.

Преподаватель

Кринкин К.В.

Санкт-Петербург

2017

Цель работы.

Реализовать 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 выводит

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 с предусловием

while (<выражение>) <оператор>

На каждой итерации цикла происходит вычисление выражения и если оно истинно, то выполняется тело цикла

Цикл с постусловием

do <оператор> **while** <выражение>;

На каждой итерации цикла сначала выполняется тело цикла, а после вычисляется выражение. Если оно истинно — выполняется следующая итерация.

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.

Оператор **continue** — досрочный переход к следующей итерации цикла

Экспериментальные результаты.

1) menu.c

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

int main()
{
    int size = 0;
    int a[21];

    size += scanf("%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d", &a[0], &a[1], &a[2], &a[3], &a[4], &a[5], &a[6], &a[7], &a[8], &a[9], &a[10], &a[11], &a[12], &a[13], &a[14], &a[15], &a[16], &a[17], &a[18], &a[19], &a[20]);

    size -= 1;

    switch(a[0])
    {
        case 0: printf("%d\n", index_first_negative((a+1), size)); break;
        case 1: printf("%d\n", index_last_negative((a+1), size)); break;
        case 2: printf("%d\n", multi_between_negative((a+1), size)); break;
        case 3: printf("%d\n", multi_before_and_after_negative((a+1), size)); break;
        default: printf("Данные некоректны");
    }

    return 0;
}
```

menu.c" 43L, 879C

2) index_first_negative.c

```
#include "index_first_negative.h"

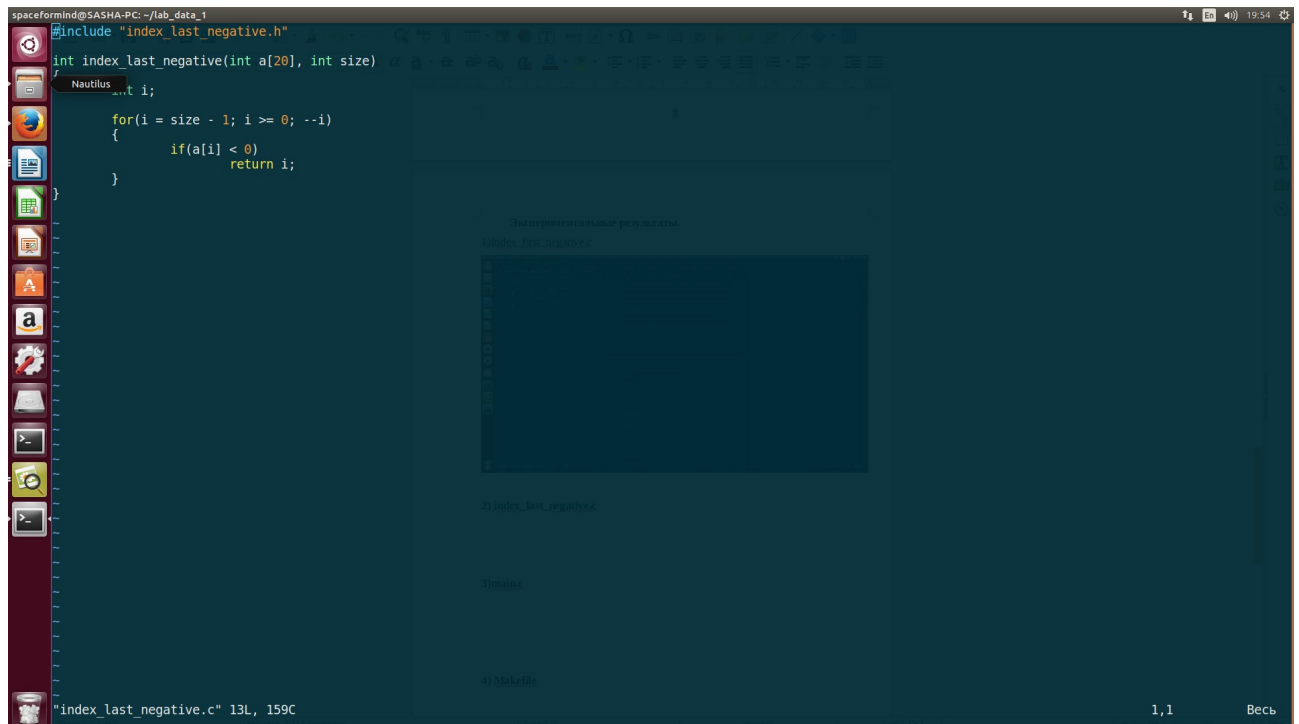
int index_first_negative(int a[20], int size)
{
    int i;

    for(i = 0; i < size; ++i)
    {
        if(a[i] < 0)
            return i;
    }

    return -1;
}
```

index_first_negative.c" 12L, 155C

3) index_last_negative.c



```
#include "index_last_negative.h"

