**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 распределения попаданий псевдослучайных целых чисел в заданные интервалы**

**Вариант 20**

Студент гр. 1304 Новицкий М. Д

Преподаватель Кирьянчиков В.А.

Санкт-Петербург

2023

**Цель работы.**

Изучение основных принципов организации связи ассемблера с ЯВУ. Написание программы построения 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 (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.

***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])

1. Массив псевдослучайных целых чисел {Xi}.
2. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt ( <=24 )
3. 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.

Задание на разработку программы выбирается из таблицы 1 в зависимости от номера студента в группе. Варианты заданий различаются:

1. видом распределения псевдослучайных чисел: равномерное или нормальное (гаусовское);
2. количеством ассемблерных модулей, формирующих требуемое распределение:
   * если указан 1 модуль, то он сразу формирует распределение по заданным интервалам и возвращает его в головную программу, написанную на ЯВУ;
   * если указаны 2 модуля, то первый из них формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат (это распределение должно выводится на экран для контроля); затем вызывается второй модуль который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами).

Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла.

1. условием – может ли число интервалов быть больше-равно (Nint ≥ Dx) или меньше (Nint < Dx) диапазона изменения входных чисел;
2. условием – может ли первая левая граница быть больше Xmin (Lg1 >

Xmin) или могут ли какие-то левые границы быть меньше Xmin (Lgi ≤ Xmin);

1. условием – может ли правая граница последнего интервала быть больше Xmax (ПГпосл > Xmax) или меньше-равна Xmax (ПГпосл > Xmax) .

Вариант 20

Вид распределения: нормальное.

Число ассемблерных процедур: 2.

Nint < Dx и Lgi <= Xmin.

**Выполнение работы.**

В ходе выполнения лабораторной работы было написано три файла: main.cpp, units\_counting.asm, intervals\_counting.asm.

Рассмотрим файл main.cpp. После подключения всех необходимых заголовочных файлов объявляются две внешние функции: unit\_distribution

и intervals\_distribution.

extern "C" void unit\_distribution(int\* numbers, int n, int\* res, int x\_min);

extern "C" void intervals\_distribution(int\* intervals, int n\_int, int\* units, int n\_units, int x\_min, int\* res);

Эти функции описаны в файлах units\_counting.asm и intervals\_counting.asm соответственно. В главной функции программы происходит считывание количества генерируемых чисел, их минимальное и максимальное значения, интервалы.

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

cout << "Enter amount of numbers:" << endl;

cin >> n;

cout << "Enter Xmin and Xmax seperated by space:" << endl;

cin >> x\_min >> x\_max;

cout << "Enter number of intervals:" << endl;

cin >> n\_int;

auto intervals = new int[n\_int + 1];

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

cin >> intervals[i];

}

intervals[n\_int] = x\_max;

double mean = (x\_min + x\_max) / 2;

double sigma = (x\_max - x\_min) / 4;

После этого происходит генерация случайных чисел в соответствии с заданными ограничениями.

random\_device r\_d;

mt19937 generator(r\_d());

normal\_distribution<double> distribution(mean, sigma);

auto numbers = new int[n];

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

int number = distribution(generator);

while (number < x\_min || number > x\_max)

number = distribution(generator);

numbers[i] = number;

cout << number << " ";

if ((i + 1) % 50 == 0) cout << "\n";

}

cout << endl;

auto units = new int[x\_max - x\_min + 1];

auto result = new int[n\_int + 1];

for (int i = 0; i < x\_max - x\_min + 1; ++i)

units[i] = 0;

for (int i = 0; i < n\_int + 1; ++i)

result[i] = 0;

float mid = 0.0;

После этого происходит вызов функции unit\_distribution.

unit\_distribution(numbers, n, units, x\_min);

После вызова данной функции массив units будет содержать в каждой ячейке частоту появления числа в сгенерированной последовательности (со

смещением в x\_min).

После этого вызывается функция intervals\_distribution, и выводится ответ.

intervals\_distribution(intervals, n\_int, units, x\_max - x\_min + 1, x\_min, result);

cout << "\n\n";

auto head = "N\tLeft border\tAmount of numbers";

cout << head << endl;

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

auto row = to\_string(i + 1) + "\t" + to\_string(intervals[i]) + "\t\t" + to\_string(result[i]) + "\n";

cout << row;

}

Рассмотрим файл units\_counting.asm, содержащий реализацию функции unit\_distribution. Данная функция принимает на вход массив сгенерированных чисел, их количество, массив для записи результата и ограничения этих чисел снизу.

