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.

.

Студент гр. 0383 _____ Козлов Т.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

На языке Спрограммируется вводс 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.

Исходные данные:

1. Длина массива псевдослучайных целыхчисел - 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 праваяграница 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.

Ход работы.

Вариант 4:

Нормальное (гауссовское) распределение чисел, две ассемблерные функции, Nint < Dx, Lgi <= Xmin, Lg1 <= Xmin, ПГпосл <= Xmax

В ходе выполнения лабораторной работы была написана программа на C++ в VS, которая проверяет входные данные согласно заданному варианту, генерирует рандомные числа (для этого создан класс *Generator,* принимающий диапазон значений, в котором будут генерироваться числа. Среднее значение для нормального распределения равно *(Xmin + Xmax) /2* , *Xmin* и *Xmax* находятся на 3 стандартных отклонения от среднего значения).

На ассемблере написано два модуля: первый распределяет числа по единичным отрезкам (используя массив mod*_result* в качетсве результата, по индексу которого находится количество сгенерированных чисел на данном единичном отрезке (т.к по сути индекс = числу)). В цикле *for_* реализованном с помощью команды *loop* программа пробегается по массиву и увеличивает значение mod_result по нужному индексу (который вычисляется через разность числа и *Xmin*).

Второй модуль распределяет числа по заданным интервалам, путем нескольких циклов *for* (реализованных с помощью команды *loop*) – он проходит по массиву *mod_result* с ограничениями в качестве интервалов, которые вычисляются через разность следующего и предыдущего интервала. Последняя граница вычисляется отдельно путем подсчета всех чисел, вошедших в результирующий массив, а затем берется разность начального количества чисел и посчитанного – и данная разность записывается в качестве результата на последнем интервале.

Табл.1: Тестирование работы lab_6

Исходные данные	Результат	Комментарий
NumDatRun = 5 Xmin = -1 Xmax = 8 Nint = 3 LGrInt = {-1, 5, 8} Сгенерированные числа: {3, 3, 0, 4, 3}	№ Граница Количество 0 -1 5 1 5 0 2 8 0	Верно
NumDatRun = 8 Xmin = -22 Xmax = 1 Nint = 5 LGrInt = {-22, -18, -4, -3, -2} Сгенерированные числа: {-12, -11, -18, -9, -12, -10, -14, -10}	№ Граница Количество 0 -22 0 1 -18 8 2 -4 0 3 -3 0 4 -2 0	Верно
NumDatRun = 6 Xmin = -1 Xmax = 14 Nint = 3 LGrInt = {7, 1, 5} Сгенерированные числа: {6, 6, 1, 7, 5, 6}	№ Граница Количество 0 1 1 1 5 4 2 7 1	Верно
NumDatRun = 6 Xmin = -1 Xmax = 7	Число интервалов должно быть строго меньше Dx = Xmax - Xmin	Верно

Nint = 23		

Код программы см. в приложении А.

Выводы.

В ходе выполнения работы была изучена работа с модулями на языке ассемблер, изучен способ связи модулей на языке ассемблер с языком высокого уровня (C++). Была реализована программа, выполняющая распределение случайных чисел (с нормальным распределением) по заданным интервалам.

