МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Организация ЭВМ и систем»

Тема: Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервал.

Студент гр.0382 _____ Диденко Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить способ организации связи ассемблера с ЯВУ.

Задание.

Вариант 4.

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

На языке C 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.

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.

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.

Для выполнения задания создано два модуля на языке ассемблера. Первый распределяет числа по единичным отрезкам (используя массив mod_result в качестве результата, по индексу которого находится количество сгенерированных чисел на данном единичном отрезке. 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.

Результаты тестирования программы lab6.cpp представлены в табл. 1.

Таблица 1 – Тестирование программы lab4.cpp.

№ п/п	Ввод	Вывод	Результат
1.	NumDatRun = 6 Xmin = 2 Xmax = 55 Nint = 3 LGrInt = {2,25,46} Сгенерированные числа: { 27 29 11 32 26 29}	№ Граница Кол-во чисел 0 2 1 1 25 5 2 46 0	Программа работает верно
2.	NumDatRun = 40 Xmin = -9 Xmax = 26 Nint = 5 LGrInt = { -9 14 18 20 24} Сгенерированные числа: { 7 9 -2 11 7 9 3 10 14 8 15 5 2 12 4 17 3 -6 3 5 14 4 5 10 7 20 6 6 -4	№ Граница Количество чисел 0 -9 31 1 14 8 2 18 0 3 20 1 4 24 0	Программа работает верно

	2 8 16 16 13 7 17 9 14 11 11}		

Выводы.

Изучен способ организации связи ассемблера с ЯВУ.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.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;
}
```

Название файла: 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
```

Название файла: 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
```