**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)**

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Дисциплина: «Машинно-зависимые языки программированияе»

Лабораторная работа №4.

**Тема: «Обработка одномерных массивов»**

Выполнил:

Студент группы ИКПИ-21

Козлова А. И.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

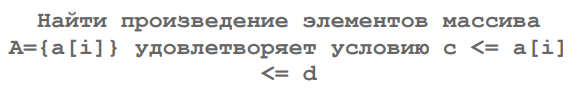
Старший преподаватель кафедры ПИиВТ

Анохин Ю. В.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**А. *Постановка задачи***

Задав одномерный массив целочисленных данных А в одном из заданных форматов (unsigned char — BYTE, unsigned short int — WORD, char — SHORTEST, short int — INTEGER или long int — LONGINT), реализовать обработку массива, как указано в варианте. Длина массива N. Исходные данные задать самостоятельно, учитывая формат элементов массива А. В программе на C++ должны быть предусмотрены функции ввода - вывода элементов массива А и его обработки. Исходные данные должны вводиться корректно и быть приближенными к максимально возможным для данного типа данных. Тип результата определяется из контекста задачи.



**Б. *Разработка алгоритма***

С помощью глобальных переменных в C и external-объявленных переменных будет осуществляться обмен данными между модулями на Assembler и C. С помощью операций mov, div, sub, add, модуль на Assembler будет высчитывать необходимые значения.

**В. *Описание программы***

Варианты программы для разных типов данных. Программа принимает на вход одномерный массив, затем выдаёт результат, высчитанный на С и на Assembler

**Д. Контрольный расчёт**

Массив: 1 2 3 4 5

Границы: 2 и 4

2\*3\*4 = 24

**Д. *Результаты работы программы***

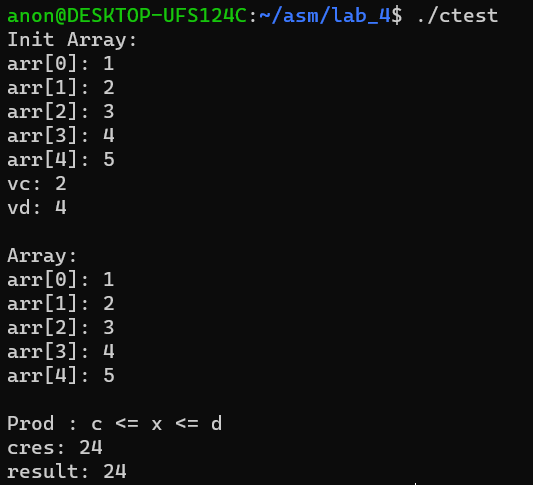
******

рис 1. Результаты работы программы

**E. Код программы.**

**C**

#include <stdio.h>  
#include <inttypes.h>  
  
extern uint8\_t vn;  
extern int8\_t vc, vd, arr[6];  
extern int64\_t result;  
extern void arrproc(void);  
int main()  
{  
 vn = 5;  
 printf("Init Array:\n");  
 for (uint8\_t i = 0; i < vn; i++)  
 {  
 printf("arr[%hhd]: ", i);  
 scanf("%hhd", arr + i);  
 }  
 printf("vc: ");  
 scanf("%hhd", &vc);  
 printf("vd: ");  
 scanf("%hhd", &vd);  
 uint64\_t cres = 1;  
 int8\_t ai;  
 printf("\nArray:\n");  
 for (uint16\_t i = 0; i < vn; i++)  
 {  
 ai = \*(arr + i);  
 if ((vc <= ai) && (ai <= vd))  
 cres \*= ai;  
 printf("arr[%hhd]: %hhd\n", i, ai);  
 }  
 arrproc();  
 printf("\nProd : c <= x <= d\ncres: %lu\n", cres);  
 printf("result: %lu\n", result);  
 return 0;  
}

**Asm**

global result  
global arr  
global vc  
global vd  
global vn  
section .bss  
 result resb 8  
 vn resb 1  
 arr resb 6  
 vc resb 1  
 vd resb 1  
section .text  
global arrproc  
  
arrproc:  
 mov rax, 1  
 push rbx  
 xor rbx, rbx  
 xor rcx, rcx  
 xor rdx, rdx  
 movsx rdi, byte [vd]  
 movsx rsi, byte [vc]  
 mov bl, [vn]  
mulforn:  
 cmp rbx, rcx  
 je end  
 movsx rdx, byte [arr+rcx]  
 inc rcx  
 cmp rdx, rsi  
 jge gec  
 jmp mulforn  
gec:  
 cmp rdx, rdi  
 jle led  
 jmp mulforn  
led:  
 imul rdx  
 jz endz  
 jmp mulforn  
end:  
 mov [result], rax  
 pop rbx  
 ret  
endz:  
 mov qword [result], 0  
 pop rbx  
 ret