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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование на Си»

Тема: Условия, циклы, оператор switch

Студент гр. 0382 _____ Азаров М.С.

Чайка К.В. ,

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2018

Цель работы.

Целью работы является освоение работы с управляющими конструкциями на языке С на примере использующей их программы

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".

Ошибкой в данном задании считается дублирование кода!

Подсказка: функция нахождения модуля числа находится в заголовочном файле stdlib.h стандартной библиотеки языка Си.
При выводе результата, не забудьте символ переноса строки

Пример

Test	Input	Result
#1	0 -5 -3 -5 -8 3 -9 -3	0

Выполнение работы.

1. Подключение библиотеки <stdio.h>.

2. Макроопределение **L** (максимальной длины массива вводимых данных).
3. Описание функции **getstr** , которая считывает вводимые данные в массив **arr** (который является аргументом функции) и возвращает количество введенных элементов.
   3.1. Инициализация переменных **ch** (вводимый символ) , **i** (счетчик цикла).
   3.2. Цикл интегрирующийся до тех пор по не будет введен **\n** (символ переноса строки). Цикл считывает текущий элемент ввода в **i**-тый элемент массив **arr[i]** , и затем считывает **пробел** или **\n** в переменную **ch**.
   3.3. Возвращение значение функции соответствующее **i** , как количеству введенных элементов.
4. Описание функции **idx_frst_otr** , которая находит индекс первого отрицательного элемента массива **arr** и возвращает это значение вместо себя. На вход функции поступают массив **arr** и длина массива **len.**
   4.1. Инициализация переменных **i** (счетчик цикла), **bol** (переменная определяющая нахождение отрицательного элемента в массиве **arr**, если **bol** = 1 — элемент найден , 0 — элемент не найден), **res** (результат вычислений функции , а именно значение первого отрицательного элемента массива **arr**).
   4.2. Цикл интегрирующийся до тех пор по не закончатся элементы массива **arr** или не будет найден первый отрицательный элемент массива **arr**. Цикл сравнивает текущее значение массива **arr[i]** с **0** , и если оно меньше 0 то присваивает номер текущего элемента **i** переменной **res.**
   4.3. Если (первый или последний) отрицательный элемент был найден возвращается **res** индекс этого элемента , в противном случае

возвращается **-1** , что означает отрицательный элемент не был найден.

5. Описание функции **idx_last_otr** , которая находит индекс последнего отрицательного элемента массива **arr** и возвращает это значение вместо себя. На вход функции поступают массив **arr** и длина массива **len.**
   5.1. Инициализация переменных **i** (счетчик цикла), **bol** (переменная определяющая нахождение отрицательного элемента в массиве **arr**, если **bol** = 1 — элемент найден , 0 — элемент не найден), **res** (результат вычислений функции , а именно значение первого отрицательного элемента массива **arr**).
   5.2. Цикл интегрирующийся до тех пор по не закончатся элементы массива **arr** . Цикл сравнивает текущее значение массива **arr[i]** с **0** , и если оно меньше 0 то присваивает номер текущего элемента **i** переменной **res.** Если после этого будет найден новый отрицательный элемент массива ,то новый индекс элемента переприсвоется в переменную **res**.
   5.3. Если (первый или последний) отрицательный элемент был найден возвращается **res** индекс этого элемента , в противном случае возвращается **-1** , что означает отрицательный элемент не был найден.
6. Описание функции **mult_btwn_otr** , которая считает произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент) и возвращает это значение вместо себя. На вход функции поступают массив **arr** , длина массива **len** и переменная успешности завершения функции **err**.
   6.1. Инициализация переменных **from** (индекс элемента с которого начинается умножение) , **to** (индекс элемента на котором

заканчивается умножение) , **i** (счетчик цикла) ,**res** (результат вычислений функции).

6.2. Присвоение **res** начального значения **1** для последующего произведение . Присвоение **err = 0** , как начального значения успешности операции (0 — программы выполнена успешно , 1 — не нашлось отрицательного числа).

