# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе № 6**

# по дисциплине «Организация ЭВМ и систем»

# Тема: **Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы**

Студент гр. 1304 Басыров В.А.

Преподаватель Кирьянчиков В. А.

Cанкт-Петербруг,2022.

**Цель работы.**

Написать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы, используя связь ассемблера с ЯВУ.

**Задание.**

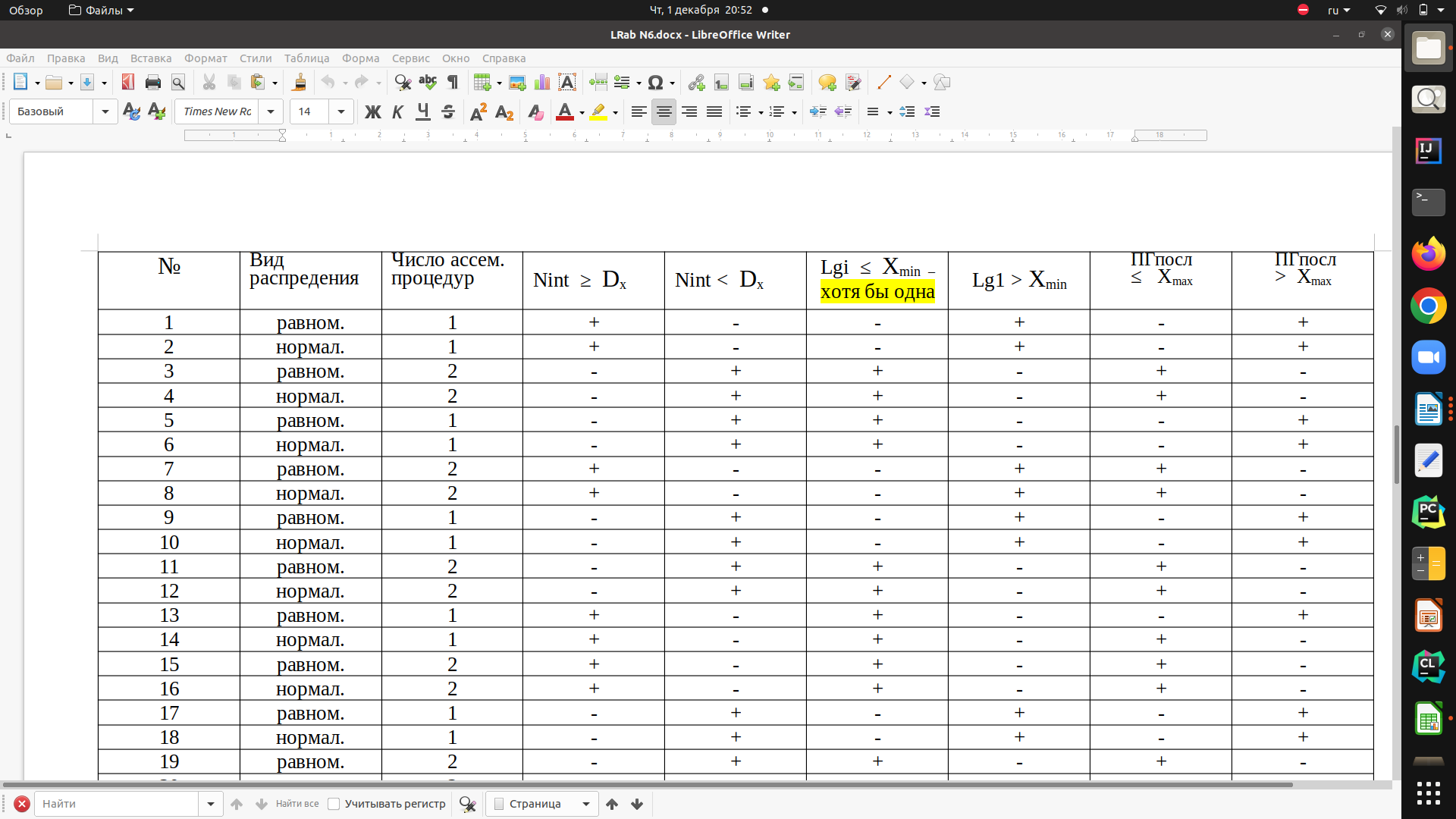
На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND\_GEN (пpи его отсутствии получить у пpеподавателя).

Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

Результирующий массив частотного распределения чисел по интервалам, сфор-мированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

**Вариант 4.**

**Ход работы.**

Main.cpp

Вначала программы согласно условиям задачи просходит считывание всех необходимых данных,которые вводит пользователь,с их обязательной проверкой.В случае некорректного ввода программа завершается и выводит ошибку.

Далее происходит сортировка левых границ пузырьком А про qsort в стандартной библиотеке ты не знаешь? Но это не замечание, можно не исправлять.

for (int i = 0; i < n\_int + 1; i++) {

for (int j = i; j < n\_int + 1; j++) {

if (intervals[i] > intervals[j]) {

std::swap(intervals[i], intervals[j]);

}

}

}

Далее происходит иницализация массива рандомными числами в пределах [Xmin,Xmax]. Для этого используются функции из папки RAND\_GEN. Сначала происходит начальная иницализцаия dnk\_randomize(). Затем,согласно варианту, с помощью функциции dnk\_normal происходит иницализация каждого элемента нормальным распределением. В качестве мат. ожидания берется среднее арифметическое между xmin и xmax, а дисперсия — треть от разности xmax и xmin. При таких значениях шанс,что число попадет в нужный диапазон составляет 99.72% Одно число всего или для каждого из n (например, 16000)? . Чтобы исключить ситуацию попадания случайного числа вне интервала,происходит проверка,что число попала в заданный интервал, если число не попало, то происходит дополнительная итерация прохода цикла.

dnk\_randomize();//начальная иницализация

for (int i = 0; i < n; i++) {

numbers[i] = (int) round(dnk\_normal((x\_max+x\_min)/2.0, (x\_max-x\_min)/3.0));//вычисление рандомного числа

if (numbers[i]>x\_max || numbers[i]<x\_min)

i--;

std::cout « numbers[i] « " ";

}

std::cout « std::endl;

Далее после инициализации нулями 2 массивов( один нужен для записи результата 1 функции,другой второй функции) происходит выполнение ассемблерных функций и вывод результатов как в файл,так и в консоль.

First.asm.

Это функция предназначена для помещения псевдослучайных чисел в единичные интервала. Функция принимает массив псевдослучайных чисел,его длину,результирующий массив,в который будет записан результат и xmin(минимально возможное случайное число). Все они обьявлена как dword, так как являются значениями int(4 байта).Поэтому в программе будут использоватлься расширенные регистры. Происходит сохранение в стек расширенных регистров и затем помещение в эти регистры наших переменных.

.586   
.MODEL FLAT, C   
.CODE   
  
PUBLIC C first\_dist   
first\_dist PROC C numbers: dword, n: dword, result1: dword, x\_min: dword   
  
push esi;сохранение регистров   
push edi   
  
mov esi, numbers;адрес 1 элемента массива случайных чисел   
mov edi, result1;адрес 1 элемента результата   
mov ecx, n;длина массива

Далее идет цикл. В нем обрабатывается очередной элемент, в eax записывается число, из него вычитается xmin и таким образом получается индекс отрезка,в который должен попасть элемент.Далее в ebx записываем количество чисел,которые попали в этот отрезок до этого,добавляем единицу в ebx и записываем получившийся результат.Далее после окончания цикла восстанавливаются регистры.

lp:   
mov eax, [esi];запись случайного числа   
sub eax, x\_min;получение индекса для итогового массива   
mov ebx, [edi+4\*eax];получение элемента итогового массива   
inc ebx;число попало в ед отрезок=>увеличиваем количесвто на единицу   
mov [edi+4\*eax], ebx;запись в соот ячейку   
add esi, 4;переходим к след элементу   
loop lp;вычесть ecx и если ecx=0 завершить цикл   
pop edi;восстановление регистров   
pop esi   
  
ret   
first\_dist endp   
end

Second.asm.

Это функция предназначена для распределения в ввведенные пользователям границы массива псевдослучайных чисел. Функция принимает массив,полученный после выполнения функции first, а также сами интервалы и массив,в который будет записан результат. Сохраняются расширенные регистры в стек, а также помещаются переменные в регистры.

.586   
.MODEL FLAT, C   
.CODE   
  
PUBLIC C second\_dist   
second\_dist PROC C result1:dword, intervals: dword, result2: dword, x\_max: dword, x\_min: dword, n\_int: dword   
  
push esi   
push edi   
  
mov esi, intervals   
mov edi, result2   
mov ecx, n\_int

Далее идет основной цикл, в eax и ebx записывается левая и правая граница очередного интервала соответственно. Затем происходит проверка,больше ли очередная левая граница минимального возможного случайного числа. Если так,то в метке F2 проверяется,является ли отрезок нулевым,и если нет,то вычитается из этой границы xmin и происходит переход к метке F5. Если отрезок является нулевым,то происходит переход к метке финальной метке F1.Если отрезок меньше xmin, то отчет будет проводиться от 1 левой границе и произойдет переход к метке F5.

