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 _____ Литягин С.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

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 (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, 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 $\geq D_x$	Nint $< D_x$	Lgi $\leq X_{min}$	Lg1 > X_{min}	ПГпосл $\leq X_{max}$	ПГпосл $>X_{max}$
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	enter the length of the array 5 enter xMin 0 enter xMax 10 enter the number of the intervals 2 enter the Lg of 0 interval -3 enter the Lg of 1 interval 0 enter the Rg of last interval -2	n_int Lg value 0 -3 6 1 -2 1 Для продолжения нажмите	верно
2	enter the length of the array 10 enter xMin 5 enter xMax 10 enter the number of the intervals 1 enter the Lg of 0 interval -4 enter the Rg of last interval 10	n_int Lg value 0 -4 1 Для продолжения нажмите	верно

3	enter the length of the array 50 enter xMin -5 enter xMax 10 enter the number of the intervals 5 enter the Lg of 0 interval 5	Lgi <= x_min!!	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

push esi
push edi

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

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

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 ebx, x_min ; если правая граница <= x_min, то переход по
метке
     jle m1

     sub ebx, x_min ; получаем правую границу чисел уже от 0, а не
от x_min (вычитаем x_min, т.к. левая граница, что в eax, может
быть максимум x_min)
     inc ebx ; поскольку тут считаем лишь последний интервал, надо
не забыть последнее в нем число
     mov eax, 0 ; левая граница интервала

     push esi
     push ecx

     mov esi, result1
     mov ecx, ebx
     mov ebx, 0
     lp2:
          add ebx, [esi+4*eax] ; складываем все числа в нужном диа-
позоне
          inc eax
          loop lp2

     mov [edi], ebx ; записываем в соответствующую ячейку массива
     add edi, 4

     pop ecx
     pop esi
     jmp m3
     m1:
          mov ebx, 0
          cmp ecx, 1 ; если последняя итерация, а интервал оказался
[.., x_min], то надо не забыть количество чисел, равных x_min
          jne m5
          mov esi, result1
          mov eax, 0
          add ebx, [esi+4*eax]
          m5:
          mov [edi], ebx ; записываем 0, либо кол-во = x_min
          add edi, 4
     m3:
     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);
extern "C" void second_dist(int* result1,int* intervals, int n_int,
int* result2, int x_min);


using namespace std;

int main()
{
	int n, x_min, x_max, n_int;
	cout << "enter the length of the array" << endl;
	cin >> n;
	cout << "enter xMin" << endl;
	cin >> x_min;
	cout << "enter xMax" << endl;
	cin >> x_max;
	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];
		if (intervals[i] > x_min) {
			cout << "Lgi <= x_min!!" << endl;
			system("Pause");
```

```
                return 0;
            }
        }
        cout << "enter the Rg of last interval" << endl;
        cin >> intervals[n_int];

        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 << "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);
        }

        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);
        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) << ": " << re-
sult1[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;
        }
        system("Pause");
        return 0;
        }
```