6.3. Присвоение **from** индекса первого отрицательного элемента массива **arr** , используя функцию **idx_frst_otr**(пункт 4) . Присвоение **to** индекса последнего отрицательного элемента массива **arr** , используя функцию **idx_last_otr**(пункт 5)

6.4. Если **to** или **from** равны **-1** значит отрицательных чисел в массиве нет , данные некорректны , переменной **err = 1** , что означает не нашлось отрицательного числа. В противном случае циклом считаем произведение от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент) запоминая это значение в переменную **res** , и возвращая ее значение в функцию.

7. Описание функции **mult_bef_and_af_otr** , которая считает произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент) и возвращает это значение вместо себя. На вход функции поступают массив **arr** , длина массива **len** и переменная успешности завершения функции **err**.

7.1. Инициализация переменных **from** (индекс элемента с которого начинается умножение) , **to** (индекс элемента на котором заканчивается умножение) , **i** (счетчик цикла) ,**res** (результат вычислений функции).

7.2. Присвоение **res** начального значения **1** для последующего произведение . Присвоение **err = 0** , как начального значения

успешности операции (0 — программы выполнена успешно , 1 — не нашлось отрицательного числа).

7.3. Присвоение **from** индекса первого отрицательного элемента массива **arr** , используя функцию **idx_frst_otr**(пункт 4) . Присвоение **to** индекса последнего отрицательного элемента массива **arr** , используя функцию **idx_last_otr**(пункт 5)

7.4. Если **to** или **from** равны **-1** значит отрицательных чисел в массиве нет , данные некорректны , переменной **err = 1** , что означает не нашлось отрицательного числа. В противном случае циклом считаем произведение до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент) запоминая это значение в переменную **res** , и возвращая ее значение в функцию.

8. Описание основной функции **main** .

8.1. Инициализация переменных **flag**(значение команды действия), **len** (длина массива) , **i** (счетчик цикла) , **res** (результат вычислений подфункций), **err** (переменная успешности завершения подфункций) и массива **arr** длиной **L** ,для вводимых данных.

8.2. Считывание значения команды в **flag**.

8.3. Считывание вводимых данных в массив **arr** и присвоение **len** количества введенных элементов с помощью функции **getstr** (пункт 3).

8.4. Опираясь на значение **flag**(переменая команды действий) выполнение соответствующих действий .

8.4.1. Если **flag** равно **0** (команда нахождения индекс первого отрицательного элемента) , то вызывается функции **idx_frst_otr** (пункт 4) и присвоение ее значения **res**. Если **res** равно **-1**(не нашелся отрицательный элемент) ,то вывод строки «Данные некорректны», иначе вывод значения **res**.

8.4.2. Если **flag** равно **1** (команда нахождения индекс последнего отрицательного элемента) , то вызывается функции **idx_last_otr** (пункт 5) и присвоение ее значения **res**. Если **res** равно **-1**(не нашелся отрицательный элемент) ,то вывод строки «Данные некорректны», иначе вывод значения **res**.

8.4.3. Если **flag** равно **2** (команда нахождения произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент)) , то вызывается функции **mult_btwn_otr** (пункт 6) и присвоение ее значения **res**. Если **err** равно **1**(не нашелся отрицательный элемент) ,то вывод строки «Данные некорректны», иначе вывод значения **res**.

8.4.4. Если **flag** равно **3** (команда нахождения произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент)) , то вызывается функции **mult_bef_and_af_otr** (пункт 7) и присвоение ее значения **res**. Если **err** равно **1**(не нашелся отрицательный элемент) ,то вывод строки «Данные некорректны», иначе вывод значения **res**.