G:   
mov eax, [esi] ; левая граница интервала   
mov ebx, [esi + 4] ; правая граница   
  
cmp eax, x\_min ; если левая граница >= x\_min   
jge F2   
mov eax, 0 ; иначе, eax = 0, начало массива result1   
  
sub ebx, x\_min ; если длина интервала = 0   
jle F1   
jmp F5;иначе перейти к обработке не 0 интервала   
  
F2:   
sub ebx, eax ; количество элементов в интервале   
cmp ebx, 0 ; если длина интервала = 0   
je F1   
sub eax, x\_min ; индекс первого элемента из текущего интервала в массиве result1

Затем в метке F5 происходит основная обработка. Сохраняются нужные регистры,высчитывается сколько единичных отрезков нужно пройти, и во вложенном цикле G2 в интервал,который задал пользователь,добавляется очередное количество попаданий случайного числа в единичном интервале.

F5:   
push esi ;сохраняем регистры   
push ecx   
  
mov ecx, ebx ; количество элементов из result1 по которым нужно пройти   
mov esi, result1 ; массив   
mov ebx, 0 ; считает сумму подходящих элементов   
  
G2: ; цикл, считает сумму элементов, входящих в интервал   
add ebx, [esi + 4\*eax];к ebx добавляем количество попаданий в очередной еденичный интервал   
inc eax   
loop G2

Затем восстанавливаются регистры и проверяется,является ли очередной интервал последним,если да,то добавляем еще элемент правой скобки,иначе в результирующий массив в соответствующую ячейку записывается результат.

Далее в метке F1 происходит добавление соответствующим индексным регистрам 4(длины двойного слова).После чего восстанавливаются регистры и функция заверается.

pop ecx;восстановление регистра   
  
  
cmp ecx, 1 ; если обрабатывали не последний элемент, то записываем сумму в массив результат   
jne F3   
add ebx, [esi + 4\*eax] ; иначе добавляем еще элемент последней правой скобки   
  
F3:   
  
mov [edi], ebx ; записываем результат   
pop esi;восстановление регистра   
jmp F1   
  
