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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование Си»

Тема: Условия, циклы, оператор switch.

Студент гр. 0382 _____ Ильин Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Получение базовых знаний по языку программирования Си.

Задание.

Вариант 1.

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>]

}

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.

Выполнение работы.

Переменные:

- command- команда, в соответствии с которой программа должна обработать поступающие данные
- return_command- то, что возвращает функция, соответствующая определённой команде
- length- количество поступивших чисел, которые надо обработать
- mass- массив чисел, которые надо обработать
- read- считываемые данные(поэлементно)
- elem- переменная, пи помощи которой заполняется «mass»
- index- индекс элемента «mass», при помощи которого происходит выполнение соответствующих функций
- first- индекс первого отриц. числа в «mass»
- last- индекс последнего отриц. числа в «mass»
- between- произведение элементов массива, расположенных от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент)
- before_and_after- произведение элементов массива, расположенных до первого отрицательного элемента (не включая элемент) и после последнего отрицательного (включая элемент)

Функции:

main – считывает входящие данные, при помощи функции getchar()

Сначала считывает первый элемент, по условию задачи- это и есть команда, далее считывает каждый элемент по отдельности и создаёт массив целых чисел.

Алгоритм создания массива целых чисел:

Пока считываемый элемент не \n берётся следующий элемент из ввода, если он не пробел, то элемент массива соответствующего индекса умножается на 10 и к нему прибавляется считанное число (изначально массив наполнен нулями), если же элемент- пробел, то индекс увеличивается на 1.

Далее при помощи оператора switch вызывается функция соответствующая поданной команде, результат которой печатается.

index_first_negative- принимает на вход массив, в котором ищет первый отрицательный элемент при помощи цикла for.

index_last_negative- принимает на вход массив, в котором ищет последний отрицательный элемент при помощи цикла for.

multi_between_negative- принимает на вход массив, перемножает элементы массива, расположенные от первого отрицательного элемента (включая элемент) и до последнего отрицательного (не включая элемент)

multi_before_and_after_negative- принимает на вход массив и его длину, перемножает элементы массива, расположенные до первого отрицательного элемента (не включая элемент) и от последнего отрицательного (включая элемент)

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	Первый отрицательный элемент массива, имеет

			индекс ноль.
2.	2 124 513 -2 2 2 2 -14 512	-16	Произведение элементов массива, стоящих между первым(включая) и последним(не включая) отрицательным равна -16.
3.	2 51 -2 -2 2 2 -2 13256 234	16	Произведение элементов массива, стоящих между первым(включая) и последним(не включая) отрицательным равна 16.
4.	3 11 -623 -512 -25 125 54 3 2 -5 1 -1 11 121	-14641	Произведение элементов массива, стоящих до первого(не включая) и после последнего(включая) отрицательного равна -14641.

Выводы.

Были изучены основы языка C, 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-else и циклы while, for.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: prlb1.c

```
#include <stdio.h>

int index_first_negative(int mass[20]) {
    int index, first;
    first = -1;
    for (index = 0; index < 20; index++) {
        if ((first == -1) && (mass[index] < 0)){
            first = index;
        }
    }
    return first;
}

int index_last_negative(int mass[20]) {
    int index, last;
    last = -1;
    for (index = 19; index >= 0; index--) {
        if ((last == -1) && (mass[index] < 0)){
            last = index;
        }
    }
    return last;
}

int multi_between_negative(int mass[20]){
    int index, between, first, last;
    between = 1;
    first = index_first_negative(mass);
    last = index_last_negative(mass);
    for (index = first; index < last; index++) {
        between = between * (mass[index]);
    }
    return between;
}

int multi_before_and_after_negative(int mass[20], int length) {
    int index, before_and_after, first, last;

    before_and_after = 1;
    first = index_first_negative(mass);
    last = index_last_negative(mass);

    for (index = 0; index < first; index++) {
        before_and_after = before_and_after * (mass[index]);
    }

    for (index = last; index < length; index++) {
        before_and_after = before_and_after * (mass[index]);
    }

    return before_and_after;
}
```

```
int main() {
    int return_command, command, length, elem;
    command = getchar();
    length = 0;
    char read = getchar();
    int mass[20] = {0};
    /*printf("%d\n", a);*/

    while (read != '\n'){
        read = getchar();

        if (read == '-'){
            while ((read != ' ') && (read != '\n')){
                if (read == '-'){
                    read = getchar();
                }
                else {
                    elem = read - '0';
                    /*printf("%d\n", mass[length]);*/
                    mass[length] = mass[length]*10 - elem;
                    /*printf("%d\n", mass[length]);*/
                    read = getchar();
                }
            }
        }
        else {
            while ((read != ' ') && (read != '\n')){
                elem = read - '0';
                /*printf("%d\n", mass[length]);*/
                mass[length] = mass[length]*10 + elem;
                /*printf("%d\n", mass[length]);*/
                read = getchar();
            }
        }
        ++length;

    }

    switch(command){
        case '0':
            return_command = index_first_negative(mass);
            printf("%d", return_command);
            break;
        case '1':
            return_command = index_last_negative(mass);
            printf("%d", return_command);
            break;
        case '2':
            return_command = multi_between_negative(mass);
            printf("%d", return_command);
            break;
        case '3':
             return_command = multi_before_and_after_negative(mass,
length);
            printf("%d", return_command);
            break;
        default:
            printf ("Данные некорректны");
            break;
```

```
    }
    return 0;

}
```