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 построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы

Студент гр. 1381 _____ Тарасов К.О.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы

Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы, связав модуль на Ассемблере с файлом на 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.

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.

Вариант работы — 1

Вид распределения — равномерный, число процедур — 1, Nint >= Dx, Lg1 > Xmin, ПГпосл > Xmax

Ход работы

Программа была реализована из 2-х модулей на C++ и 1 на ассемблере.

На ЯВУ написана программа, которая собирает информацию: количество чисел, интервал генерации псевдослучайных чисел, количество интервалов и сами интервалы, а также генерирует массив псевдослучайных чисел. Все данные передаются в ассемблерный модуль.

Ассемблерный модуль выполняет проверку принадлежности элемента массива array интервалу и если он принадлежит, то увеличивает соответствующий элемент массива result_array.

Связь между модулями осуществлена с помощью спецификатора extern, который позволяет выполнять раздельную компиляцию модулей.

Тестирование

Табл. 1. Результат тестирования.

Вызванные команды	Результат	Комментарий
NumDatRan=10 xmin = 0 xmax = 10 Nint = 10 LgrInt = 1 2 3 4 5 6 7 8 9 10 Generated numbers: 2 4 0 6 7 0 8 0 5 9	№ Граница Количество чисел 1 1 0 2 2 1 3 3 0 4 4 1 5 5 1 6 6 1 7 7 1 8 8 1 9 9 1 10 10 0	Верно
NumDatRan = 10 xmin = -6 xmax = 6 Nint = 3 LgrInt = -3 0 3 Generated numbers: 6 -4 5 -2 2 -5 3 1 -5 0	№ Граница Количество чисел 1 -3 1 2 0 3 3 3 3	Верно
NumDatRan = 12 xmin = -6 xmax = 6 Nint = 12 LgrInt = (-5 -4 -3 -2 -1 0 1 2 3 4 5 6) Generated numbers: -1 -2 3 -2 0 0 1 4 -6 -1 1 2	№ Граница Количество чисел 1 -5 0 2 -4 0 3 -3 0 4 -2 2 5 -1 2 6 0 2 7 1 2 8 2 1 9 3 1 10 4 1 11 5 0 12 6 0	Верно

Выводы:

В ходе выполнения работы была реализована программа частотного распределения случайных чисел по заданным интервалам на языке C++ с использование ассемблерных модулей

ПРИЛОЖЕНИЕ А

Текст компонентов программы lr6

main.cpp

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


extern "C" void FUNC(int* array, int array_size, int* left_boarders, int intervals_size, int*
result_array);

int main()
{
	int Xmin, Xmax, NumRanDat, interval_amount;
	int* left_borders;
	int* arr;
	int* result_array;
	std::random_device rand;
	std::mt19937 gen(rand());
	std::ofstream file("out.txt");

	std::cout << "Amount of numbers:" << std::endl;
	std::cin >> NumRanDat;

	std::cout << "Min: " << std::endl;
	std::cin >> Xmin;
	std::cout << "Max: " << std::endl;
	std::cin >> Xmax;

	while (Xmax < Xmin) {
		std::cout << "Wrong!" << std::endl;
		std::cout << "Min: " << std::endl;
		std::cin >> Xmin;
		std::cout << "Max" << std::endl;
```

```
        std::cin >> Xmax;
}

std::cout << "Amount of intervals: " << std::endl;
std::cin >> interval_amount;

while (interval_amount <= 0)
{
        std::cout << "Wrong!" << std::endl;
        std::cout << "Amount of intervals: " << std::endl;
        std::cin >> interval_amount;
}

left_borders = new int[interval_amount];
std::cout << "Enter left borders" << std::endl;
for (int i = 0; i < interval_amount; i++)
        std::cin >> left_borders[i];

for (int i = 0; i < interval_amount - 1; i++)
{
        for (int j = i + 1; j < interval_amount; j++)
        {
                if (left_borders[j] < left_borders[i])
                        std::swap(left_borders[j], left_borders[i]);
        }
}

std::uniform_int_distribution<> dis(Xmin, Xmax);
arr = new int[NumRanDat];

for (int i = 0; i < NumRanDat; i++)
        arr[i] = dis(gen);

std::cout << "Generated numbers: ";
for (int i = 0; i < NumRanDat; i++)
```

```
		std::cout << arr[i] << ' ';
	std::cout << '\n';

	file << "Generated numbers: ";
	for (int i = 0; i < NumRanDat; i++)
		file << arr[i] << ' ';
	file << '\n';

	result_array = new int[interval_amount];
	for (int i = 0; i < interval_amount; i++)
		result_array[i] = 0;

	FUNC(arr, NumRanDat, left_borders, interval_amount, result_array);

	std::cout << "Interval index \tInterval left border \tAmount of numbers in interval" << '\n';
	file << "Interval index \tInterval left border \tAmount of numbers in interval" << '\n';

	for (int i = 0; i < interval_amount; i++)
	{
		std::cout << "\t" << i + 1 << "\t\t" << left_borders[i] << "\t\t\t" << result_array[i] <<
'\n';
		file << "\t" << i + 1 << "\t\t" << left_borders[i] << "\t\t\t" << result_array[i] << '\n';
	}

	delete[] left_borders;
	delete[] arr;
	delete[] result_array;
	file.close();
	return 0;
}
```

mod.asm

```
.586
.MODEL FLAT, C
.CODE
```

```
FUNC PROC C array:dword, array_size:dword, left_borders:dword, interval_amount:dword,
result_array:dword
push ecx
push esi
push edi
push eax
push ebx

mov ecx, array_size
mov esi, array
mov edi, left_borders
mov eax, 0
l1:
        mov ebx, 0; обнуляем ebx для записи кол-ва интервалов
        borders:
                cmp ebx, interval_amount; Если ebx больше либо равно кол-ву интервалов
                jge borders_exit; То выходим
                push eax
                mov eax, [esi+4*eax]; Кладём в eax элемент из array
                cmp eax, [edi+4*ebx]; Сравниваем элемент массива и один из left_borders
                pop eax
                jl borders_exit; Если элемент меньше левой границы, то выходим
                inc ebx; Берём следующий интервал
                jmp borders
        borders_exit:
        dec ebx; Возвращаемся на предыдущий интервал

        cmp ebx, -1; Если ebx равен -1, то элемент не принадлежит не одному из интервалов
        je skip; Если ebx >= 0, то элемент принадлежит какому-либо интервалу
        mov edi, result_array; Соответствующий элемент результирующего массива
увеличиваем на 1
        push eax
        mov eax, [edi+4*ebx]
        inc eax
        mov [edi+4*ebx], eax
```

```
        pop eax
        mov edi, left_borders
        skip:
        inc eax; Следующий элемент массива
loop l1; Пока ecx(Размер массива) != 0

pop ebx
pop eax
pop edi
pop esi
pop ecx
ret
FUNC ENDP
END
```