**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 |  | Кропотов Н.Д. |
| 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 должны вызываться ассемблерные процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должна вызываться как независимо скомпилированный модуль. Передача параметров в процедуры 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,

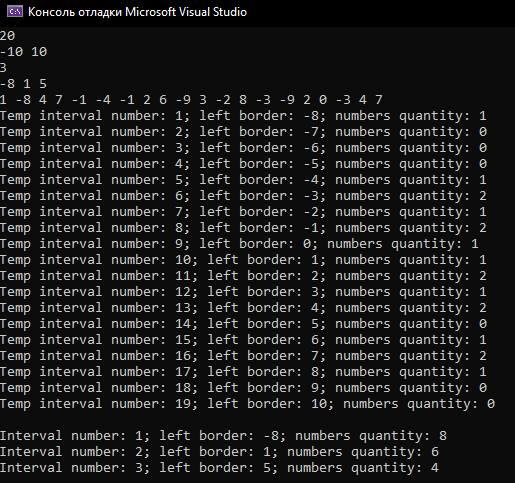
- количество псевдослучайных чисел, попавших в интервал.

Количество строк должно быть равно числу интервалов разбиения.

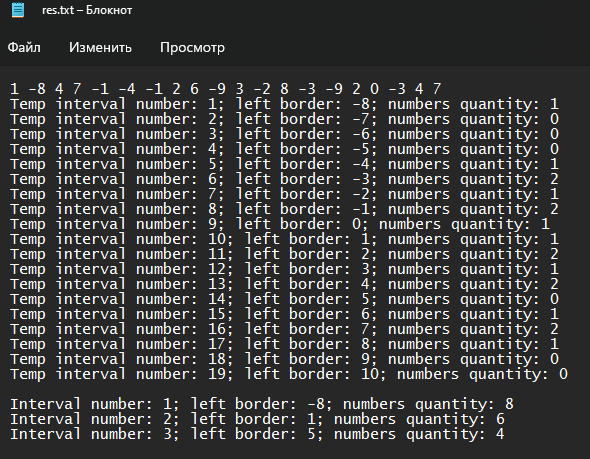
Таблица должна выводиться на экран и сохраняться в файле.

## Выполнение работы.

На языке C++ было реализовано считывание исходных данных, числа хранятся в массиве *numbers*, левые границы и правая граница последнего интервала хранятся в *intervals*. Здесь же генерируется необходимое количество псевдослучайных чисел в соответствии с равномерным распределением. Первая ассемблерная процедура генерирует распределение чисел по единичным интервалам, промежуточный результат фиксируется в консоли и в файле, после чего передается второй процедуре для распределения по заданным интервалам.



*Рисунок 1 – Тестирование программы в консоли*



*Рисунок 2 – Результат работы программы в файле*

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.

## ConsoleApplication.cpp

#include <iostream>

#include <random>

#include <fstream>

using namespace std;

extern "C" {

void firstFunc(int numCount, int tempResSize, int\* numbers, int\* tempIntervals, int\* tempRes);

void secondFunc(int tempResSize, int intCount, int\* tempRes, int\* tempIntervals, int\* intervals, int\* res);

}

int main() {

int numCount, xMin, xMax, intCount;

cin >> numCount;

cin >> xMin >> xMax;

if (xMin > xMax) {

cout << "xMin is larger than xMax" << endl;

exit(-1);

}

cin >> intCount;

int\* numbers = new int[numCount];

int\* intervals = new int[intCount + 1];

for (int i = 0; i < intCount; i++) {

cin >> intervals[i];

}

intervals[intCount] = xMax + 1;

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> d(xMin, xMax);

for (int i = 0; i < numCount; i++) {

numbers[i] = d(gen);

}

fstream resFile;

resFile.open("res.txt", ios::out | ios::trunc);

for (int i = 0; i < numCount; i++) {

cout << numbers[i] << ' ';

resFile << numbers[i] << ' ';

}

cout << endl;

resFile << endl;

int tempResSize = xMax + 1 - intervals[0];

int\* tempIntervals = new int[tempResSize + 1];

int\* tempRes = new int[tempResSize] {0};

for (int i = 0; i < tempResSize; i++) {

tempIntervals[i] = intervals[0] + i;

}

tempIntervals[tempResSize] = xMax + 1;

firstFunc(numCount, tempResSize, numbers, tempIntervals, tempRes);

for (int i = 0; i < tempResSize; i++) {

cout << "Temp interval number: " << i + 1 << "; left border: " << tempIntervals[i] << "; numbers quantity: " << tempRes[i] << endl;

resFile << "Temp interval number: " << i + 1 << "; left border: " << tempIntervals[i] << "; numbers quantity: " << tempRes[i] << endl;

}

cout << endl;

resFile << endl;

int\* res = new int[intCount] {0};

secondFunc(tempResSize, intCount, tempRes, tempIntervals, intervals, res);

for (int i = 0; i < intCount; i++) {

cout << "Interval number: " << i + 1 << "; left border: " << intervals[i] << "; numbers quantity: " << res[i] << endl;

resFile << "Interval number: " << i + 1 << "; left border: " << intervals[i] << "; numbers quantity: " << res[i] << endl;

}

resFile.close();

delete[] numbers;

delete[] intervals;

delete[] res;

return 0;

}

## source.asm

.MODEL FLAT, C

.CODE

firstFunc PROC C numCount: dword, tempResSize: dword, numbers: dword, tempIntervals: dword, tempRes: dword

push esi

push edi

push eax

push ebx

push ecx

push edx

mov ecx, numCount

mov esi, numbers

mov edi, tempIntervals

mov eax, 0

mov edx, tempResSize

inc edx

lp:

mov ebx, 0

check\_num:

cmp ebx, edx

jge next\_num

push eax

mov eax, [esi + eax \* 4]

cmp eax, [edi + ebx \* 4]

pop eax

jl right\_num

inc ebx

jmp check\_num

right\_num:

dec ebx

cmp ebx, -1

je next\_num

mov esi, tempRes

push eax

mov eax, [esi + ebx \* 4]

inc eax

mov [esi + ebx \* 4], eax

pop eax

mov esi, numbers

next\_num:

inc eax

loop lp

finish:

pop edx

pop ecx

pop ebx

pop eax

pop edi

pop esi

ret

firstFunc ENDP

secondFunc PROC C tempResSize: dword, intCount: dword, tempRes: dword, tempIntervals: dword, intervals: dword, res: dword

push esi

push edi

push eax

push ebx

push ecx

push edx

mov ecx, tempResSize

mov esi, tempIntervals

mov edi, intervals

mov eax, 0

mov edx, intCount

inc edx

lp:

mov ebx, 0

check\_int:

cmp ebx, edx

jge next\_int

push eax

mov eax, [esi + eax \* 4]

cmp eax, [edi + ebx \* 4]

pop eax

jl right\_int

inc ebx

jmp check\_int

right\_int:

dec ebx

cmp ebx, -1

je next\_int

mov esi, tempRes

push eax

mov eax, [esi + eax \* 4]

mov esi, res

push ecx

mov ecx, [esi + ebx \* 4]

add ecx, eax

mov [esi + ebx \* 4], ecx

pop ecx

pop eax

mov esi, tempIntervals

next\_int:

inc eax

loop lp

finish:

pop edx

pop ecx

pop ebx

pop eax

pop edi

pop esi

ret

secondFunc ENDP

END