

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. 0383

Преподаватель

Бояркин Н.А.

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Вариант 22.

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 (≤ 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.

Выполнение работы.

Для начала на ЯВУ считываются входные данные: кол-во генерируемых чисел, границы распределения, кол-во интервалов и сами интервалы. По условию задания кол-во интервалов \geq диапазона чисел, но реализация при обратном условии не меняется. Далее высчитываются математическое ожидание и среднеквадратическое отклонения для гауссовского распределения, после чего генерируются сами числа. Затем вызывается функция из ассемблерного модуля, подсчитывающий кол-во 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 кол-во вхождений для этого интервала увеличивается на единицу. По условию $L_{g1} > X_{min}$, поэтому проверяется ситуация, когда число меньше крайней левой границы, и в этом случае не учитывается.

Тексты исходного файла программы см. в приложении А.

Рис. 1 - Проверка работы программы.

```
Введите кол-во псевдослучайных целых чисел: 1000
Введите границы: -10 10
Введите количество интервалов: 4
Введите левые границы: -8 -4 0 4

Результат:


| Номер | Интервал | Кол-во значений |
|-------|----------|-----------------|
| 1     | [-8; -4) | 101             |
| 2     | [-4; 0)  | 377             |
| 3     | [0; 4)   | 384             |


Для продолжения нажмите любую клавишу . . .
```

Рис. 2 - Проверка работы программы.

```
Введите кол-во псевдослучайных целых чисел: 1000
Введите границы: -10 0
Введите количество интервалов: 11
Введите левые границы: -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0

Результат:


| Номер | Интервал  | Кол-во значений |
|-------|-----------|-----------------|
| 1     | [-10; -9) | 5               |
| 2     | [-9; -8)  | 15              |
| 3     | [-8; -7)  | 56              |
| 4     | [-7; -6)  | 134             |
| 5     | [-6; -5)  | 174             |
| 6     | [-5; -4)  | 226             |
| 7     | [-4; -3)  | 195             |
| 8     | [-3; -2)  | 121             |
| 9     | [-2; -1)  | 45              |
| 10    | [-1; 0)   | 21              |


Для продолжения нажмите любую клавишу . . .
```

Выводы.

В ходе выполнения данной лабораторной работы был изучен принцип организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ТЕКСТЫ ИСХОДНЫХ ФАЙЛОВ ПРОГРАММЫ

Название файла: LAB6.cpp

```
#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <random>

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) {
    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 main() {
    setlocale(LC_ALL, "ru");

    int randNumCount;

    cout << "Введите кол-во псевдослучайных целых чисел: ";
    cin >> randNumCount;

    if (randNumCount <= 0) { cout << "Некорректное кол-во чисел"; return -1; };

    int max, min;

    cout << "Введите границы: ";
    cin >> min >> max;
```

```

if (max <= min) { cout << "Некорректные границы распределения"; return -1; };

int intervalCount;
cout << "Введите количество интервалов: ";
cin >> intervalCount;
if (randNumCount <= 0) { cout << "Некорректное кол-во интервалов"; return -1; };

cout << "Введите левые границы: ";
int* leftBorders = new int[intervalCount];
int* result = new int[intervalCount];
for (int i = 0; i < intervalCount; i++) {
    cin >> leftBorders[i];

    int index = i;
    while (index && leftBorders[index] < leftBorders[index - 1]) {
        swap(leftBorders[index--], leftBorders[index]);
    }
    result[i] = 0;
}
cout << endl;

random_device rd{};
mt19937 gen(rd());

float mean = float(max + min) / 2;
float stddev = float(max - min) / 6;
normal_distribution<float> dist(mean, stddev);

int* nums = new int[randNumCount];

```

```
for (int i = 0; i < randNumCount; i++) {  
    nums[i] = round(dist(gen));  
}
```

```
func(nums, randNumCount, leftBorders, result);
```

```
ofstream file("output.txt");  
cout << "Результат:\n";  
output("Номер", "Интервал", "Кол-во значений", file);  
for (int i = 0; i < intervalCount - 1; i++) {  
    output(  
        to_string(i + 1),  
        '[' + to_string(leftBorders[i]) + "; " + to_string(leftBorders[i + 1]) + ")",  
        to_string(result[i + 1]),  
        file  
    );  
}
```

```
file.close();  
system("pause");  
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; index of current number

l:

mov ebx, [esi+4*edx]; current number

cmp ebx, [edi]; most left border

jl continue; if x < most left border

mov eax, 0; index of interval

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
```