**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.**

Студентка гр. 9383 _____ Карпекина А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

**Цель работы.**

Изучение организации связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых 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 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;

**Выполнение работы.**

Первоначально у пользователя запрашиваются все необходимые данные: длина массива, нижняя и верхняя границы значений, количество интервалов, нижние границы интервалов. Затем генерируется массив псевдослучайных чисел и передается в модуль asm для дальнейшей обработки. В модуле заполняется массив с количеством чисел в каждом диапазоне. После этого данные выводятся пользователю, программа завершается.

Исходный код программы см. в приложении А.

**Тестирование.**

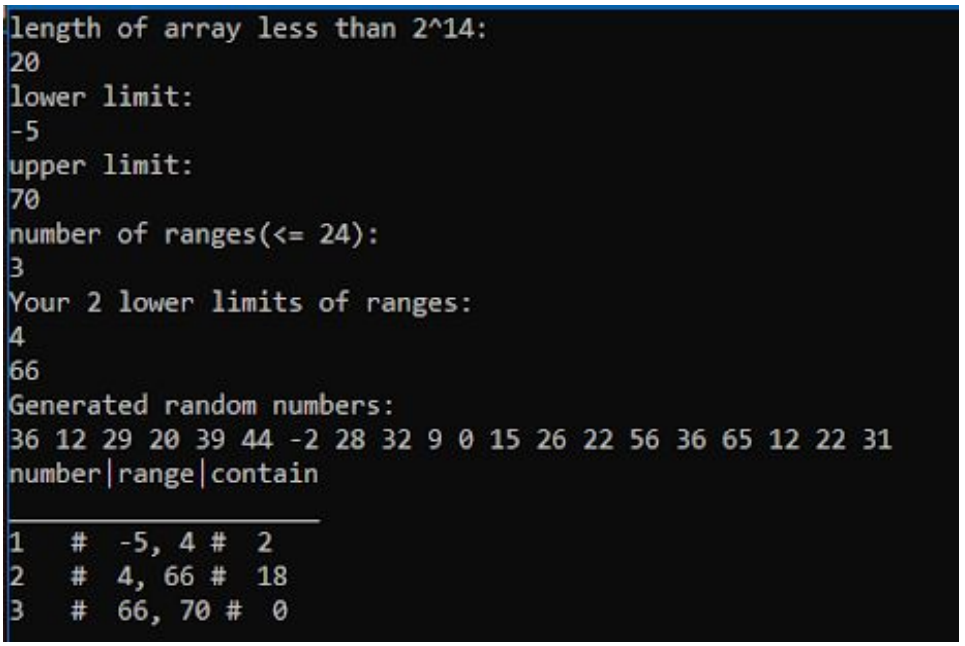


Рисунок 1 - Пример вывода программы №1

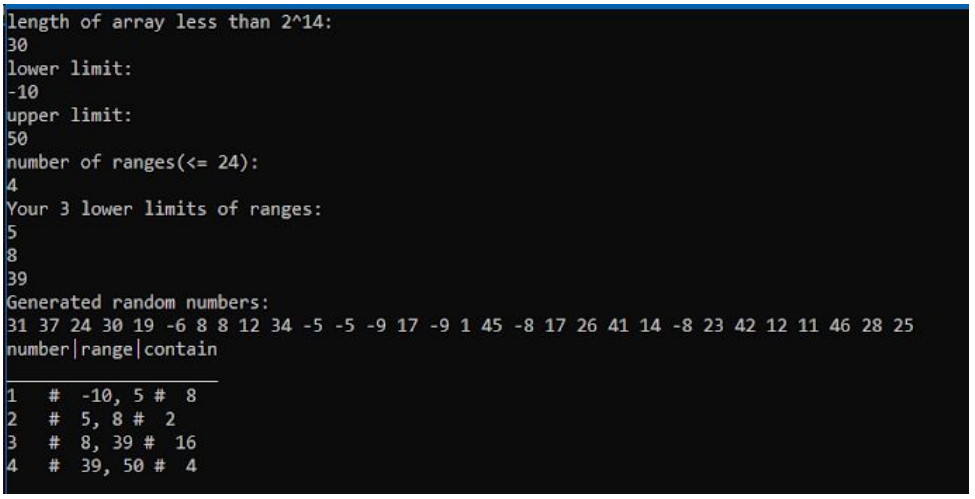


Рисунок 2 - Пример вывода программы №2

**Вывод.**

В ходе работы была изучена организация связи Ассемблера с ЯВУ и написана программа построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

