

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

**ОТЧЕТ**

**по лабораторной работе №6  
по дисциплине «ОргЭВМиС»**

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

Студент гр. 0382

Тюленев Т.В.

Преподаватели

Ефремов М.А.

Санкт-Петербург

2021

### Цель работы.

Изучить организацию связи Ассемблера с ЯВУ. Применить эти знания для написания программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

### 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.

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, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

Таблица 1.  $D_x = X_{\max} - X_{\min}$ ;  $Lg1, Lgi$  – первая или любая левая граница;  $ПГ\text{ посл}$  – правая граница последнего интервала

| Вид распределения | Число ассем. процедур | $N_{int} \geq D_x$ | $N_{int} < D_x$ | $Lgi \leq X_{\min}$ | $Lgi > X_{\min}$ | $ПГ\text{ посл} \leq X_{\max}$ | $ПГ\text{ посл} > X_{\max}$ |
|-------------------|-----------------------|--------------------|-----------------|---------------------|------------------|--------------------------------|-----------------------------|
| нормал.           | 1                     | +                  | -               | +                   | -                | +                              | -                           |

## **Выполнение работы.**

Для начала на ЯВУ считываются входные данные: количество генерируемых чисел, границы распределения, количество интервалов и сами интервалы.

Далее высчитываются математическое ожидание и среднеквадратическое отклонение для гауссовского распределения, после чего генерируются сами числа.

При небольшом количестве псевдослучайных чисел, не превышающем 100 значений мы выводим их на экран.

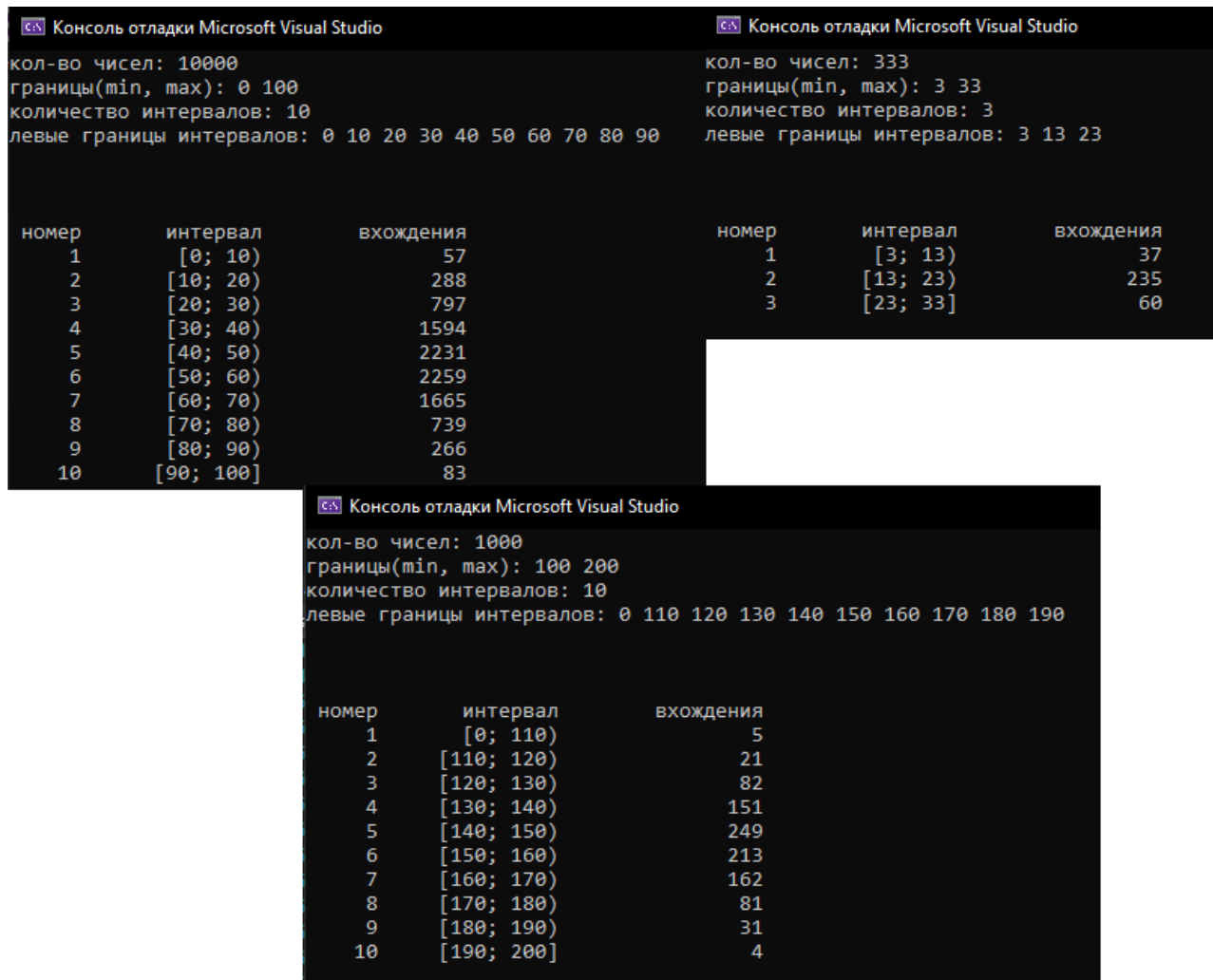
Затем вызывается функция из ассемблерного модуля, подсчитывающая количество вхождений в каждый интервал. Результат выводится в виде таблицы на экран и в файл.

