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

# 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. 9383 _____ Гордон Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

**Цель работы.**

Разработать программу для построения попадания псевдослучайных чисел в интервалы, состоящую из двух файлов: одного, написанного на языке Ассемблера, другого – на языке ЯВУ.

**Текст задания.**

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 целыхчисел - NumRanDat (<= 16K, K=1024)

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] , значения могут быть биполярные; 14

3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt ( <=24 )

4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]).

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)

В зависимости от номера бригады формирование частотного распределения должно производиться по одному из двух вариантов:

1. Для бригад с нечетным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в головную программу, написанную на ЯВУ;

2. Для бригад с четным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами). Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла и, возможно, графика.

**Вариант №1 (бригада 1)**

1. Для бригад с нечетным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в головную программу, написанную на ЯВУ;

# ПРОТОКОЛ

```
Введите размер массива (меньше 16384):
5
Xmin:
0
Xmax:
5
Введите не более 4 интервалов:
2
Введите нижние пределы для 1 интервалов
3
Случайные числа:
1 2 4 0 4
номер|интервал  |количество
1    |  [0, 3]  |   3
2    |  [3, 5]  |   2
```

```
Введите размер массива (меньше 16384):
10
Xmin:
0
Xmax:
10
Введите не более 9 интервалов:
2
Введите нижние пределы для 1 интервалов
6
Случайные числа:
1 7 4 0 9 4 8 8 2 4
номер|интервал  |количество
1    |  [0, 6]  |   6
2    |  [6, 10]  |   4
```

```
Введите размер массива (меньше 16384):
-1
Введенное значение больше 16384 или меньше 0! Введите еще раз!
3
Xmin:
0
Xmax:
10
Введите не более 2 интервалов:
2
Введите нижние пределы для 1 интервалов
5
Случайные числа:
1 7 4
номер|интервал  |количество
1    |  [0, 5]  |   2
2    |  [5, 10]  |   1
```

# ВЫВОДЫ

Была разработана программа, включающая в себя два файла .asm и .cpp, по попаданию псевднослучайных чисел в заданные пользователем интервал(ы),. В .cpp файле – определение функции (extern), которая реализована в .asm . Таким образом, когда нам это нужно мы делегируем работу этой функции.

# ПРИЛОЖЕНИЕ

LR6.cpp:

```
#include <iostream>
#include <fstream>

using namespace std;

extern "C" void func(int NumRanDat, int* arr, int* LGrInt, int* ans);

void printArr(int* arr, int size)
{
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
}

void writeNumRanDat(int& NumRanDat)
{
    bool exit = false;
    cout << "Введите размер массива (меньше 16384):\n";
    while (exit != true)
    {
        cin >> NumRanDat;
        if (NumRanDat < 16 * 1024 and NumRanDat > 0)
        {
            exit = true;
        }
        else
        {
            cout << "Введенное значение больше 16384 или меньше 0! Введите еще раз!\n";
        }
    }
}

void writeRangesAmount(int& NInt, const int arr_size)
{
    bool exit = false;
    int amount = arr_size - 1 > 24 ? 24 : arr_size - 1;
    cout << "Введите не более " << amount << " интервалов:\n";
    while (exit != true)
    {
        cin >> NInt;
        if (NInt > amount)
        {
            cout << "Введенное количество интервалов неверно! Введите еще раз\n";
        }
        else
        {
            exit = true;
        }
    }
}

void writeRangesBorders(const int NInt, const int Xmin, const int Xmax, int* LGrInt)
{
    int i = 0;
    cout << "Введите нижние пределы для " << NInt - 1 << " интервалов\n";
    while (i < NInt - 1)
    {
        cin >> LGrInt[i];
        if (LGrInt[i] > Xmax or LGrInt[i] < Xmin)
        {
```

```
            cout << "Введенный нижний предел не попадает в интервал" << "[" << Xmin << ","
<< Xmax << "]" <<"\n";
            cout << "Введите значение еще раз:\n";
        }
        else if (i > 0 and LGrInt[i] < LGrInt[i - 1])
        {
            cout << "Нижняя граница " << LGrInt[i - 1] << " больше, чем предыдущая - " <<
LGrInt[i] << '\n';
            cout << "Введите значение заново:\n";
        }
        else
        {
            i++;
        }
    }
    LGrInt[NInt - 1] = Xmax;
}

int main()
{
    setlocale(LC_ALL, "Russian");
    int NumRanDat = 0, Xmin = 0, Xmax = 0, NInt = 0;
    writeNumRanDat(NumRanDat);
    cout << "Xmin:\n";
    cin >> Xmin;
    cout << "Xmax:\n";
    cin >> Xmax;
    writeRangesAmount(NInt, NumRanDat);
    int* LGrInt = new int[NInt];
    writeRangesBorders(NInt, Xmin, Xmax, LGrInt);
    int* arr = new int[NumRanDat];
    for (int i = 0; i < NumRanDat; i++)
    {
        arr[i] = Xmin + rand() % (Xmax - Xmin);
    }
    int* ans = new int[NInt];
    for (int i = 0; i < NInt; i++)
    {
        ans[i] = 0;
    }
    func(NumRanDat, arr, LGrInt, ans);
    ofstream file("output.txt");
    cout << "Случайные числа:\n" ;
    file << "Случайные числа:\n";
    for (int i = 0; i < NumRanDat; i++)
    {
        cout << arr[i] << " ";
        file << arr[i] << " ";
    }
    cout << '\n';
    file << '\n';
    cout << "номер|интервал  |количество\n";
    file << "номер|интервал  |количество\n";
    for (int i = 0; i < NInt; i++)
    {
        int n1, n2;
        n1 = i != 0 ? LGrInt[i - 1] : Xmin;
        n2 = i != NInt ? LGrInt[i] : Xmax;
        file << i + 1 << "    |  " << "[" << n1 << ", " << n2 << "]" << "  |  " << ans[i] <<
"\n";
        cout << i + 1 << "    |  " << "[" << n1 << ", " << n2 << "]" << "  |   " << ans[i]
<< "\n";
    }
}
```

mod.asm:

```
.MODEL FLAT, C
.CODE
func PROC C NumRanDat:dword, arr:dword, LGrInt:dword, ans:dword

	mov ecx,0			;счетчик для прохода по массиву
	mov ebx,arr
	mov esi,LGrInt
	mov edi,ans

main:
	mov edx,[ebx]			;берем элемент входного массива
	push ebx			; сохраняем указатель на текущий элемент
	sub ebx,ebx		; обнуляем указатель

compare:
	mov eax,ebx		; eax содержит текущий индекс массива границ
	shl eax,2			; j >> 2
	cmp edx,[esi+eax]	; сравниваем arr[i] (edx отвечает за перемещение) и LGrInt[j]
(esi -> LGrInt, eax = j)
	jle append				; arr[i] <= LGrInt[j]
	inc ebx				; arr[i] > LGrInt[j] => i++
	jmp compare			; проверяем arr[i]

append:
	add eax,edi		;edi -> ans => eax -> ans
	mov edx,[eax]
	inc edx
	mov [eax],edx
	pop ebx				;забираем текущий элемент и ссылаемся на новый
	add ebx,4
	inc ecx				;i++
	cmp ecx, NumRanDat
	jl main				;i < NumRanDat

ret
func ENDP

END
```