  
F1:   
add edi, 4 ; двигаемся к след. элементам массивов   
add esi, 4

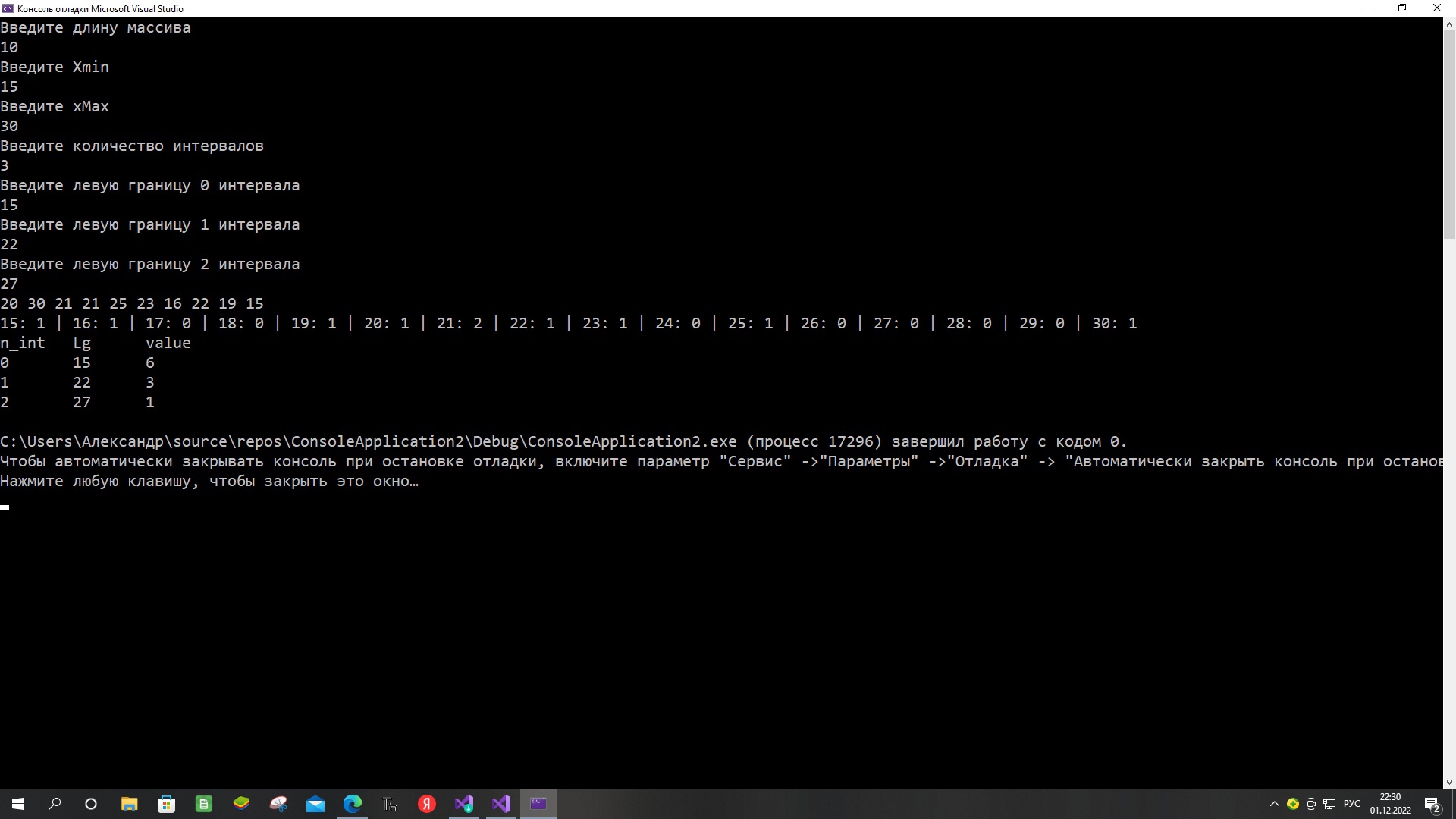
loop G   
  
  
pop edi   
pop esi

ret   
second\_dist ENDP   
END

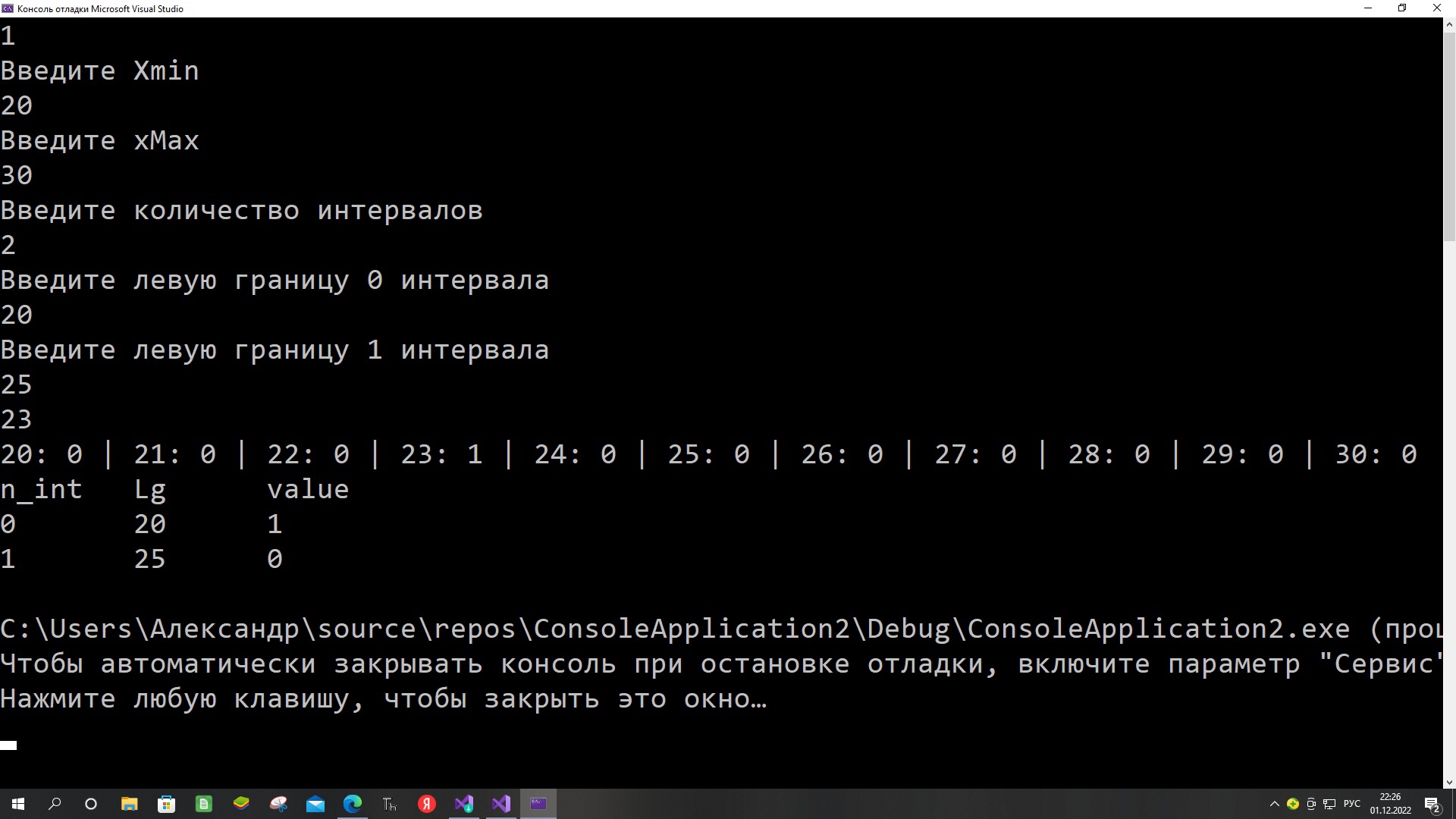