PUBLIC C unit\_distribution

unit\_distribution PROC C numbers: dword, n: dword, res: dword, xmin: dword

mov esi, numbers

mov edi, res

mov ecx, n

mov edx, 0h

В теле функции происходит перебор всех чисел в массиве с записью каждого из них в соответствующую ячейку в массиве результата ([сгенерированное число] – [x\_min]).

start\_loop:

mov eax, [esi]

sub eax, xmin

mov ebx, [edi + 4\*eax]

add ebx, 1

mov [edi + 4\*eax], ebx

add esi, 4

loop start\_loop

Рассмотрим файл intervals\_counting, содержащий описание функции intervals\_distribution, принимающей на вход массив интервалов, их количество, массив units, его размер, минимальное значение и массив для

записи результата.

PUBLIC C intervals\_distribution

intervals\_distribution PROC C intervals: dword, n\_int: dword, units: dword, n\_units: dword, x\_min: dword, res: dword

mov esi, intervals

mov edi, res

mov ecx, n\_int

В данной функции посредством массива units подсчитываются количество всех чисел в заданном интервале и записывается в массив для записи результата.

start\_loop:

mov eax, [esi]

add esi, 4h

mov ebx, [esi]

sub ebx, eax

sub eax, x\_min

cmp eax, 0

jge to\_units

neg eax

sub ebx, eax

mov eax, 0h

cmp ebx, 0

jg to\_units

sub ecx, 1

add edi, 4h

jmp start\_loop

to\_units:

push ecx

push esi

mov ecx, ebx

mov ebx, 0h

mov esi, units

start\_loop2:

add ebx, [esi + eax\*4]

add eax, 1

loop start\_loop2

mov [edi], ebx

add edi, 4h

pop esi

pop ecx

loop start\_loop

sub edi, 4h

mov esi, units

mov ebx, [esi + eax\*4]

add [edi], ebx

**Тестирование.**

Тестирование см. в приложении Б. По результатам тестирования был сделан вывод, что программа работает верно.

**Выводы.**

В ходе работы были изучены основные принципы организации связи ассемблера с ЯВУ. Была написана программа, частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Файл main.cpp

#include <iostream>

#include <fstream>

#include <random>

#include <string>

using namespace std;

extern "C" void unit\_distribution(int\* numbers, int n, int\* res, int x\_min, float\* mid);

extern "C" void intervals\_distribution(int\* intervals, int n\_int, int\* units, int n\_units, int x\_min, int\* res);

int main() {

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

cout << "Enter amount of numbers:" << endl; cin >> n;

cout << "Enter Xmin and Xmax seperated by space:" << endl; cin >> x\_min >> x\_max;

cout << "Enter number of intervals:" << endl;

cin >> n\_int;

if (n\_int >= (x\_max - x\_min)) { cout << "Nint < D\_x must be" << endl;

return 0;

}

cout << "Enter Lgi seperated by spaces:" << endl; auto intervals = new int[n\_int + 1];

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

cin >> intervals[i];

}

intervals[n\_int] = x\_max;

double mean = (x\_min + x\_max) / 2; double sigma = (x\_max - x\_min) / 4; if (!sigma) sigma = 1;

cout << "\nNumbers generated with normal distribution (mean = "

<< mean << ", sigma = " << sigma << ").\n" << endl;

random\_device r\_d; mt19937 generator(r\_d());

normal\_distribution<double> distribution(mean, sigma);

auto numbers = new int[n]; for (int i = 0; i < n; ++i) { int number = distribution(generator); while (number < x\_min || number > x\_max) number = distribution(generator); numbers[i] = number; cout << number << " "; if ((i + 1) % 50 == 0) cout << "\n";

}

cout << endl;

auto units = new int[x\_max - x\_min + 1]; auto result = new int[n\_int + 1]; for (int i = 0; i < x\_max - x\_min + 1; ++i)

units[i] = 0;

for (int i = 0; i < n\_int + 1; ++i) result[i] = 0; float mid = 0.0;

unit\_distribution(numbers, n, units, x\_min);

cout << "units:" << "\n";

