# 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 и систем»

**Тема: Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы Вариант 11**

|  |  |  |
| --- | --- | --- |
| Студент гр.0382 |  | Мамин Р.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

# Цель работы.

Написать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы, используя связь ассемблера с ЯВУ.

# 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 (пpи его отсутствии получить у пpеподавателя).

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, сфор-мированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

# Вариант №11:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Вид распредения | Число ассем. процедур | Nint  ≥ Dx | Nint  <  Dx | Lgi  ≤ Xmin | Lg1 >  Xmin | ПГпосл  ≤ Xmax | ПГпосл >Xmax |
| 11 | равном. | 2 | - | + | + | - | + | - |

**Выполнение работы:**

Программа на ЯВУ:

В самом начале программы, используя ключевое слово extern “C” связываем программу с функциями first\_dist и second\_dist, что реализованы на ассемблере в файлах с соответствующими названиями. Первая – распределяет сгенерированные числа по единичным интервалам. Вторая же, используя первое распределение, находит распределение чисел по заданным интервалам.

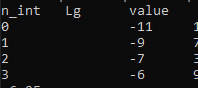
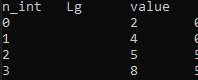
В функции main() вводим исходные данные, согласно условиям задач; в случае каких-то несоответствий – выводим сообщение и завершаем программу. Если все было введено верно – сортируем границы интервалов по возрастанию (чтобы избежать интервалов вида [7, 1)). Затем массив, сгенерированный с помощью генератора mt19937 в соответствии с равномерным распределением, чисел обрабатывается функциями, написанными на ассемблере. Результат выводится в консоль и в файл “text.txt”.

О функции first\_dist:

Берется число из массива сгенерированных чисел. Вычисляем единичный интервал, куда оно входит. Затем вычисляем индекс для этого интервала в массива с результатом и записываем его туда.

О функции second\_dist:

Рассмотрим несколько ситуаций: по условию, все левые границы интервалов <= Xmin. Это значит, что все интервалы, кроме последнего, будут иметь правую границу <= Xmin. Последний же интервал будет иметь левую границу точно <= Xmin, а правую <= Xmax. Поэтому его имеет смысл рассматривать от Xmin до правой границы (если она не <= Xmin, тогда все сводится к предыдущей ситуации). По итогу, если мы попадаем в первую ситуацию, то количество чисел, входящих в данный интервал, либо 0, либо количество Xmin. Если же попали во вторую ситуацию, то вычисляем все единичные интервалы, что входят в текущий, суммируем количество чисел в

этих единичных интервалах и также записываем элемент массива с соответствующим интервалу индексу.

Исходный код программы см. в приложении А.

# Тестирование:

Для проверки работоспособности были проведены тесты, результаты представлены в таблице 1.

Таблица 1 – Тестирование и результаты

|  |  |  |  |
| --- | --- | --- | --- |
| Номер  теста | Входные данные | Выходные данные | Верность  результата |
| 1 |  |  | верно |
| 2 |  |  | верно |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 |  |  | верно |

# Выводы.

В ходе работы была написана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы с использованием связи ассемблера и ЯВУ.

# ПРИЛОЖЕНИЕ А

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: first\_dist.asm

.586

.MODEL FLAT, C

.CODE

PUBLIC C first\_dist

first\_dist PROC C numbers: dword, n: dword, result1: dword, x\_min: dword, mid: dword

push esi push edi

mov esi, numbers mov edi, result1 mov ecx, n

;fild dword ptr mid ; ST(0) = 0

lp:

mov eax, [esi] sub eax, x\_min

mov ebx, [edi+4\*eax] inc ebx

mov [edi+4\*eax], ebx add esi, 4

loop lp

mov ecx, n

mov esi, numbers sub ecx, 1

mov edi, mid

fild dword ptr [esi] ; ST(1) = 0; ST(0) = number

fild dword ptr n ; ; ST(2) = 0; ST(1) = number; ST(0) = n fdiv

add esi, 4

lp2:

fild dword ptr [esi] ; ST(1) = 0; ST(0) = number

fild dword ptr n ; ; ST(2) = 0; ST(1) = number; ST(0) = n fdiv

fadd

add esi, 4 loop lp2

fst dword ptr [edi]

pop edi pop esi

ret

first\_dist endp end

Название файла: second\_dist.asm

.MODEL FLAT, C

.CODE

PUBLIC C second\_dist

second\_dist PROC C result1: dword, intervals: dword, n\_int:dword, result2: dword, x\_min: dword

push esi push edi

mov esi, intervals mov edi, result2 mov ecx, n\_int

lp:

