

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

**ОТЧЕТ**  
**по лабораторной работе №6**  
**по дисциплине «Архитектура ЭВМ и систем»**  
**Тема: Организация связи Ассемблера с ЯВУ на примере программы**  
**построения частотного распределения попаданий псевдослучайных**  
**целых чисел в заданные интервалы.**

Студент гр. 9383

Моисейченко К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Научиться реализовывать связь Ассемблера и ЯВУ. Написать программу построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

### **Задание.**

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND\_GEN (при его отсутствии программу датчика получить у преподавателя).

Далее должен вызываться ассемблерный модуль(модули) для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину.

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 ( $\leq 16K$ ,  $K=1024$ )
2. Диапазон изменения массива псевдослучайных целых чисел  $[X_{min}, X_{max}]$ , значения могут быть биполярные;
3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt ( $\leq 24$ )
4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу  $[X_{min}, X_{max}]$ ).

Результаты:

1. Текстовый файл, строка которого содержит:

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

Количество строк равно числу интервалов разбиения.

2. График, отражающий распределение чисел по интервалам.

(необязательный результат)

Формирование частотного распределения должно производиться подпрограммой формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в главную программу, написанную на ЯВУ.

### **Ход работы.**

В качестве ЯВУ используется язык C++.

Пользователь вводит длину массива, верхнюю и нижнюю границы значений, количество интервалов и нижние границы интервалов. Программа генерирует массив из псевдослучайных целых чисел, который передается в ассемблерный модуль для формирования распределения количества попаданий этих чисел в заданные интервалы. Результат выводится на экран и записывается в текстовый файл result.txt.

### **Тестирование.**

```

Enter the size of the array (<=2^14):
20
Enter the lower limit:
0
Enter the upper limit:
300
Enter the number of ranges (<= 24):
3
Enter 2 lower range limits:
100
200

Generated numbers:
55 61 69 70 73 79 88 140 155 189 191 202 213 223 227 254 276 282 299 299

Number | Range | Quantity
-----
1 | [0, 100) | 7
2 | [100, 200) | 4
3 | [200, 300] | 9

```

Рисунок 1 - Пример работы программы №1

```

Enter the size of the array (<=2^14):
10
Enter the lower limit:
-1
Enter the upper limit:
1
Enter the number of ranges (<= 24):
3
Enter 2 lower range limits:
0
1

Generated numbers:
-1 -1 0 0 0 0 1 1 1 1

Number | Range | Quantity
-----
1 | [-1, 0) | 2
2 | [0, 1) | 4
3 | [1, 1] | 4

```

Рисунок 2 - Пример работы программы №2

### Выводы.

Была изучена организация связи Ассемблера с ЯВУ и написана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

#### Файл main.cpp:

```
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <algorithm>
#include <ctime>
using namespace std;

extern "C" void distribution(int* arr, int* LGrInt, int NumRanDat,
int* result);

int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(0));
    int NumRanDat = 0, Xmin = 0, Xmax = 0, NInt = 0;
    cout << "Enter the size of the array (<=2^14):\n";
    cin >> NumRanDat;
    if (NumRanDat > 16 * 1024 || NumRanDat <= 0) {
        cout << "error: wrong array size entered\n";
        exit(1);
    }
    cout << "Enter the lower limit:\n";
    cin >> Xmin;
    cout << "Enter the upper limit:\n";
    cin >> Xmax;
    if (Xmin > Xmax) {
        cout << "error: wrong limits entered\n";
        exit(1);
    }
    cout << "Enter the number of ranges (<= 24): \n";
    cin >> NInt;
    if (NInt > 24 || NInt < 1 || NInt > (Xmax - Xmin + 1)) {
        cout << "error: wrong number of ranges entered\n";
        exit(1);
    }
}
```

```

    }
    int* LGrInt = new int[NInt+1]();
    LGrInt[0] = Xmin;
    LGrInt[NInt] = Xmax + 1;
    if (NInt == 2)
        cout << "Enter the lower limit of the second range:\n";
    else
        cout << "Enter " << NInt - 1 << " lower range limits:\n";
    for (int i = 1; i < NInt; i++) {
        cin >> LGrInt[i];
        while (LGrInt[i] < LGrInt[i-1]) {
            cout << "The limit should be greater than the previous one (>" <<
LGrInt[i-1] << ")\n";
            cin >> LGrInt[i];
        }
        if (LGrInt[i] < Xmin || LGrInt[i] > Xmax) {
            cout << "error: wrong lower limit entered\n";
            exit(1);
        }
    }

    int* arr = new int[NumRanDat]();
    for (int i = 0; i < NumRanDat; i++) {
        arr[i] = rand() % (Xmax - Xmin + 1) + Xmin;
    }
    sort(arr, arr + NumRanDat);
    int* result = new int[NInt];
    for (int i = 0; i < NInt; i++)
        result[i] = 0;
    distribution(arr, LGrInt, NumRanDat, result);
    LGrInt[NInt] -= 1;
    ofstream file("result.txt");
    cout << "\nGenerated numbers:\n";
    file << "Generated numbers:\n";
    for (int i = 0; i < NumRanDat; i++) {
        cout << arr[i] << " ";
        file << arr[i] << " ";
    }
    cout << "\n\n";
    file << "\n\n";

```

```

cout << "Number | Range | Quantity\n";
file << "Number | Range | Quantity\n";
cout << "-----" << endl;
file << "-----" << endl;
for (int i = 0; i < NInt; i++) {
    if (i == NInt - 1) {
        file << " " << i + 1 << " | [" << LGrInt[i] << ", " << LGrInt[i
+ 1] << "]" | " << result[i] << "\n";
        cout << " " << i + 1 << " | [" << LGrInt[i] << ", " << LGrInt[i
+ 1] << "]" | " << result[i] << "\n";
    }
    else {
        file << " " << i + 1 << " | [" << LGrInt[i] << ", " << LGrInt[i
+ 1] << ") | " << result[i] << "\n";
        cout << " " << i + 1 << " | [" << LGrInt[i] << ", " << LGrInt[i
+ 1] << ") | " << result[i] << "\n";
    }
}
}
}

```

### **Файл distribution.asm:**

```

.model flat,c

.code

distribution proc C arr: dword, LGrInt: dword, NumRanDat: dword,
result: dword

    mov ebx, [arr]
    mov esi, [LGrInt]
    mov edi, [result]
    mov ecx, [NumRanDat]

my_loop:
    mov eax, [ebx] ; eax = current_arr
    add ebx, 4 ; ebx = next_arr

comparison:
    cmp eax, [esi+4]

```

```

jge next_LGrInt ; if current_arr >= next_LGrInt

; if current_arr < next_LGrInt: current_result++
mov eax, [edi]
add eax, 1
mov [edi], eax
jmp loop_end

next_LGrInt:
add esi, 4 ; esi = next_LGrInt
add edi, 4 ; edi = next_result
jmp comparison ; repeat comparison with the next LGrInt

loop_end:
dec ecx
cmp ecx, 0
jne my_loop

ret

distribution endp
end

```