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.

Студентка гр. 0382 _____ Охотникова Г.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Целью данной работы является изучения связи ассемблера с высокоуровневым языком.

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, сформированный 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 .

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 12

№	Вид распредения	Число ассем. процедур	Nint ≥ D_x	Nint < D_x	Lgi ≤ X_{min}	Lg1 > X_{min}	ПГпосл ≤ X_{max}	ПГпосл > X_{max}
24	нормал.	2	-	+	+	-	+	-

Выполнение.

В файле main.cpp происходит считывание данных и проверка их на корректность. Затем происходит генерация чисел и вызов функций distribution_1 и distribution_2, описанных на языке Ассемблер в файлах module1.asm и module2.asm.

distribution_1:

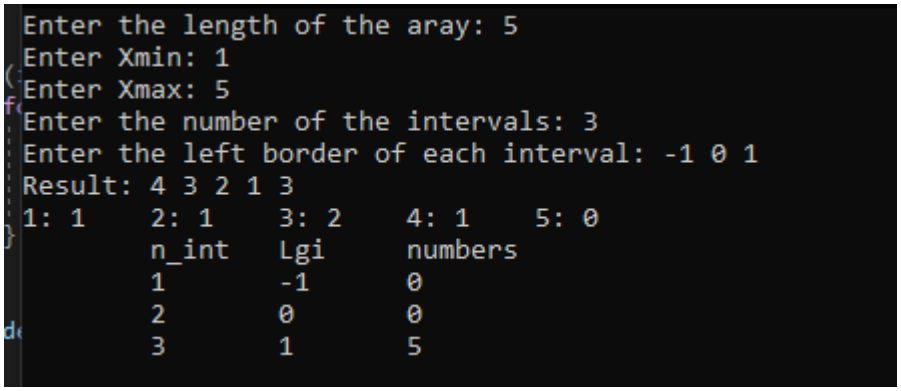
На вход принимает массив сгенерированных чисел arr, длину массива, массив result1, в который будут записаны результаты, и Xmin. Рассматривается каждое число массива arr, вычисляется, в какой единичный интервал оно входит, и по соответствующему индексу увеличивается значение в массиве result1 на 1.

distribution_2:

Так как правая граница каждого интервала меньше либо равна Xmin, рассматриваем две ситуации, то это первая ситуация. Вторая — рассмотрение последнего интервала: его левая граница меньше либо равна Xmin, а правая меньше либо равна Xmax. Тогда в первой ситуации количество чисел входящих в интервал равно количество Xmin, либо ноль, а во втором случае вычисляются все единичные интервалы, которые входят в текущий, и суммируются числа.

Исходный код программы см. в приложении А.

Тестирование.



Выводы.

В ходе работы была изучена связь Ассемблера с ЯВУ на примере построения программы частотного распределения. Была написана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: source.cpp

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

using namespace std;

extern "C" void distribution_1(int* arr, int len_arr, int* res, int
Xmin);
extern "C" void distribution_2(int* boarders, int Nint, int* units, int
Xmin, int Xmax, int* res);

int main() {

	int len_arr;
	cout << "Enter the length of the aray: ";
	cin >> len_arr;
	int Xmin, Xmax;
	cout << "Enter Xmin: ";
	cin >> Xmin;
	cout << "Enter Xmax: ";
	cin >> Xmax;
	if (Xmax < Xmin) {
		cout << "Xmax < Xmin";
		return 0;
	}
	int Nint;
	cout << "Enter the number of the intervals: ";
	cin >> Nint;
	if (Nint <= 0 || Nint > 24) {
		cout << "Nint must be > 0 and <= 24";
		return 0;
	}
	//cout << Xmax - Xmin << '\n';
	if ((Nint > Xmax - Xmin) || (Nint == Xmax - Xmin)) {
		cout << "Nint must be < Dx";
		return 0;
	}


	int* boarders = new int[Nint + 1];
	cout << "Enter the left border of each interval: ";
	cin >> boarders[0];
	for (int i = 1; i < Nint; i++) {
		cin >> boarders[i];
		if (boarders[i] > Xmax) {
			cout << "Boarders must be <= Xmax";
			return 0;
		}
	}
	if (boarders[0] > Xmin) {
		cout << "Lg1 mast be <= Xmin";
		return 0;
```