**Тестирование**

Лучше проводить тестирование с большим количеством чисел. Также нужно указать, что конкретно проверяет каждый тест.

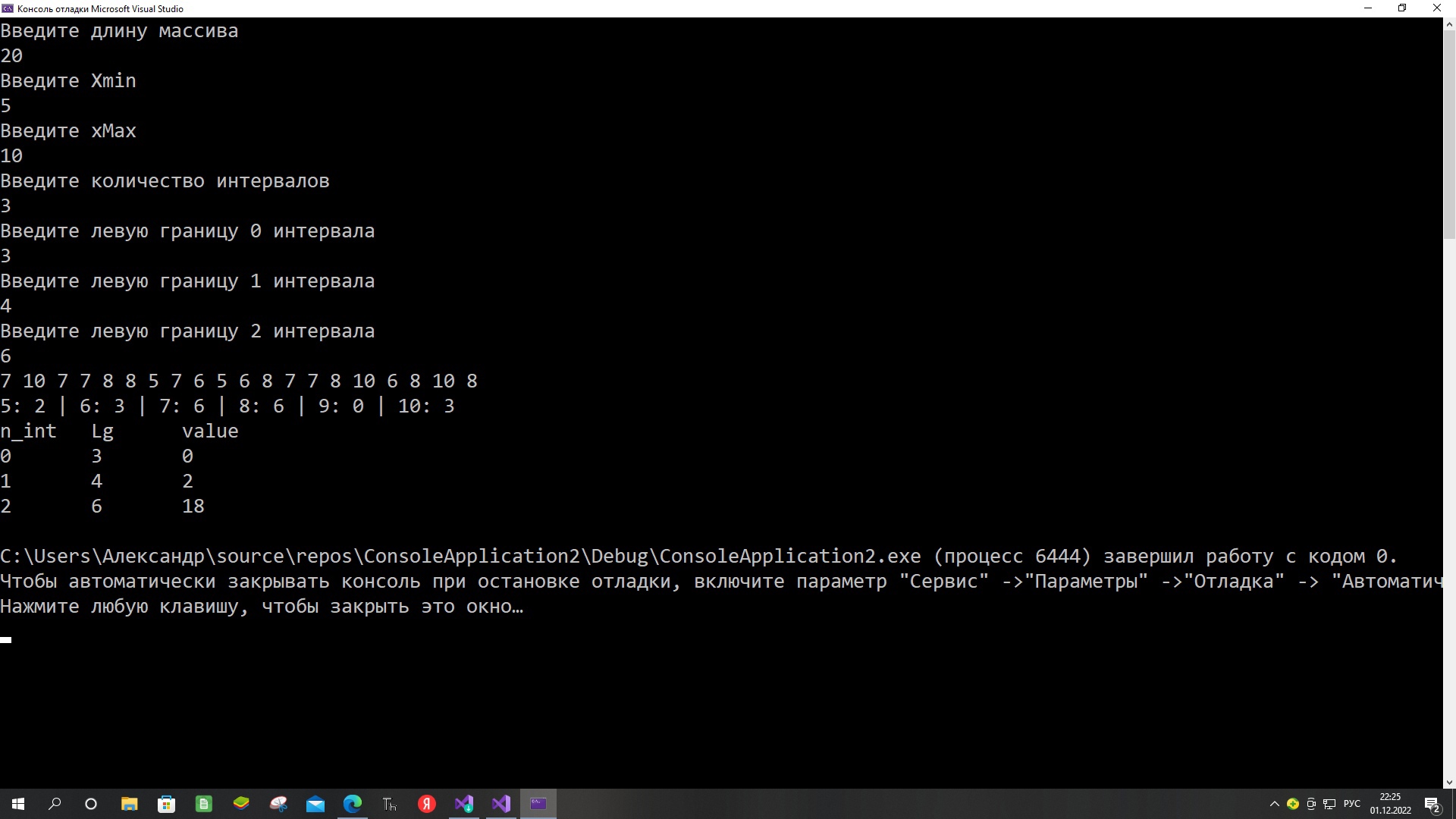
Тест 1. Базовый случай.



