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

ОТЧЕТ

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

Студент гр. 0383 _____ Смирнов И.А.
Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Используя связь Ассемблера с ЯВУ, написать программу частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

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. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при его отсутствии получить у преподавателя).
Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. 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] (м.б. биполярный, например, [-100, 100])
3. Массив псевдослучайных целых чисел $\{X_i\}$.
4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)
5. Массив левых границ интервалов разбиения LGrInt .

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину, левые границы могут задаваться в произвольном порядке и иметь произвольные значения. Если Xmin < LGrInt(1), то часть данных не будет участвовать в формировании распределения. Каждый интервал, кроме последнего, следует интерпретировать как [LGrInt(i), LGrInt(i+1)). Если у последнего интервала правая граница меньше Xmax, то часть данных не будет участвовать в формировании распределения.

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.

Вариант 15

Вид распределения: равномерно, число процедур 2, Nint >= D_x, Lgi <= X_{min}, ПГпосл <= X_{max}

Замечания:

1) На ЯВУ следует реализовать только ввод исходных данных (возможно с контролем), вывод и генерацию псевдослучайных целых чисел. Всю остальную функциональность следует программировать на ассемблере.

2) В отладочной версии программы (при небольшом количестве псевдослучайных чисел, не превышающем 100 значений) для контроля работы датчика сгенериро-ванные числа, приведенные к целому виду, следует выводить на экран или в файл. В основной версии программы, предоставляемой для защиты, вывод сгенерированных псевдослучайных чисел выполнять не нужно.

Ход работы.

В начале программы происходит считывание параметров согласно условиям варианта. Левые границы сортируются по возрастанию. Генерация рандомных чисел происходит с помощью генератора mt19937 библиотеки random. Далее инициализируются два массива pre_answer и final_answer, первый массив после вызове функции pre_func будет содержать в себе промежуточный результат с интервалами единичной длины. Второй же массив используется в функции final_func и будет содержать в себе конечную таблицу распределений чисел по интервалам.

Рассмотрим два Ассемблерных модуля:

1) pre_func – функция для распределения чисел по единичным интервалам, оперируя сгенерированными числами, их количеством и минимальной границей генерации. На выходе мы получаем массив чисел, распределенных по единичным интервалам.
2) final_func – в данной функции мы перебираем все интервалы и вычисляем, входит ли единичный интервал в данный. Далее суммируем количество чисел в единичных интервалах и записываем результат.

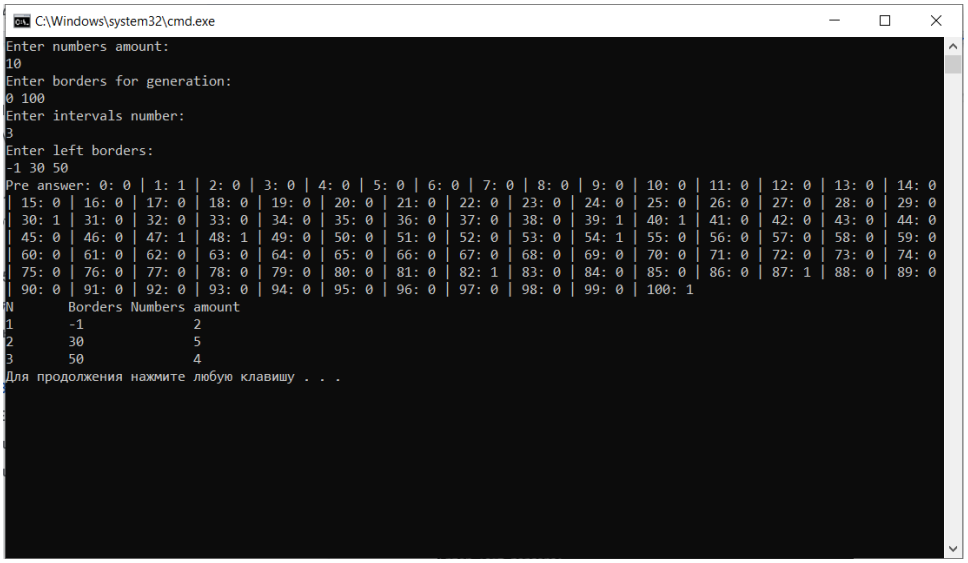
Результат записывается в переменную final_answer. В конце программы выводим таблицу результатов по массиву final_answer на экран и записываем в файл.

