

**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.**

Студентка гр. 0382

Чегодаева Е.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

### **Цель работы.**

Изучить организацию связи Ассемблера с ЯВУ на примере 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. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND\_GEN (при его отсутствии получить у преподавателя).

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.

Исходные данные:

1. Длина массива псевдослучайных целых чисел - NumRanDat ( $\leq 16K$ )
2. Диапазон изменения массива псевдослучайных целых чисел  $[X_{\min}, X_{\max}]$  (м.б. биполярный, например,  $[-100, 100]$ )
3. Массив псевдослучайных целых чисел  $\{X_i\}$ .
4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt ( $\leq 24$ )
5. Массив левых границ интервалов разбиения LGrInt .

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину, левые границы могут задаваться в произвольном порядке и иметь произвольные значения. Если  $X_{\min} < LGrInt(1)$ , то часть данных не будет участвовать в формировании распределения. Каждый интервал, кроме последнего, следует интерпретировать как  $[ LGrInt(i), LGrInt(i+1) )$ . Если у последнего интервала правая граница меньше  $X_{\max}$ , то часть данных не будет участвовать в формировании распределения.

Результаты:

Текстовая таблица, строка которой содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк должно быть равно числу интервалов разбиения.

Таблица должна выводиться на экран и сохраняться в файле.

Количеством ассемблерных модулей, формирующих требуемое распределение:

- Если указан 1 модуль, то он сразу формирует распределение по заданным интервалам и возвращает его в главную программу, написанную на ЯВУ;
- Если указаны 2 модуля, то первый из них формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат (это распределение должно выводиться на экран для контроля); затем вызывается второй модуль который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами).

Это распределение возвращается в главную программу и выдается как основной результат в виде текстового файла.

### Вариант 28:

Вид распределения: **Нормальное**;

Число ассем. процедур: **2**;

Число интервалов:  **$N_{int} \geq X_{max} - X_{min}$** ;

Значение левой границы 1 интервала:  **$Lg1 \leq X_{min}$** ;

### **Выполнение работы.**

lb6.cpp — Выполняет все действия, требуемые выполнения на ЯВУ.

Считывание всех необходимых данных сопровождается сообщениями-подсказками для пользователя, а также проверкой на выполнение всех заданных условий (количество чисел не должно превышать 16 000, количество возможных интервалов должно быть меньше-равно 24 и больше разности границ диапазона чисел, значение левой границы самого первого интервала не должно превышать наименьшее в массиве псевдослучайных чисел). Вместе с этим реализована возможность ввода левых границ интервалов в произвольном порядке и с произвольными значениями (с учётом условий) посредством сортировки полученного массива значений. Далее осуществляется генерация псевдослучайных чисел с нормальным распределением при помощи *mersenne twister*. Далее последовательно вызываются ассемблерные процедуры (Module1, Module2) с указанием всех требуемых, для обработки в каждой из процедур, значений. После завершения – в файле «result.txt» и на экране демонстрируются результаты работы в виде таблиц частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы (Таблица №1 для проверки корректности работы первой процедуры, Таблица №2 – второй).

Module1 — Выполняет распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ.

В процедуру из исходной программы поступают: Массив псевдослучайных чисел, количество этих чисел, наименьший элемент этого массива и результирующий массив.

Внутри процедуры посредством «цикла» по числу всех элементов в массиве реализован счёт количества чисел, входящих в конкретный единичный интервал.

**Module2** — Выполняет распределение исходных чисел по интервалам произвольной длины (с заданными границами), на основе результатов из ранее вызванного модуля.

В процедуру из исходной программы поступают: Массив распределения псевдослучайных чисел по единичным интервалам, левые границы заданных интервалов, число этих интервалов, наибольшее и наименьшее значения из случайных чисел и массив для хранения финального результата распределения.

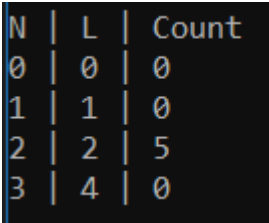
Изначально определяется значение левой границы первого интервала, все числа после этой границы и до следующей (= начало нового интервала) – обрабатываются: к результирующему массиву (который, на данном шаге, хранит количество встреч чисел до текущего (не первой итерации = 0) в пределах одного интервала) добавляется количество встреч текущего. Затем переходим к следующему псевдослучайному числу. Этот процесс повторяется до последнего элемента, обходя все заданные интервалы.

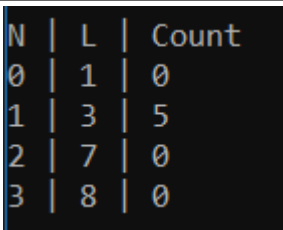
Исходный код программы см. в приложении А.

### Тестирование.

Результаты тестирования представлены в таблице 1.

Таблица 1 – результаты тестирования.