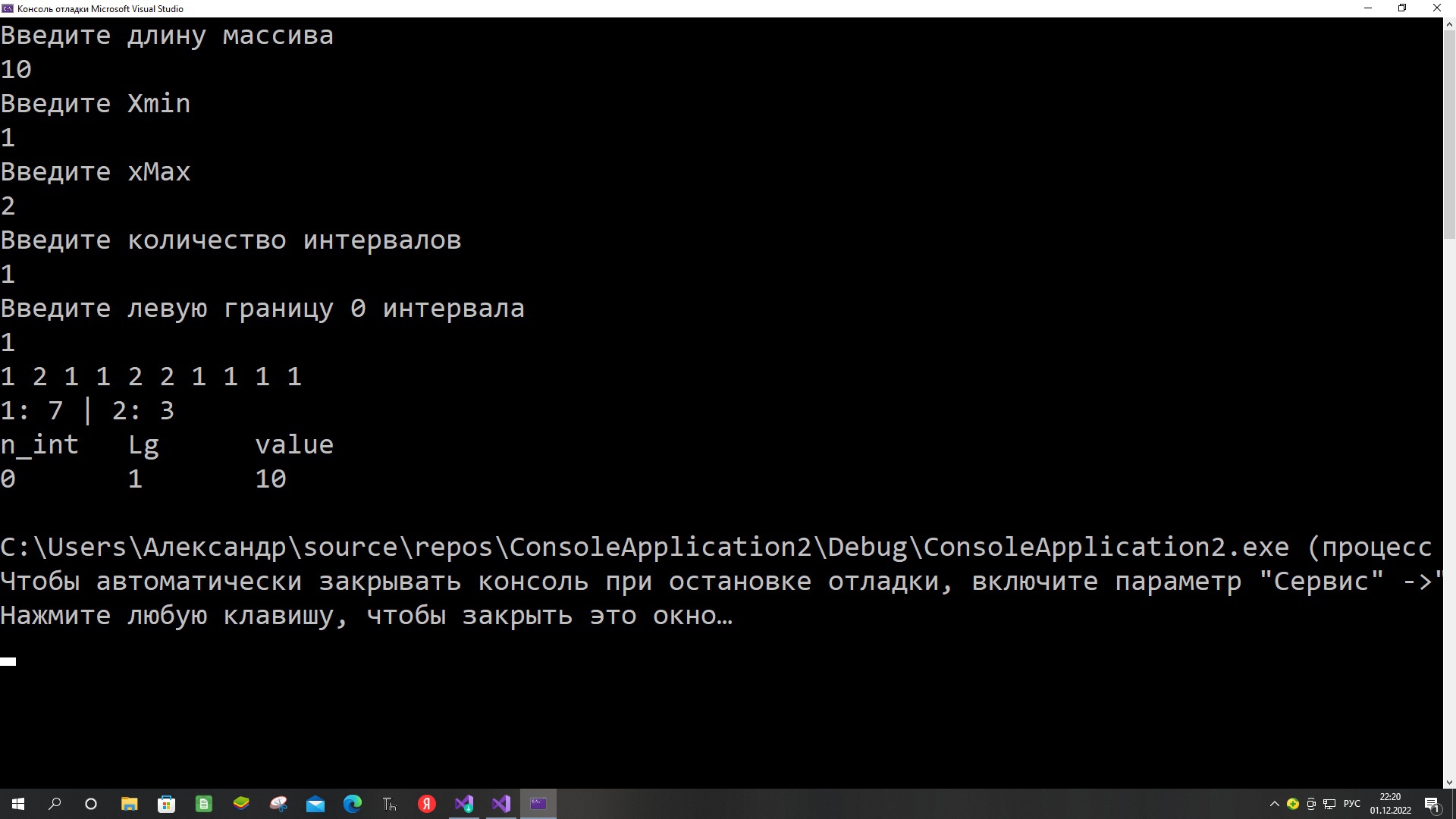
Тест 2. Проверка вырожденного случая, когда 1 элемент у псведоуслучайного массива и левые границы не упорядочены.



