**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**по лабораторной работе №6**

**по дисциплине «Организация ЭВМ и систем»**

**Тема: «Организация связи Ассемблера с ЯВУ на примере программы построения частного распределение попаданий псевдослучайных целых чисел в заданные интервалы».**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1384 |  | Усачева Д.В. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

**Вариант 26.**

**Цель работы.**

Изучить закон распределения и написать программу с вызывающейся ассемблерной процедурой.

**Задание.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Вид распредения | Число ассем. процедур | Nint ≥ Dx | Nint < Dx | Lgi ≤ Xmin | Lg1 > Xmin | ПГпосл ≤ Xmax | ПГпосл > Xmax |
| 26 | нормал. | 1 | - | + | - | + | - | + |

1. Распределение нормальное (Гауссовское).
2. 1 Ассемблерный модуль.
3. Число интервалов меньше диапазона изменения входных чисел.
4. Первая левая граница больше Xmin.
5. Правая граница последнего интервала больше Xmax.

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 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 {Xi}.

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.

**Выполнение работы.**

Были созданы файлы main.cpp и ассемблерный модуль module.asm. Вначале программы происходит считывание всех необходимых данных, которые видит пользователь, с их проверкой на критические случаи

Далее происходит инициализация массива рандомными числами в пределах [Xmin, Xmax]. Для генерации рандомных чисел используется функция из папки RAND\_GEN. Сначала происходит начальная инициализация с помощью dnk\_randomize(). И поскольку, распределение нормальное, то используется функция dnk\_normal(M, S), где M – это мат. ожидание, а S – дисперсия. В качестве мат. ожидания берётся среднее арифметическое между Xmin и Xmax, а для дисперсии треть разности Xmin и Xmax. Если число не попало в заданный интервал, то происходит проверка.

Далее инициализируется массив ответов, в котором будет храниться подсчёт всех повторяющихся цифр и вызывание процедуры.

Для подсчёта чисел, входящих в соответсвующий интервал написана функция на ассемблере. В нём реализована необходимая функция, принимающая на вход массив чисел, длину массива, массив границ, длину массива границ и массив куда надо записать результат работы функции — количество чисел, входящих в интервал.

В самой функции сгенерированные числа сравниваются с левыми границами, для этого был реализован вложенный цикл, который для каждой левой границы находит число элементов больших, чем она.

Результат работы программы записывается в файл и выводится на экран.

Все числа объявлены как dword, так как являются значениями int, то есть 4 байта. Происходит сохранения в стек расширенных регистров и затем присваивания esi, edi и eax новыми значениями.

**Тестирование**

Комментарий: Правая граница последнего интервала меньше необходимого значения. Программа работает верно

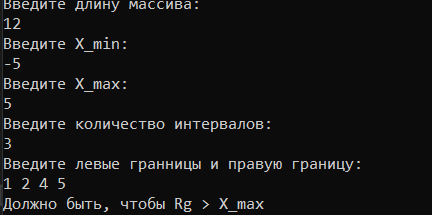


Рисунок 1. Первый тест

Комментарий: Число интервалов больше диапазона изменения входных чисел. Программа работает верно

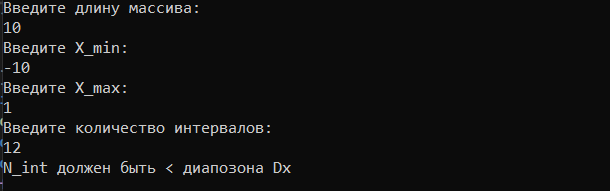


Рисунок 2. Второй тест

Комментарий: Первая левая граница равна необходимому значению. Программа работает верно.

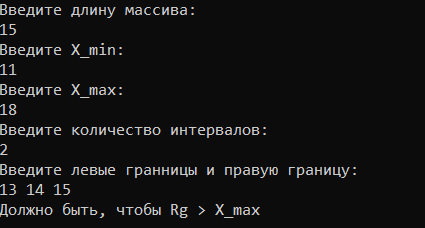


Рисунок 3. Третий тест

Комментарий: Длина массива псевдослучайных целых чисел превосходит максимальную. Программа работает верно.

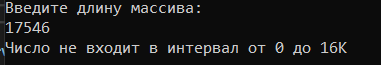
****

Рисунок 4. Четвертый тест

Комментарий: Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел превосходит максимальный. Программа работает верно.

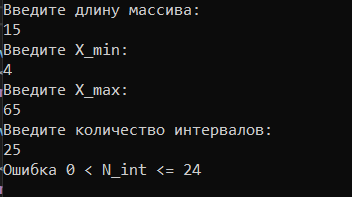
****

Рисунок 5. Пятый тест

Комментарий: Данные введены корректно. Программа работает верно.