Вывод.

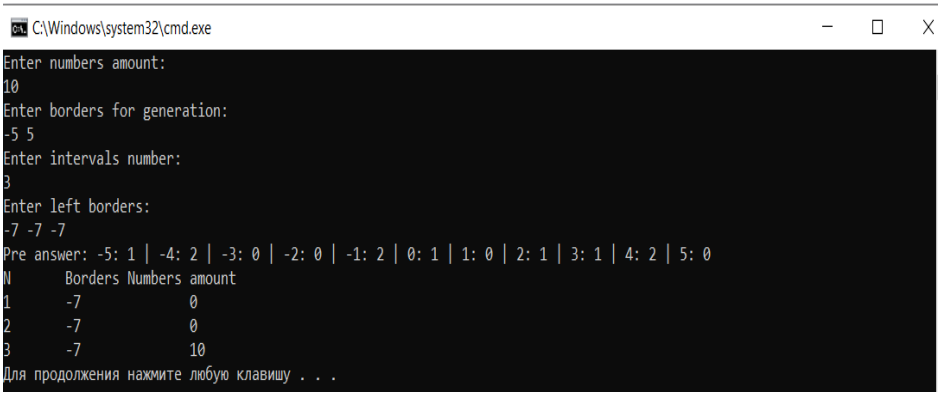
В ходе данной лабораторной работы была написана программа построения частного распределения попаданий псевдослучайных чисел в заданные интервалы с использованием связи Ассемблера и ЯВУ.

ТЕСТИРОВАНИЕ

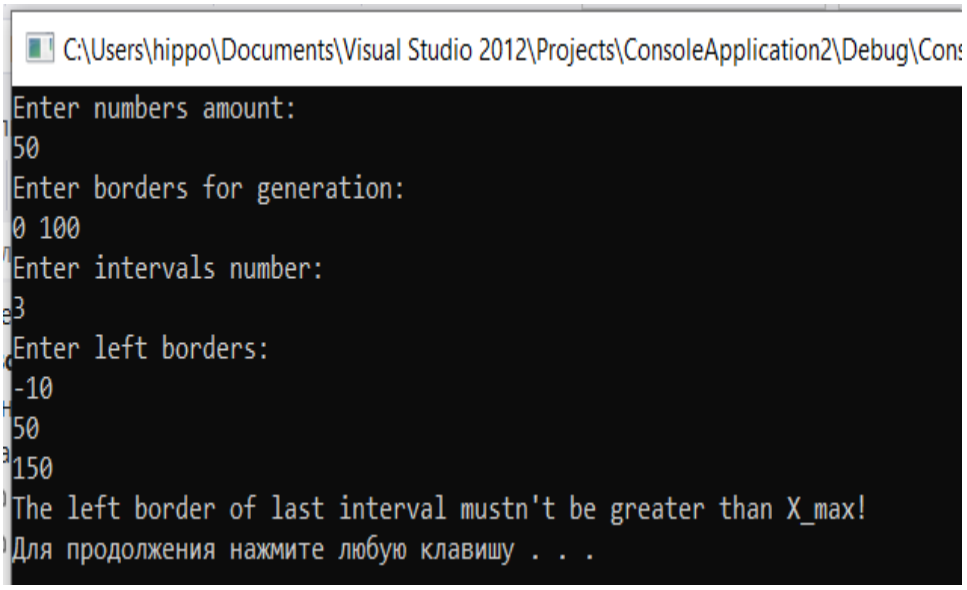
Тест 1:



Тест 2:



Тест 3:



ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Файл main.cpp

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <random>
#include <string>

using namespace std;

extern "C" void pre_func(int* numbers, int Num_Ran_Dat, int*
pre_answer, int X_min);
extern "C" void final_func(int* intervals, int N_int, int*
pre_answer, int X_min, int* final_answer);

int main() {
     int Num_Ran_Dat, X_min, X_max, N_int;
     cout << "Enter numbers amount:" << endl;
     cin >> Num_Ran_Dat;
     cout << "Enter borders for generation:" << endl;
     cin >> X_min >> X_max;
     cout << "Enter intervals number:" << endl;
     cin >> N_int;

     if (N_int <= 0 || N_int > 24) {
          cout << "Intervals number must be from 0 to 24!" << endl;
          system("Pause");
          return 0;
     }