ПРИЛОЖЕНИЕ А

Lab_6.cpp:

```
#include <iostream>
#include <random>
#include <fstream>

int comp(const void* a, const void* b)
{
	return *(int*)a - *(int*)b;
}

class Generator {
	std::mt19937 generator;
	std::normal_distribution<double> distribution;
	double min;
	double max;
public:
	Generator(double min, double max) :
		distribution((min + max) / 2, (max - min) / 6), min(min), max(max)
	{}

	double operator ()() {
		while (true) {
			double number = this->distribution(generator);
			if (number >= this->min && number <= this->max)
				return number;
		}
	}
};
```

```
extern "C" void first(int* numbers, int numbers_size, int* result, int xmin);
extern "C" void second(int* array, int array_size, int xmin, int* intervals, int
intervals_size, int* result);

const int max_numbers_size = 16000;
const int max_interval_size = 24;

int main() {
	setlocale(LC_ALL, "ru");

	srand(time(NULL));
	std::ofstream file_result("result.txt");

	int numbers_size;
	int Xmin, Xmax;

	std::cout << "Введите количество чисел:" << std::endl;
	std::cin >> numbers_size;
	if (numbers_size > max_numbers_size) {
		std::cout << "Количество чисел должно быть меньше или равно
" <<  max_numbers_size << std::endl;
		return 0;
	}
	std::cout << "Введите Xmin и Xmax:" << std::endl;
	std::cin >> Xmin >> Xmax;
	int Dx = Xmax - Xmin;

	int intervals_size;
	std::cout << "Введите число границ:" << std::endl;
	std::cin >> intervals_size;
```

```
	if (intervals_size > max_interval_size) {
		std::cout << "Число интервалов должно быть меньше или равно " << max_interval_size << std::endl;
		return 0;
	}
	if (intervals_size >= Dx) {
		std::cout << "Число интервалов должно быть строго меньше Dx = Xmax - Xmin" << std::endl;
		return 0;
	}
	int* numbers = new int[numbers_size];
	int* intervals = new int[intervals_size];
	int* additional_intervals = new int[intervals_size];

	int len_asm_mod1_res = abs(Xmax - Xmin) + 1;
	int* mod_result = new int[len_asm_mod1_res];
	for (int i = 0; i < len_asm_mod1_res; i++)
		mod_result[i] = 0;


	int* final_result = new int[intervals_size + 1];
	for (int i = 0; i < intervals_size + 1; i++)
		final_result[i] = 0;


	std::cout << "Введите границы:" << std::endl;
	for (int i = 0; i < intervals_size; i++) {
		std::cin >> intervals[i];
		additional_intervals[i] = intervals[i];
	}
```

```
	std::qsort(intervals, intervals_size, sizeof(int), comp);
	std::qsort(additional_intervals, intervals_size, sizeof(int), comp);


	Generator genetate(Xmin, Xmax);

	for (int i = 0; i < numbers_size; i++) numbers[i] = genetate();

	std::cout << "Сгенерированные числа" << std::endl;
	file_result << "Сгенерированные числа" << std::endl;
	for (int i = 0; i < numbers_size; i++) {
		std::cout << numbers[i] << " ";
		file_result << numbers[i] << " ";
	}
	std::cout << std::endl;
	file_result << std::endl;


	first(numbers, numbers_size, mod_result, Xmin);
	second(mod_result, numbers_size, Xmin, intervals, intervals_size,
final_result);


	std::cout << "Результат:" << std::endl;
	file_result << "Результат:" << std::endl;
	std::cout << "№\tГраница\tКоличество чисел" << std::endl;
	file_result << "№\tГраница\tКоличество чисел" << std::endl;

	for (int i = 0; i < intervals_size; i++) {
			std::cout << i << "\t" << additional_intervals[i] << '\t' <<
final_result[i + 1] << std::endl;
```

```
                    file_result << i << "\t" << additional_intervals[i] << '\t' <<
final_result[i + 1] << std::endl;
        }

        delete[] numbers;
        delete[] intervals;
        delete[] additional_intervals;
        delete[] mod_result;
        delete[] final_result;

        file_result.close();

        return 0;
}
```

First.asm:

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C first
first PROC C numbers: dword, numbersSize: dword, res: dword, xmin: dword

push esi
push edi
mov edi, numbers
mov ecx, numbersSize
mov esi, res

for_:
        mov eax, [edi] ; помещаем в eax очередной элемент
        sub eax, xmin ; находим его индекс
```

```
        mov ebx, [esi + 4*eax]
        inc ebx ; увеличиваем значение по индексу на 1
        mov [esi + 4*eax], ebx ; помещаем в результ. массив
        add edi, 4 ; переходим к сл.элементу
        loop for_ ; пока ecx != 0

pop edi
pop esi
ret
first ENDP
END
```

Second.asm:

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C second
second PROC C mod_numbers: dword, numbersSize: dword, xmin: dword,
intervals: dword, intervalsSize: dword, result: dword

push esi
push edi
push ebp

mov edi, mod_numbers
mov esi, intervals
mov ecx, intervalsSize


for_intervals:  ; получение индексов от интервалов
        mov eax, [esi]
```

```
        sub eax, xmin
        mov [esi], eax
        add esi, 4
        loop for_intervals

mov esi, intervals
mov eax, [esi]
mov ecx, intervalsSize
mov ebx, 0 ; счетчик для result

for_loop:
        push ecx
        mov ecx, eax ; в eax - очередной интервал минус предыдущий
        push esi
        mov esi, result ; в esi записывается результат

    for_array:
                cmp ecx, 0
                je end_for
        mov eax, [edi]
        add [esi + 4 * ebx], eax
        add edi, 4
        loop for_array; пока интервал не равен 0 (т.е пока не пройдем весь
интервал)

end_for:
    inc ebx ; увеличиваем счетчик для result
    pop esi
        mov eax, [esi] ; запоминаем текущий интервал
        add esi, 4
```

```
        sub eax, [esi]
        neg eax  ; получаем следующий интервал минус предыдущий
        pop ecx
        loop for_loop

mov esi, result
mov ecx, intervalsSize
mov eax, 0 ; всего чисел в result

fin_for:
        add eax, [esi]
        add esi, 4
        loop fin_for

mov esi, result
sub eax, numbersSize
neg eax ; получаем число оставшихся необработанных чисел

add [esi + 4 * ebx], eax ; помещаем это число в result

pop ebp
pop edi
pop esi

ret
second ENDP
END
```