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

cout << units[i] << " ";

}

intervals\_distribution(intervals, n\_int, units, x\_max - x\_min +

1, x\_min, result);

cout << "\n\n";

auto head = "N\tLeft border\tAmount of numbers";

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

auto row = to\_string(i + 1) + "\t" + to\_string(intervals[i])

+ "\t\t" + to\_string(result[i]) + "\n";

cout << row;

}

return 0;

}

Файл units\_counting.asm

.MODEL FLAT, C

.DATA

.CODE

PUBLIC C unit\_distribution

unit\_distribution PROC C numbers: dword, n: dword, res: dword, xmin: dword mov esi, numbers mov edi, res mov ecx, n

mov edx, 0h

start\_loop:

mov eax, [esi] sub eax, xmin

mov ebx, [edi + 4\*eax] add ebx, 1 mov [edi + 4\*eax], ebx add esi, 4 loop start\_loop

ret

unit\_distribution ENDP

END

Файл intervals\_counting.asm

.MODEL FLAT, C

.CODE

PUBLIC C intervals\_distribution intervals\_distribution PROC C intervals: dword, n\_int: dword, units: dword, n\_units: dword, x\_min: dword, res: dword mov esi, intervals mov edi, res mov ecx, n\_int

start\_loop:

mov eax, [esi] add esi, 4h mov ebx, [esi] sub ebx, eax

sub eax, x\_min

cmp eax, 0 jge to\_units neg eax sub ebx, eax mov eax, 0h cmp ebx, 0 jg to\_units sub ecx, 1 add edi, 4h

jmp start\_loop

to\_units:

push ecx push esi

mov ecx, ebx mov ebx, 0h

mov esi, units

start\_loop2: add ebx, [esi + eax\*4] add eax, 1 loop start\_loop2 mov [edi], ebx add edi, 4h

pop esi pop ecx

loop start\_loop

sub edi, 4h

mov esi, units

mov ebx, [esi + eax\*4]

add [edi], ebx

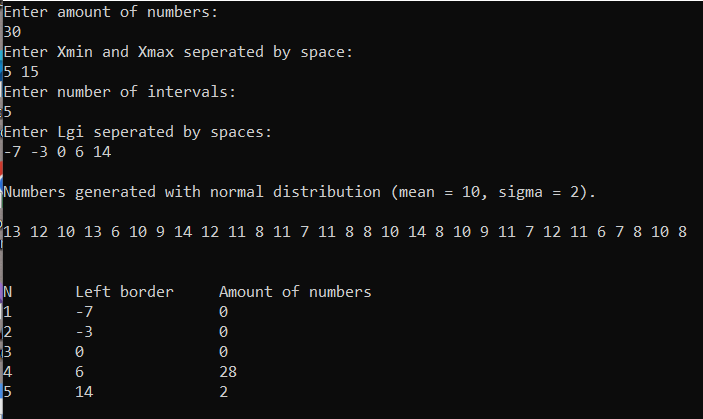
ret

intervals\_distribution ENDP END

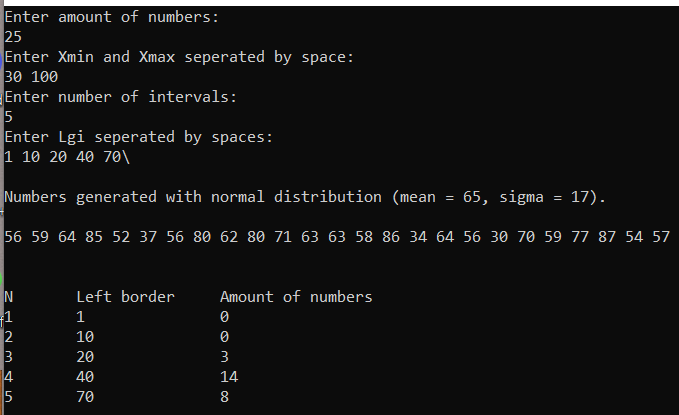
**ПРИЛОЖЕНИЕ Б**

**ТЕСТИРОВАНИЕ**

Тест 1.



Тест 2.



Тест 3.