Сам модуль содержит одну функцию, принимающую массив чисел и его размер, массив левых границ интервалов и его размер и массив для вывода. Для каждого элемента происходит поиск интервала, в который он входит, а затем количество вхождений для этого интервала увеличивается на единицу. По условию  $L_{g1} < X_{\min}$ , поэтому проверяется ситуация, когда число меньше минимума, и в этом случае не учитывается.

Разработанный программный код см. в приложении А.

## Тестирование.

Примеры работы программы:



The figure consists of three screenshots of the Microsoft Visual Studio console, each showing the output of a program. The console windows are titled "Консоль отладки Microsoft Visual Studio".

**Top Left Screenshot:**

```
кол-во чисел: 10000
границы(min, max): 0 100
количество интервалов: 10
левые границы интервалов: 0 10 20 30 40 50 60 70 80 90
```

| номер | интервал  | вхождения |
|-------|-----------|-----------|
| 1     | [0; 10)   | 57        |
| 2     | [10; 20)  | 288       |
| 3     | [20; 30)  | 797       |
| 4     | [30; 40)  | 1594      |
| 5     | [40; 50)  | 2231      |
| 6     | [50; 60)  | 2259      |
| 7     | [60; 70)  | 1665      |
| 8     | [70; 80)  | 739       |
| 9     | [80; 90)  | 266       |
| 10    | [90; 100] | 83        |

**Top Right Screenshot:**

```
кол-во чисел: 333
границы(min, max): 3 33
количество интервалов: 3
левые границы интервалов: 3 13 23
```

| номер | интервал | вхождения |
|-------|----------|-----------|
| 1     | [3; 13)  | 37        |
| 2     | [13; 23) | 235       |
| 3     | [23; 33] | 60        |

**Bottom Screenshot:**

```
кол-во чисел: 1000
границы(min, max): 100 200
количество интервалов: 10
левые границы интервалов: 0 110 120 130 140 150 160 170 180 190
```

| номер | интервал   | вхождения |
|-------|------------|-----------|
| 1     | [0; 110)   | 5         |
| 2     | [110; 120) | 21        |
| 3     | [120; 130) | 82        |
| 4     | [130; 140) | 151       |
| 5     | [140; 150) | 249       |
| 6     | [150; 160) | 213       |
| 7     | [160; 170) | 162       |
| 8     | [170; 180) | 81        |
| 9     | [180; 190) | 31        |
| 10    | [190; 200] | 4         |

Рис.1 — Примеры

### **Выводы.**

Была изучена организация связи Ассемблера с ЯВУ. Эти знания были применены для написания программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЕ КОДЫ ПРОГРАММ