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

cmp eax, x\_min jge m2

mov eax, 0

sub ebx, x\_min cmp ebx, 0

jle m1 jmp m6 m2:

sub ebx, eax cmp ebx, 0 je m1

sub eax, x\_min

m6:

push esi push ecx

mov esi, result1 mov ecx, ebx

mov ebx, 0 lp2:

add ebx, [esi+4\*eax] inc eax

loop lp2

pop ecx cmp ecx, 1 jne m4

add ebx, [esi+4\*eax] m4:

mov [edi], ebx pop esi

jmp m3

m1:

mov ebx, 0 mov [edi], ebx

m3:

add edi, 4

add esi, 4

loop lp

pop edi pop esi

ret

second\_dist endp end

Название файла: main.cpp

#include <iostream> #include <stdio.h> #include <fstream> #include <string> #include <random>

extern "C" void first\_dist(int\* numbers, int n, int\* result1, int x\_min, float\* mid);

extern "C" void second\_dist(int\* result1, int\* intervals, int n\_int, int\* result2, int x\_min);

using namespace std; int main()

{

float mid = 0;

int n, x\_min, x\_max, n\_int;

cout << "enter the length of the array" << endl; cin >> n;

if (n <= 0 || n > 16000) {

cout << "0 < n <= 16000!!" << endl;

system("Pause"); return 0;

}

cout << "enter xMin" << endl; cin >> x\_min;

cout << "enter xMax" << endl; cin >> x\_max;

if (x\_max <= x\_min) {

cout << "xMax > xMin!" << endl; system("Pause");

return 0;

}

cout << "enter the number of the intervals" << endl; cin >> n\_int;

if (n\_int <= 0 || n\_int > 24) {

cout << "0 < n\_int <= 24!!" << endl; system("Pause");

return 0;

}

if (n\_int > (abs(x\_max - x\_min))) {

cout << "n\_int <= x\_max - x\_min!!" << endl; system("Pause");

return 0;

}

int\* intervals = new int[n\_int + 1]; for (int i = 0; i < n\_int; i++) {

cout << "enter the Lg of " << i << " interval" << endl; cin >> intervals[i];

}

intervals[n\_int] = x\_max;

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[0] > x\_min) {

cout << "Lg1 <= x\_min!!" << endl; system("Pause");

return 0;

}

if (intervals[n\_int] > x\_max) {

cout << "Rg <= x\_max!!" << endl; system("Pause");

return 0;

}

int\* numbers = new int[n]; random\_device rd;

mt19937 gen(rd()); uniform\_int\_distribution<> dis(x\_min, x\_max); for (int i = 0; i < n; i++) {

numbers[i] = dis(gen);

cout << numbers[i] << " ";

}

cout << endl;

int\* result1 = new int[abs(x\_max - x\_min) + 1]; int\* result2 = new int[n\_int];

for (int i = 0; i < abs(x\_max - x\_min) + 1; i++) { result1[i] = 0;

}

for (int i = 0; i < n\_int; i++) { result2[i] = 0;

}

first\_dist(numbers, n, result1, x\_min, &mid); for (int i = 0; i < abs(x\_max - x\_min); i++) {

cout << i + x\_min << ": " << result1[i] << " | ";

}

cout << to\_string(abs(x\_max - x\_min) + x\_min) << ": " << result1[abs(x\_max - x\_min)] << endl;

second\_dist(result1, intervals, n\_int, result2, x\_min);

ofstream file("table.txt"); auto head = "n\_int\tLg\tvalue"; file << head << endl;

cout << head << endl;

for (int i = 0; i < n\_int; i++) {

auto line = to\_string(i) + "\t\t" + to\_string(intervals[i]) + "\t" + to\_string(result2[i]) + "\n";

file << line; cout << line;

}

cout << mid << endl; system("Pause"); return 0;

}