Тест 3. Проверка случая,когда левые границы меньше xmin .



Тест 4. Проверка случая,когда 1 левая граница и разность между xmax и xmin равна 1.



Сразу даю вопросы на защиту: внести в программу изменения (независимо друг от друга), объяснить результат их внесения (с указанием причины) и привести возможный тест, на котором изменение можно обнаружить. Изменения см. в листинге ниже.

Приложения.

**Main.cpp**

[#include](https://vk.com/im?sel=161900910&st=%23include) <iostream>   
[#include](https://vk.com/im?sel=161900910&st=%23include) <stdio.h>   
[#include](https://vk.com/im?sel=161900910&st=%23include) <fstream>   
[#include](https://vk.com/im?sel=161900910&st=%23include) <string>   
[#include](https://vk.com/im?sel=161900910&st=%23include) <random>   
[#include](https://vk.com/im?sel=161900910&st=%23include) "Randomer.h"   
[#include](https://vk.com/im?sel=161900910&st=%23include) <locale>   
extern "C" void first\_dist(int\* numbers, int n, int\* result1, int x\_min);   
extern "C" void second\_dist(int\* result1, int\* intervals, int\* result2, int x\_max, int x\_min, int n\_int);   
  
  
int main()   
{   
setlocale(LC\_ALL, "Russian");   
int n, x\_min, x\_max, n\_int;   
std::cout « "Введите длину массива" «std::endl;   
std::cin » n;   
if (n <= 0 || n > 16000) {//проверка на корретный ввод массива   
std::cout « "0 < n <= 16000!!" « std::endl;   
return 0;   
}   
std::cout « "Введите Xmin" « std::endl;   
std::cin » x\_min;   
std::cout « "Введите xMax" « std::endl;   
std::cin » x\_max;   
if (x\_max <= x\_min) {   
  
  
std::cout « "xMax > xMin!" « std::endl;   
return 0;   
}

У тебя в варианте трбуется, чтобы Nint ≥ Dx, где Dx = Xmax - Xmin . Я не вижу проверки на это

std::cout « "Введите количество интервалов" « std::endl;   
std::cin » n\_int;   
if (n\_int <= 0 || n\_int > 24) {   
std::cout « "0 < n\_int <= 24!!" « std::endl;   
system("Pause");   
return 0;   
}   
  
int\* intervals = new int[n\_int + 1];   
for (int i = 0; i < n\_int; i++) {   
std::cout « "Введите левую границу " « i « " интервала" « std::endl;   
std::cin » intervals[i];   
}   
  
intervals[n\_int] = x\_max;   
  
for (int i = 0; i < n\_int + 1; i++) {   
for (int j = i; j < n\_int + 1; j++) {   
if (intervals[i] > intervals[j]) {   
std::swap(intervals[i], intervals[j]);   
}   
}   
}   
  
if (intervals[n\_int] > x\_max + 1) {   
std::cout « "Rg <= x\_max!!" « std::endl;   
return 0;   
}