## ПРИЛОЖЕНИЕ А

## РАЗРАБОТАННЫЙ КОД ПРОГРАММЫ

**main.cpp**

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <random>
using namespace std;

extern "C"
{
   void _sort(int NumRanDat, int* arr, int* LGrInt, int* res_arr);
}

int main()
{
   int NumRanDat = 0, Xmin = 0, Xmax = 0, NInt = 0;
   cout << "length of array less than 2^14: " << endl;
   cin >> NumRanDat;
   if (NumRanDat > 16 * 1024 || NumRanDat < 0) {
      cout << "Oops";
      exit(1);
   }
   cout << "lower limit: " << endl;
   cin >> Xmin;
   cout << "upper limit: " << endl;
   cin >> Xmax;
   std::cout << "number of ranges(<= 24): " << endl;
   std::cin >> NInt;
   if (NInt > 24) {
      cout << "Oops";
      exit(1);
   }
   int* LGrInt = new int[NInt]();
   std::cout << "Your " << NInt - 1 << " lower limits of ranges:" << endl;
   for (int i = 0; i < NInt - 1; i++) {
      cin >> LGrInt[i];
      if (LGrInt[i] < LGrInt[i - 1]) {
            cout << "Entered limit " << LGrInt[i] << " is bigger than previous,
enter again" << endl;
         cin >> LGrInt[i];
```

```
        }
        if (LGrInt[i] < Xmin || LGrInt[i] > Xmax) {
            cout << "Oops";
            exit(1);
        }
    }
    LGrInt[NInt - 1] = Xmax;
    int* mas = new int[NumRanDat]();
    for (int i = 0; i < NumRanDat; i++) {
        mas[i] = Xmin + rand() % (Xmax - Xmin);
    }
    int* rmas = new int[NInt];
    for (int i = 0; i < NInt; i++)
        rmas[i] = 0;
    _sort(NumRanDat, mas, LGrInt, rmas);
    ofstream file("res.txt");
    cout << "Generated random numbers:" << endl;
    file << "Generated random numbers:" << endl;
    for (int i = 0; i < NumRanDat; i++) {
        cout << mas[i] << " ";
        file << mas[i] << " ";
    }
    cout << endl;
    file << endl;
    cout << "number|range|contain" << endl;
    file << "number|range|contain" << endl;
    cout << "_____________________"<< endl;
    file << "_____________________"<< endl;
    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 << " #  " << rmas[i] << "\n";
        cout << i + 1 << "   #  " << n1 << ", " << n2 << " #  " << rmas[i] << "\n";
    }
}
```

**count.asm**

```
.686
.MODEL FLAT, C
.STACK
.DATA
.CODE
_sort PROC C NumRanDat:dword, mas:dword, LGrInt:dword, rmas:dword
```

```
        mov ecx,0 ;счетчик для прохода по массиву
        mov ebx,[mas]
        mov esi,[LGrInt]
        mov edi,[rmas]

f1:
        mov edx,[ebx] ;берем элемент входного массива
        push ebx ; сохраняем указатель на текущий элемент
        sub ebx,ebx ; обнуляем указатель

f2:
        mov eax,ebx ; eax содержит текущий индекс массива границ
        shl eax,2 ; индекс умножаем на 4, так как каждый элемент по 4 байт
        cmp edx,[esi+eax] ; сравниваем текующий элемент с текущей левой
границей
        jle fe
        inc ebx
        jmp f2

fe:
        add eax,edi ;после сложения указываем на элемент в
результирующем массиве для инкрементирования
        mov edx,[eax]
        inc edx
        mov [eax],edx;
        pop ebx ;забираем текущий элемент и ссылаемся на новый
        add ebx,4
        inc ecx ;инкрементируем индекс массива
        cmp ecx, NumRanDat
        jl f1

ret
_sort ENDP

END
```