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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: Условия, циклы, оператор switch

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Отработать на практике базовые принципы построения программ. Изучить и научиться работать с такими основными управляющими конструкциями языка Си, как условия, циклы, оператор множественного выбора switch.

Задание.

Напишите программу, выделив каждую подзадачу в отдельную функцию.

Реализуйте программу, 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)

1 : индекс последнего отрицательного элемента. (index_last_negative)

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

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

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>]
    }
```

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.

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. Поэтому сначала инициализируем переменные *int a=index_first_negative(arr,arr_size)* и *int b=index_last_negative(arr,arr_size),* в которые поместятся возвращенные соответствующими функциями значения индексов первого и последнего отрицательного элемента массива. Затем будет вызвана функция *multi_between_negative(arr, arr_size,a, b),* 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,a, b)).* 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. Поэтому сначала инициализируем переменные *int a=index_first_negative(arr,arr_size)* и *int b=index_last_negative(arr,arr_size),* в

которые поместятся возвращенные соответствующими функциями значения индексов первого и последнего отрицательного элемента массива. Затем будет вызвана функция *multi_before_and_after_negative(arr,arr_size, a, b),* 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, a, b)).* 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.*

8.Определим каждую функцию, соответствующую определенной подзадаче, то есть для каждой укажем тип возвращаемого значения, имя функции, информацию о получаемых аргументах, а также тело самой функции.

1) Первая 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) Следующая 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 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).* Каждый элемент, удовлетворяющий условию, будем считать «претендентом» на последний отрицательный элемент, поэтому во время итерации цикла при выполнении условия будем записывать индекс этого «претендента» в переменную *int i_last*(инициализируем ее еще в начале тела функции). Таким образом, перебрав все элементы массива, в переменной *i_last* останется значение индекса последнего отрицательного элемента массива. Следующим шагом

возвращаем значение переменной i_last(искомое значение последнего отрицательного элемента массива), с помощью оператора return .

3) Третья 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,int a,int b)*

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*.
Также для решения нам понадобятся индексы первого и последнего отрицательных элементов массива. Передадим их в целочисленные переменные *int a* и *int b* соотвественно(о том, каким образом передаю индексы в функцию, см. пункт 7 case2/case3)

Тело функции: 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) Четвертая 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,int a,int b)*

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*.
Также для решения нам понадобятся индексы первого и последнего отрицательных элементов массива. Передадим их в целочисленные переменные *int a* и *int b* соотвественно(о том, каким образом передаю индексы в функцию, см. пункт 7 case2/case3)

Тело функции: 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 фунции*).* После выполнения двух циклов for, мы получим итоговое значение искомого произведения. Вернем это значение с помощью оператора return.

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 базовые принципы построения программ, а также исследованы и изучены основные управляющие конструкции языка: условия, циклы, оператор switch.

Разработана программа, выполняющая считывание с клавиатуры команды пользователя и исходного целочисленного масива. 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.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
int index_first_negative(int arr[], int n){ // n - размер массива
    for(int i=0;i<n;i++){
        if(arr[i]<0){
            return i;
        }
    }
}
int index_last_negative(int arr[], int n){
    int i_last=-1;
    for(int i=0;i<n;i++){
       if(arr[i]<0)i_last=i;
    }
    return i_last;
}
int multi_between_negative(int arr[],int n,int a,int b){ //a - индекс первого отриц.элемента массива; b - индекс последнего отриц.элемента массива;
    int p=1;
    for(int i=a;i<b;i++)p*=arr[i];
    return p;
}

int multi_before_and_after_negative(int arr[],int n,int a,int b){
    int p=1;
    for(int i=0;i<a;i++)p*=arr[i];
    for(int i=b;i<n;i++)p*=arr[i];
    return p;
}
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: {
            int a=index_first_negative(arr,arr_size);
            int b=index_last_negative(arr,arr_size);
            printf("%d\n", multi_between_negative(arr,arr_size,a,b));
            break;
        }
        case 3: {
            int a=index_first_negative(arr,arr_size);
            int b=index_last_negative(arr,arr_size);
            printf("%d\n",
multi_before_and_after_negative(arr,arr_size,a,b));
            break;
        }
        default: {
            printf("%s\n","Данные некорректны");
            break;
        }
    }
}
```