У тебя в варианте трбуется, чтобы хотя бы одна левая граница была меньше x\_min. Я не вижу проверки на это  
  
int\* numbers = new int[n];   
dnk\_randomize();//начальная иницализация   
for (int i = 0; i < n; i++) {   
numbers[i] = (int) round(dnk\_normal((x\_max+x\_min)/2.0, (x\_max-x\_min)/3.0));//вычисление рандомного числа   
std::cout « numbers[i] « " ";   
}   
std::cout « std::endl;   
  
int\* result1 = new int[abs(x\_max - x\_min) + 1];   
int\* result2 = new int[n\_int];   
for (int i = 0; i < abs(x\_max - x\_min) + 1; i++) {   
result1[i] = 0;   
}   
for (int i = 0; i < n\_int; i++) {   
result2[i] = 0;   
}   
first\_dist(numbers, n, result1, x\_min);   
for (int i = 0; i < n; i++)   
printf("%d ", numbers[i]);   
for (int i = 0; i < abs(x\_max - x\_min); i++) {   
std::cout « i + x\_min « ": " « result1[i] « " | ";   
}   
std::cout « std::to\_string(abs(x\_max - x\_min) + x\_min) « ": " « result1[abs(x\_max - x\_min)] « std::endl;   
second\_dist(result1, intervals, result2, x\_max, x\_min, n\_int);   
  
std::ofstream file("table.txt");   
auto head = "n\_int\tLg\tvalue";   
file « head « std::endl;   
std::cout « head « std::endl;   
for (int i = 0; i < n\_int; i++) {   
auto line = std::to\_string(i) + "\t" + std::to\_string(intervals[i]) + "\t" + std::to\_string(result2[i]) + "\n";   
file « line;   
std::cout « line;   
}   
return 0;   
}

**First.asm**

.586 Почему не .386, .486 или .686?  
.MODEL FLAT, C   
.CODE   
  
PUBLIC C first\_dist   
first\_dist PROC C numbers: dword, n: dword, result1: dword, x\_min: dword   
  
push esi;сохранение регистров   
push edi   
  
mov esi, numbers;адрес 1 элемента массива случайных чисел   
mov edi, result1;адрес 1 элемента результата   
mov ecx, n;длина массива   
  
  
lp:   
mov eax, [esi];запись случайного числа   
sub eax, x\_min;получение индекса для итогового массива   
mov ebx, [edi+4\*eax];получение элемента итогового массива   
inc ebx;число попало в ед отрезок=>увеличиваем количесвто на единицу   
mov [edi+4\*eax], ebx;запись в соот ячейку Упростить на две команды Заменить [edi+4\*eax] на [edi+2\*eax]  
add esi, 4;переходим к след элементу   
loop lp;вычесть ecx и если ecx=0 завершить цикл   
pop edi;восстановление регистров   
pop esi   
  
ret   
first\_dist endp   
end

**Second.asm**

.586   
.MODEL FLAT, C   
.CODE   
  
PUBLIC C second\_dist   
second\_dist PROC C result1:dword, intervals: dword, result2: dword, x\_max: dword, x\_min: dword, n\_int: dword Зачем сюда передаётся x\_max?  
  
push esi   
push edi   
  
mov esi, intervals   
mov edi, result2   
mov ecx, n\_int   
  
G:   
mov eax, [esi] ; левая граница интервала Поменять eax и ebx местами  
mov ebx, [esi + 4] ; правая граница   
  
cmp eax, x\_min ; если левая граница >= x\_min   
jge F2   
mov eax, 0 ; иначе, eax = 0, начало массива result1   
  
sub ebx, x\_min ; если длина интервала = 0   
jle F1 Заменить на jle F1  
jmp F5;иначе перейти к обработке не 0 интервала   
  
F2:   
sub ebx, eax ; количество элементов в интервале   
cmp ebx, 0 ; если длина интервала = 0   
je F1   
sub eax, x\_min ; индекс первого элемента из текущего интервала в массиве result1   
  
F5:   
push esi ;сохраняем регистры   
push ecx Удалить  
  
mov ecx, ebx ; количество элементов из result1 по которым нужно пройти   
mov esi, result1 ; массив   
mov ebx, 0 ; считает сумму подходящих элементов   
  
G2: ; цикл, считает сумму элементов, входящих в интервал   
add ebx, [esi + 4\*eax];к ebx добавляем количество попаданий в очередной еденичный интервал   
inc eax Удалить  
loop G2   
  
pop ecx;восстановление регистра   
  
  
cmp ecx, 1 ; если обрабатывали не последний элемент, то записываем сумму в массив результат   
jne F3   
add ebx, [esi + 4\*eax] ; иначе добавляем еще элемент последней правой скобки   
  
F3:   
  
mov [edi], ebx ; записываем результат   
pop esi;восстановление регистра   
jmp F1   
  
  
F1:   
add edi, 4 ; двигаемся к след. элементам массивов   
add esi, 4   
  
loop G   
  
  
pop edi  
pop esi   
  
ret   
second\_dist ENDP   
END