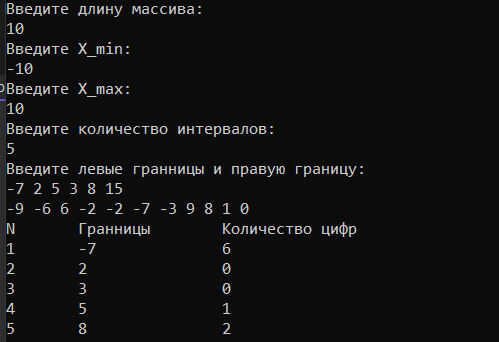


Рисунок 6. Шестой тест

**Выводы.**

Рассмотрен способ организации связи ассемблера с ЯВУ. Разработана программа, строящая частотное распределение попадание случайных чисел, сгенерированных функцией рандома.

**Приложение A  
Исходный код программ**

Файл main.cpp

#include <iostream>

#include <stdio.h>

#include <fstream>

#include <string>

#include <random>

#include <locale>

#define RAND\_A 7.74597E-1

#define RAND\_M 1.37438953472E+11

using namespace std;

extern "C" void func(int\* intervals, unsigned int N\_int, unsigned int N, int\* numbers, int\* final\_answer);

int cmp(const void\* a, const void\* b) {

return \*(int\*)a - \*(int\*)b;

}

static double xrand;

double dnk\_random(void)

{

xrand \*= 3125.0;

xrand = xrand - ((long)(xrand / RAND\_M)) \* RAND\_M;

return xrand / RAND\_M;

}

double dnk\_normal(double M, double S)

{

int i;

double sum = 0.0, xn;

for (i = 0; i < 5; i++)

{

xn = (2.0 \* dnk\_random() - 1.0) \* RAND\_A;

sum += xn;

}

return M + S \* sum \* (1.0 + 0.01 \* (sum \* sum - 3.0));

}

void dnk\_randomize(void)

{

char k;

xrand = 1.0 \* rand() / RAND\_MAX;

for (k = 0; k < 10; k++)

dnk\_random();

}

int main()

{

setlocale(LC\_ALL, "Russian");

int N, X\_min, X\_max, N\_int;

cout << "Введите длину массива:\n";

cin >> N;

if (N <= 0 || N > 16000)

{

cout << "Число не входит в интервал от 0 до 16К\n";

return 0;

}

cout << "Введите X\_min:\n";

cin >> X\_min;

cout << "Введите X\_max:\n";

cin >> X\_max;

if (X\_min >= X\_max)

{

cout << "X\_min больше X\_max\n";

return 0;

}

cout << "Введите количество интервалов:\n";

cin >> N\_int;

if (N\_int <= 0 || N\_int > 24)

{

cout << "Ошибка 0 < N\_int <= 24\n";

return 0;

}

if (N\_int >= X\_max - X\_min)

{

cout << "N\_int должен быть < диапозона Dx\n";

return 0;

}

cout << "Введите левые гранницы и правую границу:\n";

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

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

{

cin >> intervals[i];

if (intervals[i] < X\_min)

{

cout << "Гранница должна быть в интервале [X\_min, X\_max]\n";

delete[] intervals;

return 0;

}

}

if (intervals[0] <= X\_min) {

cout << "Должно быть, чтобы Lg > X\_min\n";

return 0;

}

if (intervals[N\_int] <= X\_max)

{

cout << "Должно быть, чтобы Rg > X\_max\n";

return 0;

}

int\* numbers = new int[N];

dnk\_randomize();

double M = (X\_max + X\_min) / 2.0;

double S = (X\_max - X\_min) / 3.0;

for (int i = 0; i < N; i++)

{

numbers[i] = (int)round(dnk\_normal(M, S));

if (numbers[i] > X\_max || numbers[i] < X\_min)

i--;

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

}

cout << endl;

auto final\_answer = new int[N\_int];

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

{

final\_answer[i] = 0;

}

qsort(intervals, N\_int + 1, sizeof(int), cmp);

func(intervals, N\_int, N, numbers, final\_answer);

ofstream file("output.txt");

auto str = "N\tГранницы\tКоличество цифр";

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;

}**Приложение B**

**Исходный код функции на Assembler**

Файл Module.asm

.MODEL FLAT, C

.CODE

func PROC C intervals: dword, N\_int: dword, N: dword, numbers: dword, final\_answer: dword

push eax

push ebx

push ecx

push edi

push esi

mov esi, numbers

mov edi, final\_answer

mov eax, 0

checking\_loop:

mov ebx, 0

iter:

cmp ebx, N\_int

jge out\_cur\_iter

mov ecx, [esi + 4\*eax] ; Заменить [esi+4\*eax] на [esi+3\*eax + 1]

mov edi, intervals

cmp ecx, [edi+4\*ebx] ; Заменить [edi+4\*ebx] на [edi+2\*ebx+3]

jl out\_cur\_iter

inc ebx

jmp iter

out\_cur\_iter:

dec ebx

mov edi, final\_answer

inc dword ptr [edi+4\*ebx]; Заменить [edi+4\*ebx] на [esi+4\*ebx]

inc eax

cmp eax, N

jg exit ; Удалить

jmp checking\_loop

exit:

pop edx

pop ecx

pop ebx

pop eax

pop edi

pop esi

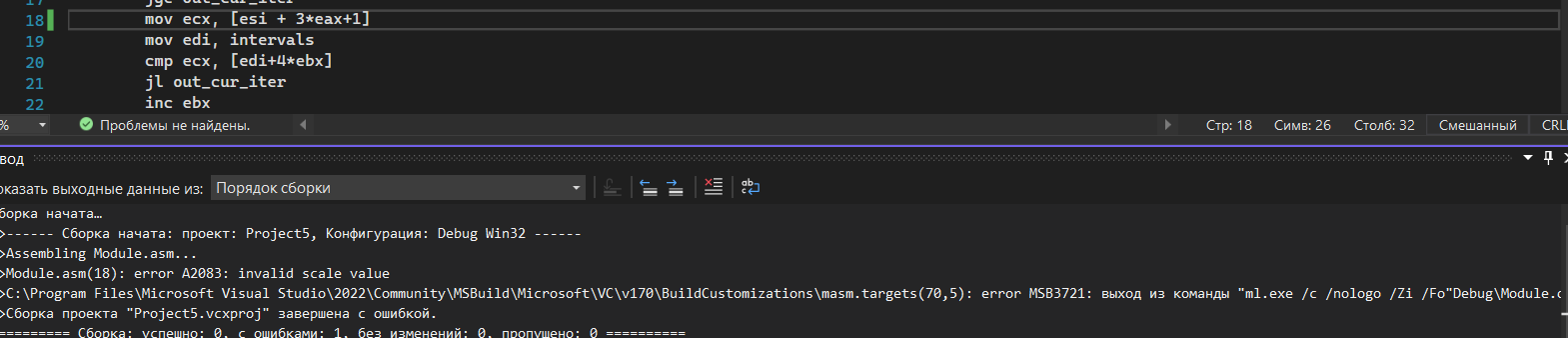
ret

func ENDP

END

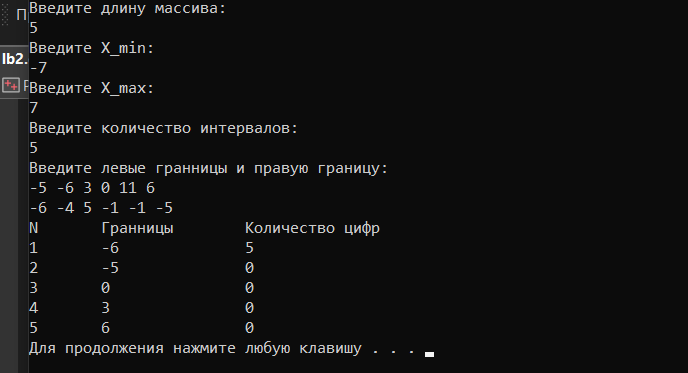
**ОТВЕТЫ НА ВОПРОСЫ.**

1. mov eax, [esi+4\*eax] – замена на [esi+3\*eax + 1]. Так как здесь используется адресация по базе с индексированием и масштабированием, то множитель индексного регистра должен быть или байтом, или словом, или двойным словом, а нас - 3. Программа не скомпилируется:



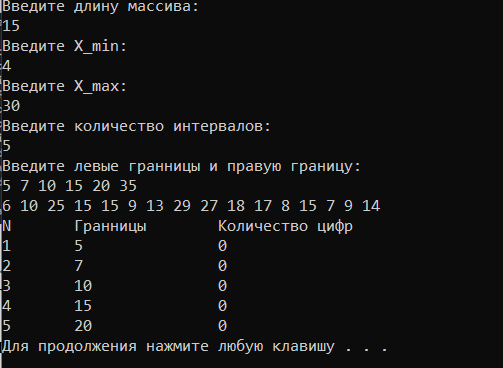
1. cmp eax, [edi+4\*ebx] – замена на cmp eax, [edi+2\*ebx + 3]. Здесь адрес будет масштабироваться для слов (множитель 2), а нам нужны двойные слова, так как массив из int (4 байта) значений. Также будет добавляться смещение. По итогу алгоритм будет сравнивать числа с неопределенными двухбайтными значениями из массива левых границ. Числа не будут распределены по интервалам.

Пример теста:



1. Замена [edi+4\*ebx] на [esi+4\*ebx] в команде inc dword ptr [edi+4\*ebx]. Так как значение eax в этот момент выполнения программы является количеством чисел в конкретном интервале, то программа запишет его в массив чисел. То есть на выходе мы не увидим распределения чисел по интервалам, а массив чисел будет содержать последовательность нулей. Если же количество интервалов будет больше, чем количество чисел, то программа аварийно завершится. Это произойдет из-за того, что индекс интервала в какой-то момент станет больше, чем размер массива чисел, и единицы будут записываться уже за пределы массива.

Тест:



1. Если удалить команду je exit, тогда произойдет ошибка, так как не будет совершаться переход, когда текущий индекс больше размера массива случайных чисел. Мы выйдем за пределы этого массива и программа завершится с ошибкой.