

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А.  
БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

**ЛАБОРАТОРНАЯ РАБОТА №4**

**«ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ»**

по дисциплине «Машинно-зависимые языки программирования»

Выполнил:

студент 2 курса

дневного отделения

группы ИКПИ-23

Даненко Д. А.

Санкт-Петербург 2023

### А. Постановка задачи

Задав одномерный массив целочисленных данных в одном из заданных форматов (signed/unsigned char — byte, signed/unsigned int— word, signed/unsigned long — dword), реализовать обработку массива, как указано в варианте. Длина массива N. Исходные данные задать самостоятельно, учитывая формат элементов массива А.

В программе должны быть предусмотрены функции ввода-вывода элементов массива и его обработки. Исходные данные должны вводиться корректно (организовать проверку). Тип результата определяется из контекста задачи.

Найти сколько положительных элементов  
массива  $A=\{a[i]\}$  удовлетворяет условию  $c$   
 $\leq a[i] \leq d$

### Б. Таблица идентификаторов

N	Обозначение в задаче	Идентификатор	Назначение
1	A	A	Входные данные
2	c	c	
3	d	d	
4	N	n	Промежуточные данны
6	Результат	asm count	Выходные данные

### В. Таблица результатов

Результаты вычислений приведены ниже в таблице вычислений.

Тип	C	D	Массив	X
Signed Word	-4	8	5 0 -3 9	NASM: 2 C: 2
	-645	256	-700 0 800	NASM: 1 C: 1
	712	713	34 -346 0 712 713 714 100	NASM: 2 C: 2

```
> $ ./c
n = 4
c = -4
d = 8
A[0] = 5 0 -3 9

A[1] = A[2] = A[3] = c >>>2
asm>>>2
```

```
> $ ./c
n = 3
c = -645
d = 256
A[0] = -700
A[1] = 0
A[2] = 800
c >>>1
asm>>>1
```

```
> $ ./c
n = 7
c = 712
d = 713
A[0] = 34
A[1] = -346
A[2] = 0
A[3] = 712
713
714
100
A[4] = A[5] = A[6] = c >>>2
asm>>>2
```

// простите мне мою поспешность в вводе значений в массив

// до девяти вечера всего ничего

## Д. Программа

### ФАЙЛ C

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
extern void asm_func(void);
```

```
int16_t A[128];
uint8_t n; // 0<=N<=127
int16_t c, d;
uint16_t asm_count = 0;
```

```
int
c_func(int16_t c, int16_t d)
{
    int count = 0;
    for (int i = 0; i < n; ++i) {
        if ((A[i] >= 0) && (A[i] >= c) && (A[i] <= d)) {
            count++;
        }
    }
    return count;
}
```

```
int
main(int argc, char *argv[])
{
    // Найти сколько положительных элементов
    // массива A={a[i]} удовлетворяет условию
    // c <= a[i] <= d
```

```
printf("n = "); scanf("%d", &n);
printf("c = "); scanf("%d", &c);
```

```

printf("d = "); scanf("%d", &d);

for (int i = 0; i < n; ++i) {
    printf("A[%d] = ", i);
    scanf("%d", &A[i]);
}

printf(" C >>>");
printf("%d\n", c_func(c, d));

asm_func();
printf("asm>>>");
printf("%d\n", asm_count);

return 0;
}

```

### ФАЙЛ ASM

```

section .data
    extern A, n, c, d
    extern asm_count
section .text
    global asm_func

asm_func:
    xor     eax, eax    ; \
    xor     ebx, ebx    ; \
    xor     ecx, ecx    ; \
    xor     edx, edx    ; cleaning

    mov     bx,  [n]      ; bx = n
    mov     cx,  0        ; cx = i = 0
@cycle:
    cmp     cx,  bx      ; if i==n {
    je      @end         ; exit } else {
    mov     ax,  [A+ecx*2] ; ax = A[i]
@first_condition:
    mov     dx,  0
    cmp     ax,  dx
    jnl     @second_condition
    inc     cx
    jmp     @cycle
@second_condition:
    mov     dx,  [c]      ; dx = c
    cmp     ax,  dx      ; if A[i]>=c {
    jnl     @third_condition ; go to second } else {
    inc     cx            ; i++
    jmp     @cycle        ; go to beggining }
@third_condition:
    mov     dx,  [d]      ; dx = d

```

```

cmp    ax,  dx    ; if A[i]<=d {
jng    @conditions_met    ; go to met } else {
inc    cx                ; i++
jmp    @cycle            ; go to beggining }
@conditions_met:
xor    eax,  eax
mov    eax,  [asm_count] ; ax = asm_count
inc    eax                ; ax++
mov    [asm_count], eax   ; asm_count = ax
xor    eax,  eax
inc    cx                ; i++
jmp    @cycle            ; go to beggining
@end:
ret

```

### **Е. Выводы**

Тот факт, что результаты, выполненные на NASM, достаточно совпадают с результатами, полученными на С, свидетельствует о том, что программа составлена корректно.