Название файла: lr6.cpp

```
#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <random>
#include <stdlib.h>

using namespace std;

extern "C" void func(int* nums, int numsCount, int* leftBorders, int* result);

void output(string A, string B, string C, ofstream& file) {    // функция для вывода на
экран и в файл
    cout << setw(6) << right << A << setw(15) << right << B << setw(17) << right << C <<
endl;
    file << setw(6) << right << A << setw(15) << right << B << setw(17) << right << C <<
endl;}

int comp1(const void* a, const void* b){    // функция необходимая для qsort
    return (*(int*)a - *(int*)b);}

int main() {
    setlocale(LC_ALL, "ru");
    int i;
    int randNumCount;
    int max, min;
    int intervalCount;

    cout << "кол-во чисел: ";                // ввод необходимых параметров
    cin >> randNumCount;                      //
    cout << "границы(min, max): ";           //
    cin >> min >> max;                        //
    cout << "количество интервалов: ";       //
    cin >> intervalCount;                    //
    cout << "левые границы интервалов: ";    //
                                           //
    int* leftBorders = new int[intervalCount+1]; //
    int* result = new int[intervalCount+1];     //
    leftBorders[intervalCount] = max+1;         //
    result[intervalCount] = 0;                  //
                                           //
    for (i = 0; i < intervalCount; i++) {      //
        cin >> leftBorders[i];               //
        result[i] = 0;                       //
    }                                           //
    cout << endl;                             //

    qsort(leftBorders, intervalCount, sizeof(int), comp1); // сортировка левых границ по
возрастанию

    random_device rd{};                       //генерация случайных чисел
    mt19937 gen(rd());                         //(нормальное (гаусовское))
    float mean = float(max + min) / 2;         //
    float stddev = float(max - min) / 6;       //
    normal_distribution<float> dist(mean, stddev); //
    int* nums = new int[randNumCount];         //
    for (i = 0; i < randNumCount; i++) {       //
        nums[i] = round(dist(gen));            //
    }

    if (randNumCount < 101) {                  // вывод сгенерированных чисел (если их меньше 100)
```

```

        for (i = 0; i < randNumCount; i++) {
            cout << nums[i] << " ";}}
cout << endl << endl;
func(nums, randNumCount, leftBorders, result);    // вызов функции asm
ofstream file("output.txt");    // отправка на печать необходимых параметров
output("номер", "интервал", "вхождения", file); // номер интервала; границы
интервала; количество чисел
        for (i = 0; i < intervalCount; i++) {
            if (i == intervalCount - 1) {
                output(
                    to_string(i + 1),
                    '[' + to_string(leftBorders[i]) + "; " + to_string(leftBorders[i + 1]-1)
+ "]",
                    to_string(result[i + 1]),
                    file);}
            else {
                output(
                    to_string(i + 1),
                    '[' + to_string(leftBorders[i]) + "; " + to_string(leftBorders[i + 1]) +
")",
                    to_string(result[i + 1]),
                    file);}}
        file.close();
        return 0;
    }

```

Название файла: module.asm

```

.586
.MODEL FLAT, C
.CODE
    func PROC C nums:dword, numsCount:dword, leftBorders:dword, result:dword
        push eax
        push ebx
        push ecx
        push edx
        push esi
        push edi

        mov ecx, numsCount
        mov esi, nums
        mov edi, leftBorders

        mov edx, 0                ; индекс текущего номера
l:
        mov ebx, [esi+4*edx]      ; текущий номер
        cmp ebx, [edi]           ; крайняя левая граница
        jl continue              ; если x < крайняя левая граница

        mov eax, 0                ; индекс интервала
searchInterval:
        cmp ebx, [edi+4*eax]
        jl endSearch
        inc eax
        jmp searchInterval
endSearch:

        mov edi, result
        mov ebx, [edi+4*eax]      ; интервал в массиве результатов
        inc ebx
        mov [edi+4*eax], ebx
        mov edi, leftBorders

        continue:
    endfunc

```

```
                inc edx
                loop l

                pop edi
                pop esi
                pop edx
                pop ecx
                pop ebx
                pop eax
                ret
func ENDP
END
```



```

        searchInterval:
            cmp ebx, [edi+4*eax]
            jl endSearch
            inc eax
            jmp searchInterval
        endSearch:

        mov edi, result
        mov ebx, [edi+4*eax]; interval in result array
        inc ebx
        mov [edi+4*eax], ebx
        mov edi, leftBorders

        continue:
        inc edx
        loop l

    pop edi
    pop esi
    pop edx
    pop ecx
    pop ebx
    pop eax
    ret
func ENDP
END

```