№	Входные данные	Выходные данные	Комментарии
1	>>5 >>1 5 >>4 >>0 1 2 4	 <pre> N   L   Count 0   0   0 1   1   0 2   2   5 3   4   0 </pre>	Верно
2	>>10 >>0 10 >>3	“Количество должно быть больше-ровно разности диапазона!!!”	Верно

3	>>10 >>4 6 >> 3 >>1 5 7	 <pre> N   L   Count 0   1   0 1   3   5 2   7   0 3   8   0 </pre>	Верно
---	----------------------------------	--	-------

### **Выводы.**

Была изучена организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы. Реализована программа, которая формирует таблицы частотного распределение псевдослучайных чисел с учётом условий задания входных данных, основываясь на двух ассемблерных процедурах, которые вызываются как независимо скомпилированные модули.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lb6.cpp

```
#include <iostream>
#include <random>
#include <stdio.h>
#include <fstream>
#include <string>

extern "C" void module1(int* arr, int n, int* res1, int min);
extern "C" void module2(int* distr, int* interv, int min, int max, int* res2);

using namespace std;

int comp(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    setlocale(LC_ALL, "Russian");
    int n;
    int min;
    int max;
    int NInt;
    cout << "\nВведите количество чисел: " << endl;
    cin >> n;
    if (n > 16000) {
        cout << "Слишком большое число" << endl;
        return 0;
    }
    cout << "Введите (через пробел) диапазон чисел: " << endl;
    cin >> min >> max;
    int D = max - min;
    cout << "Введите число интервалов: " << endl;
    cin >> NInt;
    if (NInt >= 24) {
        cout << "Слишком большое число интервалов" << endl;
        return 0;
    }
    if (NInt < abs(D)) {
        cout << "Количество должно быть больше-равно разности диапазона!!!" <<
endl;
        return 0;
    }
    int* interv = new int[NInt+1];
    int* result_modul2 = new int[n];
    cout << "Введите " << NInt << " левых границ интервалов: " << endl;

    for (int i = 0; i < NInt; i++) {
        cin >> interv[i];
        result_modul2[i] = 0;
    }
    qsort(interv, NInt, sizeof(int*), comp);

    if (interv[0] > min) {
        cout << "Левая граница первого интервала должна быть меньше первого
элемента диапазона!!!" << endl;
    }
}
```

```

        return 0;
    }

    random_device rd;
    mt19937 gen(rd());
    normal_distribution<> conc_gen((min + max) / 2, abs(max - min) / 4);

    interv[NInt] = max + 1;
    int* result_module1 = new int[abs(D) + 1];
    int* arr = new int[n];

    for (int i = 0; i < abs(D) + 1; i++) {
        result_module1[i] = 0;
    }

    cout << "\nМассив прсевдослучайных чисел:" << endl;
    for (int i = 0; i < n; i++) {
        arr[i] = int(conc_gen(gen));
        cout << arr[i] << ' ';
    }
    cout << endl;

    module1(arr, n, result_module1, min);

    ofstream file("result.txt");
    string bunner1 = "Таблица 1:\n";
    string head1 = "\nL | Count";

    file << bunner1 << head1 << endl;
    cout << "\n" << bunner1 << head1 << endl;
    for (int i = 0; i < abs(D) + 1; i++) {
        string res1 = (to_string(min + i) + " | " + to_string(result_module1[i]) + "\n");
        file << res1;
        cout << res1;
    }

    module2(result_module1, interv, min, max, result_modul2);
    string bunner2 = "\nТаблица 2:\n";
    string head2 = "\nN | L | Count";

    file << bunner2 << head2 << endl;
    cout << "\n" << bunner2 << head2 << endl;
    for (int i = 0; i < NInt; i++) {
        string res2 = (to_string(i) + " | " + to_string(interv[i]) + " | " +
to_string(result_modul2[i]) + "\n");
        file << res2;
        cout << res2;
    }

    return 0;
}

```



Название файла: modul1.asm

```
.586
.MODEL FLAT, C
.CODE
PUBLIC C module1
module1 PROC C arr: dword, n: dword, res: dword, min: dword

push esi
push edi
mov esi, res
mov edi, arr
mov ecx, n

lp:
    mov eax,[edi]
    sub eax,min
    mov ebx,[esi+4*eax]
    add ebx, 1
    mov [esi+4*eax],ebx
    add edi,4
    loop lp

pop edi
pop esi
ret
module1 ENDP
END
```

Название файла: modul2.asm

```
.586
.MODEL FLAT, C
.CODE
PUBLIC C module2
module2 PROC C distr: dword, interv: dword, min: dword, max: dword, res: dword

push esi
push edi
push eax
push ebx
push ecx

mov esi, res
mov edi, interv
mov eax, min
mov ebx, 0
mov ecx, 0

Start:
cmp eax, [edi+4*ebx]
jl Act
add ebx, 1
jmp Start

Act:
    push ecx
    mov ecx, max
    cmp eax, ecx
    pop ecx
    je final
    push edi
```

```
push eax
mov edi, distr
sub ebx, 1; ")"
mov eax, [esi+4*ebx]
mov edx, [edi+4*ecx]
add eax, edx
mov [esi+4*ebx], eax
pop eax
pop edi
add ecx, 1
add eax, 1
jmp Start
```

final:

```
pop eax
pop ebx
pop ecx
pop edi
pop esi
ret
```

```
module2 ENDP
END
```