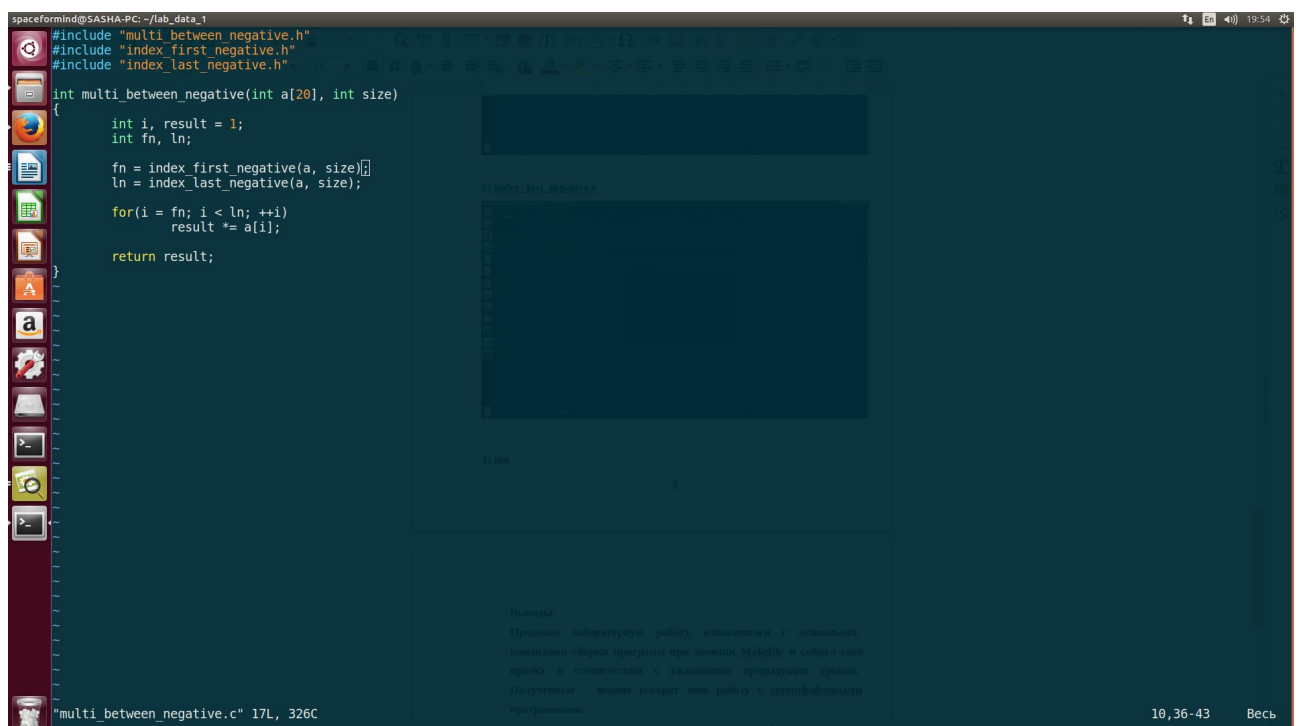
int index_last_negative(int a[20], int size)
{
    int i;

    for(i = size - 1; i >= 0; --i)
    {
        if(a[i] < 0)
            return i;
    }
}
```

index_last_negative.c" 13L, 159C

1,1 Весь

4) multy_between_negative.c



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

int multy_between_negative(int a[20], int size)
{
    int i, result = 1;
    int fn, ln;

    fn = index_first_negative(a, size);
    ln = index_last_negative(a, size);

    for(i = fn; i < ln; ++i)
        result *= a[i];

    return result;
}
```

"multy_between_negative.c" 17L, 326C

10,36-43 Весь

5) multy_before_and_after_negative.c

```
spaceformind@SASHA-PC: ~/lab_data_1
#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[20], int size)
{
    int i, result = 1;
    int fn, ln;

    fn = index_first_negative(a, size);
    ln = index_last_negative(a, size);

    for(i = 0; i < fn; ++i)
        result *= a[i];

    for(i = ln; i < size; ++i)
        result *= a[i];

    return result;
}

"multi_before_and_after_negative.c" 20L, 391C
```

6) Makefile

```
spaceformind@SASHA-PC: ~/lab_data_1
all: menu

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

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

"Makefile" 21L, 936C
```

Выводы.

Проделав лабораторную работу, применил на практике конструкцию `switch(key) {case <value1>: ... break; ... ; default: ...; break;}` и написал код без

повторяющихся инструкций(использовал вызовы уже существующих функций для получения значений). Для определения размера массива использовал новый для себя инструмент(возвращаемое значение функции scanf). Прделанная работа-хороший опыт в написании программ и их структурировании.