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. 9382 _____ Дерюгин Д.А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2020

Цель работы.

Разработать программу, которая считает количество попаданий псевдослучайных чисел в заданные интервалы.

Задание.

Реализовать программу формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля (процедуры), сразу получающего требуемое распределение и возвращающего его в головную программу, написанную на ЯВУ.

Ход работы.

На языке высокого уровня программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне.

Далее вызывается процеду, которая формирует единичные интервалы.

Далее вызывается ассемблерная процедура для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы.

Результирующий массив частотного распределения чисел по интервалам, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле.

Тестирование.

Размер массива	Диапазон [X_min, X_max]	Массив	Левые границы	Количество значений
10	[-10; 10]	-8 -6 6 8 -4 10 0 8 1 4	-5 5	№1 2 №2 4 №3 4

10	[10, 100]	49 53 16 80 94 37 100 63 47 38	50	№1 5 №2 5
1	[1, 1]	1	1	№ 1 0 №2 1

Вывод.

В ходе выполнения данной лабораторной работы была написана программа на языке Ассемблера, которая строит частотное распределение попаданий псевдослучайных чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

Исходный код программы source.cpp

```
#include <iostream>

#include <time.h>

#include <fstream>

using namespace std;


extern "C" void asmfirst(int* array, int arrayLength, int* oncefrequency, int xMin);

extern "C" void asmsecond(int frequencyLength, int xMin, int* LGrInt, int countOfBorder,
int* frequency, int* result);


int main() {


	srand(time(nullptr));

	int  xMin, xMax, array_length, countOfBorders;
```

```
	cout << "enter array length\n";
	cin >> array_length;
	if (array_length <= 0 || array_length > 16 * 1024) {
		cout << "array length must be positive and less then 16 * 1024. Set by
default 10\n";
		array_length = 10;
	}

	cout << "Enter minimum and maximum numbers\n";
	cin >> xMin >> xMax;
	if (xMin > xMax) {
		cout << "Minimum number more then maximum. They will be swap\n";
		int tmp = xMin;
		xMin = xMax;
		xMax = tmp;
	}

	cout << "Enter count of borders\n";
	cin >> countOfBorders;
	if (countOfBorders <= 0 || countOfBorders > 24) {
		cout << "Count of borders must be in range (0;24). Set by defauld 5\n";
		countOfBorders = 5;
	}

	int* array = new int[array_length];
	int* LGrInt = new int[countOfBorders + 1];
	int* result = new int[countOfBorders + 1];

	cout << "Enter " << countOfBorders<< " intervals\n";
	for (int i = 0; i < countOfBorders; i++) {
			cin >> LGrInt[i];
			if (i != 0) {
				if (LGrInt[i] < LGrInt[i - 1]) {
```

```
                                cout << "Incorrect value\n";
                                return 0;
                        }
                }
                result[i] = 0;

        }
        result[countOfBorders] = 0;
        LGrInt[countOfBorders] = xMax + 1;

        int* oncefrequency = new int[xMax - xMin + 1];
        for (int i = 0; i < xMax - xMin + 1; i++) oncefrequency[i] = 0;

        for (int i = 0; i < array_length; i++) {
                cout << "random numbers:\n";
                array[i] = xMin + rand() % (xMax - xMin + 1);
        }

        for (int i = 0; i < array_length; i++) cout << array[i] << " ";
        cout << endl;

        asmfirst(array, array_length, oncefrequency, xMin);

        for (int i = 0; i < xMax - xMin + 1; i++) {
                if (oncefrequency[i] == 0) continue;
                cout << xMin + i<< " ";
                cout << oncefrequency[i] << endl;
        }

        asmsecond(xMax - xMin + 1, xMin, LGrInt, countOfBorders + 1,  oncefrequency,
result);
```

```
	ofstream fout;
	fout.open("file.txt");
	for (int i = 0; i < countOfBorders + 1; i++) {
		if (i == countOfBorders) fout <<"№"<< i + 1 << " " << xMin + LGrInt[i] -
1<< " " << result[i] << endl;
		else fout << "№"<< i + 1 << " " <<"	" << xMin + LGrInt[i] <<" " <<
result[i] << endl;
	}
	fout.close();

	return 0;
                                                }
```

Asm1.asm

```
.586
.model flat, C
.code
PUBLIC C asmfirst

asmfirst proc C array:dword, arrayLength:dword, oncefrequency:dword, xMin:dword
	mov edx, array
	mov ecx, arrayLength
	mov edi, oncefrequency

	loop_numbers:
		mov eax, [edx]
		add edx, 4
		sub eax, xMin
		mov ebx, [edi + eax*4]
		add ebx, 1
		mov [edi + eax*4], ebx
		loop loop_numbers

	ret
```

```
	asmfirst ENDP

	END
```

Asm2.asm

```
.586p

.MODEL FLAT, C

.CODE

PUBLIC C asmsecond

asmsecond PROC C  frequencyLength: dword, xMin: dword, LGrInt: dword, countOfBorder:
dword, frequency: dword, result: dword



mov esi, LGrInt; массив интервалов

mov ecx, countOfBorder; кол-во в массиве интервалов

mov edi, frequency; сами частоты



change_intervals:

	mov eax, [esi]

	sub eax, xMin

	mov [esi], eax

	add esi, 4

	loop change_intervals




mov esi, LGrInt

mov ecx, countOfBorder

mov eax, 0

mov ebx, [esi]

mov edx, 0;ndex of result



main:

	push ecx

	mov ecx, ebx

	push esi

	mov esi, result

	looping:
```

```
                cmp ecx, 0
                je next
                mov ebx, [edi]
                add [esi + eax * 4], ebx
                add edi, 4
                loop looping

next:
        pop esi
        mov ebx, [esi]
        add esi, 4
        sub ebx, [esi]
        neg ebx
        add eax, 1
        pop ecx
        loop main


ret
asmsecond ENDP
END
```