     cout << "Enter left borders:" << endl;
     auto intervals = new int[N_int + 1];
     for (int i = 0; i < N_int; ++i) {
          cin >> intervals[i];
     }
     intervals[N_int] = X_max;

     if (intervals[0] > X_min) {
          cout << "First border mustn't be greater than X_min" <<
endl;
          system("Pause");
          return 0;
     }

     for (int i = 0; i < N_int + 1; i++) {
          for (int j = i; j < N_int + 1; j++) {
               if (intervals[i] > intervals[j]) {
                    swap(intervals[i], intervals[j]);
               }
          }
     }

     if (intervals[N_int] > X_max) {
```

```
          cout << "The left border of last interval mustn't be
greater than X_max!" << endl;
          system("Pause");
          return 0;
     }

     auto numbers = new int[Num_Ran_Dat];
     random_device rd;
     mt19937 generator(rd());
     uniform_int_distribution<> dist(X_min, X_max);
     for (int i = 0; i < Num_Ran_Dat; i++) {
          numbers[i] = dist(generator);
     }


     auto pre_answer = new int[abs(X_max - X_min) + 1];
     auto final_answer = new int[N_int];
     for (int i = 0; i < abs(X_max - X_min) + 1; i++) {
          pre_answer[i] = 0;
     }
     for (int i = 0; i < N_int; i++) {
          final_answer[i] = 0;
     }

     pre_func(numbers, Num_Ran_Dat, pre_answer, X_min);
     cout << "Pre answer: ";
     for (int i = 0; i < abs(X_max - X_min); i++) {
          cout << i + X_min << ": " << pre_answer[i] << " | ";
     }
     cout << to_string(abs(X_max - X_min) + X_min) << ": " <<
pre_answer[abs(X_max - X_min)] << endl;
     final_func(intervals, N_int, pre_answer, X_min, final_answer);

     ofstream file("output.txt");
     auto str = "N\tBorders\tNumbers amount";
     file << str << endl;
     cout << str << endl;
     for (int i = 0; i < N_int; i++) {
          auto str_res = to_string(i + 1) + "\t" +
to_string(intervals[i]) + "\t\t" + to_string(final_answer[i]) +
"\n";
          file << str_res;
          cout << str_res;
     }

     system("pause");
     return 0;
}
```

Файл Module1.asm

```
.MODEL FLAT, C
.CODE

PUBLIC C
```

```
final_func
final_func
PROC C
intervals:
dword, N_int:
dword,
pre_answer:dwo
rd, X_min:
dword,
final_answer:
dword

push esi
push edi

mov esi,
intervals
mov edi,
final_answer
mov ecx, N_int

start:
     mov eax,
[esi]
     mov ebx,
[esi+4]

     cmp ebx,
X_min
     jle mark1

     cmp eax,
X_min
     jge mark2

     sub ebx,
X_min
     inc ebx
     mov eax, 0
     jmp mark4

     mark2:
          sub
ebx, eax
          inc
ebx
          sub
eax, X_min

     mark4:
          push
esi
          push
ecx
```

```
          mov
esi,
pre_answer
          mov
ecx, ebx
          mov
ebx, 0
     start2:
          add
ebx,
[esi+4*eax]
          inc
eax
          loop
start2

     mov [edi],
ebx
     add edi, 4

     pop ecx
     pop esi
     jmp mark3
     mark1:
          mov
ebx, 0
          cmp
ecx, 1
          jne
mark5
          mov
esi,
pre_answer
          mov
eax, 0
          add
ebx,
[esi+4*eax]

mark5:

     mov [edi],
ebx

     add edi, 4
     mark3:
          add
esi, 4
          loop
start

pop edi
pop esi

ret
```

```
final_func
endp
end
```

Файл Module2.asm

```
.586
.MODEL FLAT,
C
.CODE

PUBLIC C
pre_func
pre_func
PROC C
numbers:
dword,
Num_Ran_Dat:
dword,
pre_answer:
dword,
X_min: dword

push esi
push edi

mov ecx,
Num_Ran_Dat
mov esi,
numbers
mov edi,
pre_answer

start:
     mov eax,
[esi]
     sub eax,
X_min
     mov ebx,
[edi+4*eax]
     inc ebx
     mov
[edi+4*eax],
ebx
     add esi,
4
     loop
start

pop edi
pop esi

ret
pre_func
endp
END
```