8.4.5. Если **flag** не равен одному из этих значений выводится «Данные некорректны»

Разработанный программный код см. вприложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 -4 5 7 -5 4	0	Программа работает правильно
2.	1 -3 5 -6 83 40 59 4	2	Программа работает правильно
3.	2 5 3 -6 3 5 -1 -6 9	90	Программа работает правильно
4.	2 84 54 9 3 -3 -90 32	-3	Программа работает правильно
5.	3 4 4 6 -1 4 9 -3	-288	Программа работает правильно
6.	3 4 -4 1	-16	Программа работает правильно
7.	5 9 39 -4 9	Данные некорректны	Программа работает правильно
8.	0 34 6 4 35 6	Данные некорректны	Программа работает правильно

Выводы.

Были изучена и освоена работа с управляющими конструкциями на языке С на примере использующей их программы

Разработана программа, выполняющая считывание с клавиатуры исходных данных и команды пользователя. Для считывания данных и вывода их в консоль были использованы функции стандартной библиотеки *printf()* и *scanf()*. Для декомпозиции кода были созданы собственные функции *setstr()* для считывания данных , *idx_frst_otr()* для нахождение первого отрицательного элемента последовательности , *idx_last_ost*() для нахождения последнего элемента последовательности , *mult_btwn_otr*() для нахождения произведения элементов последовательности от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент), *mult_bef_and_af_otr*() для нахождения произведения элементов последовательности расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент). Также был применен оператор switch для создания вариативности хода программы. И цикл for для n-ого повторения одинаковых действий .

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Azarov_Maksim_lb1.c

```
#include <stdio.h>

#define L 20


int getstr(int arr[]){

      char ch = 0;
      int i ;

      for (i = 0; ch != '\n'; i++){
           scanf("%i",&arr[i]);
           ch = getchar();
      }

      return i;

}



int idx_frst_otr(int arr[],int len){
      int i, bol = 0,res;

      for (i = 0; (i < len)&&(bol == 0);++i)
           if (arr[i] < 0){
                res = i;
                bol = 1;
           }
      if (bol == 1) return res;
      else return -1;


}


int idx_last_otr(int arr[],int len){
      int i,bol = 0,res;

      for (i = 0; i < len ;++i)
              if (arr[i] < 0){
                    res = i;
                   bol = 1;
              }
     if (bol == 1) return res;
     else return -1;



}
```

```
int mult_btwn_otr(int arr[],int len,int *err){
      int from,to,i,res;
      res = 1;
      *err = 0;
      from = idx_frst_otr(arr,len);
      to = idx_last_otr(arr,len);

      if ((to == -1)||(from == -1)){
           *err = 1;
            return 0;
      }
     else {
            for (i = from; i < to; ++i)
                  res = res * arr[i];
            return res;
      }
}


int mult_bef_and_af_otr(int arr[],int len,int *err){
      int from,to,i,res;
      res = 1;
      *err = 0;
      to = idx_frst_otr(arr,len);
     from = idx_last_otr(arr,len);

      if ((to == -1)||(from == -1)){
           *err = 1;
            return 0;
      }
      else{
            for (i = 0; i < to; ++i)
                 res = res * arr[i];
            for (i = from; i < len; ++i)
                  res = res * arr[i];
            return res;
      }
}







int main(){

      int flag,len,i,res,err;
      int arr[L];

      scanf("%d",&flag);

      len = getstr(arr);
```

```
	switch (flag){
		case 0: res = idx_frst_otr(arr,len);
			if (res == -1) printf("Данные некорректны\n");
			else printf("%i\n",res);
			break;

		case 1: res = idx_last_otr(arr,len);
			if (res == -1) printf("Данные некорректны\n");
			else printf("%i\n",res);
			break;

		case 2: res = mult_btwn_otr(arr,len,&err);
			if (err == 1) printf("Данные некорректны\n");
			else printf("%i\n",res);
			break;

		case 3: res = mult_bef_and_af_otr(arr,len,&err);
			if (err == 1) printf("Данные некорректны\n");
			else printf("%i\n",res);
			break;

		default: printf("Данные некорректны\n");
	}

	return 0;
}
```