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

ОТЧЕТ

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

Студент гр. 1303 _____ Мусатов Д.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить детали организации связи Ассемблера с ЯВУ, написать ассемблерный модуль для использования в программе.

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. Следует привести числа к целому виду с учетом диапазона изменения.

Далее должна вызываться ассемблерная процедура для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерная процедура должна вызываться как независимо скомпилированный модуль. 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 (<= 16K)
2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax]
3. Массив псевдослучайных целых чисел {Xi}
4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)
5. Массив левых границ интервалов разбиения LGrInt

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.

На языке C++ было реализовано считывание исходных данных (сколько псевдослучайных чисел он должен сгенерировать, потом минимальное и максимальное значение этих чисел, потом количество интервалов, по которым эти числа нужно раскидать, и наконец, левые границы этих интервалов), числа хранятся в массиве *nums*, левые границы и правая граница последнего интервала хранятся в *intervals*. Здесь же генерируется необходимое количество псевдослучайных чисел в соответствии с нормальным распределением с дисперсией, равной 1.

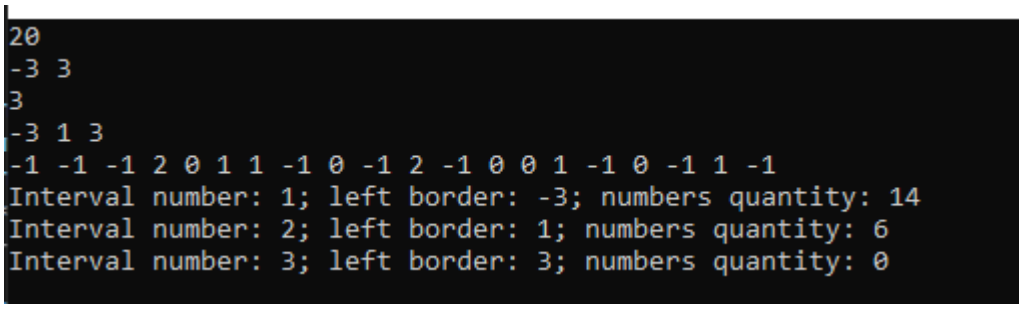

Рисунок 1 – Тестирование программы в консоли

Исходный код программы см. в приложении А.

Вывод.

В ходе выполнения лабораторной работы были изучены детали организации связи Ассемблера с ЯВУ, написан ассемблерный модуль для использования в программе.

ПРИЛОЖЕНИЕ А

Тексты исходных файлов программ.

lab6.cpp

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

using namespace std;

extern "C" {
    void func(int numRanDat, int nInt, int* nums, int* intervals,
int* res);
}

int main() {
    int numRanDat, xMin, xMax, nInt;
    cin >> numRanDat;
    if (numRanDat <= 0 || numRanDat > 16000) {
        cout << "Entered array length is wrong" << endl;
        exit(-1);
    }
    cin >> xMin >> xMax;
    if (xMin >= xMax) {
        cout << "Entered limits are wrong" << endl;
        exit(-1);
    }
    cin >> nInt;
    if (nInt <= 0 || nInt > 24) {
        cout << "Entered number of intervals is wrong" << endl;
        exit(-1);
    }
    int* nums = new int[numRanDat];
    int* ints = new int[nInt + 1];
    for (int i = 0; i < nInt; i++) {
        cin >> ints[i];
```

```
    bool wrong = false;
    if (i > 0) {
        if (ints[i] < ints[i - 1]) {
            wrong = true;
        }
    }
    if (wrong) {
        cout << "Entered border is wrong" << endl;
        delete[] nums;
        delete[] ints;
        exit(-1);
    }
}
ints[nInt] = xMax + 1;
random_device rd;
mt19937 gen(rd());
normal_distribution<> d(0, 1);
int iter = 0;
while (iter < numRanDat) {
    double value = round(d(gen));
    if (value >= xMin && value <= xMax) {
        nums[iter] = int(value);
        iter++;
    }
}
fstream outFile;
outFile.open("result.txt", ios::out | ios::trunc);
for (int i = 0; i < numRanDat; i++) {
    cout << nums[i] << ' ';
    outFile << nums[i] << ' ';
}
cout << endl;
outFile << endl;
int* res = new int[nInt] {0};
func(numRanDat, nInt, nums, ints, res);
for (int i = 0; i < nInt; i++) {
```

```
            cout << "Interval number: " << i + 1 << "; left border: "
      << ints[i] << "; numbers quantity: " << res[i] << endl;
            outFile << "Interval number: " << i + 1 << "; left border:
      " << ints[i] << "; numbers quantity: " << res[i] << endl;
        }
        outFile.close();
        delete[] nums;
        delete[] ints;
        delete[] res;
        return 0;
      }
```

lab6.asm

```
.MODEL FLAT, C
.CODE

func PROC C numRanDat: dword, nInt: dword, nums: dword, ints: dword,
res: dword
      push esi
      push edi
      push eax
      push ebx
      push ecx
      push edx
      mov ecx, numRanDat
      mov esi, nums
      mov edi, ints
      mov eax, 0
      mov edx, nInt
      inc edx
      lp:
            mov ebx, 0
            checking:
                  cmp ebx, edx
                  jge skip
```

```
                push eax
                mov eax, [esi + eax * 4]
                cmp eax, [edi + ebx * 4]
                pop eax
                jl out_num
                inc ebx
                jmp checking
          out_num:
                dec ebx
                cmp ebx, -1
                je skip
                mov esi, res
                push eax
                mov eax, [esi + ebx * 4]
                inc eax
                mov [esi + ebx * 4], eax
                pop eax
                mov esi, nums
          skip:
                inc eax
     loop lp
     finish:
     pop edx
     pop ecx
     pop ebx
     pop eax
     pop edi
     pop esi
     ret
func ENDP
     END
```