```
}

for (int i = 0; i < Nint - 1; i++) {
     for (int j = i + 1; j < Nint; j++) {
          if (boarders[j] < boarders[i]) {
               swap(boarders[j], boarders[i]);
          }
     }
}

boarders[Nint] = Xmax;

random_device rd;
mt19937 gen(rd());
double mean = (Xmax + Xmin) / 2;
double stddev = (Xmax - Xmin) / 4;
if (!stddev) {
     stddev = 1;
}
normal_distribution<double> dis(mean, stddev);

int* arr = new int[len_arr];
for (int i = 0; i < len_arr; i++) {
     arr[i] = dis(gen);
     while (arr[i] < Xmin || arr[i] > Xmax) {
          arr[i] = dis(gen);
     }
}

ofstream file("out.txt");
file << "Result: ";
for (int i = 0; i < len_arr; i++) {
     file << arr[i] << ' ';
}
file << '\n';

cout << "Result: ";
for (int i = 0; i < len_arr; i++) {
     cout << arr[i] << ' ';
}
cout << '\n';

int* result1 = new int[Xmax - Xmin + 1];
int* result2 = new int[Nint];
for (int i = 0; i < Xmax - Xmin + 1; ++i) {
     result1[i] = 0;
}
for (int i = 0; i < Nint; ++i) {
     result2[i] = 0;
}
distribution_1(arr, len_arr, result1, Xmin);
for (int i = 0; i < Xmax - Xmin + 1; i++) {
     cout << i + Xmin << ": " << result1[i] << '\t';
}
cout << '\n';
distribution_2(boarders, Nint, result1, Xmin, Xmax, result2);

cout << "\tn_int\tLgi\tnumbers" << '\n';
```

```
	file << "n_int\tLgi\tnumbers" << '\n';
	for (int i = 0; i < Nint; i++) {
		cout << "\t" << i + 1 << "\t" << boarders[i] << "\t" <<
result2[i] << '\n';
		file << "\t" << i + 1 << "\t" << boarders[i] << "\t" <<
result2[i] << '\n';
	}
	file.close();
	return 0;
}
```

Название файла: mod1.asm

```
	.586
	.MODEL FLAT, C
	.DATA

	.CODE

	PUBLIC C distribution_1
	distribution_1 PROC C arr: dword, len_arr: dword, res: dword, Xmin:
dword

	push esi
	push edi

	mov esi, arr
	mov edi, res
	mov ecx, len_arr
	mov edx, 0h

	start_loop:
		mov eax, [esi]
		sub eax, Xmin
		mov ebx, [edi + 4*eax]
		add ebx, 1
		mov [edi + 4*eax], ebx
		add esi, 4
		loop start_loop

	pop edi
	pop esi
```

```
ret
distribution_1 ENDP
END
```

Название файла: mod2.asm

```
.MODEL FLAT, C
.CODE

PUBLIC C distribution_2
distribution_2 PROC C boarders: dword, Nint: dword, result1: dword,
Xmin: dword, Xmax: dword, res: dword, flag: dword


mov esi, boarders
mov edi, res
mov ecx, Nint


start_loop:
     mov eax, [esi]
     mov ebx, [esi + 4]

     cmp eax, Xmin
     jge label1
     mov eax, 0
     sub ebx, Xmin
     cmp ebx, 0
     jle label2
     jmp label4

     label1:
          sub ebx, eax
          cmp ebx, 0
          je label2
          sub eax, Xmin

          label4:
```

```
            push esi
            push ecx

            mov esi, result1
            mov ecx, ebx
            mov ebx, 0

            loop_2:
                  add ebx, [esi+4*eax]
                  inc eax
                  loop loop_2

            pop ecx
            cmp ecx, 1
            jne label5
            add ebx, [esi + 4*eax]

            label5:
                  mov [edi], ebx
                  pop esi
                  jmp label3

      label2:
            mov ebx, 0
            mov [edi], ebx

      label3:
            add edi, 4
            add esi, 4
      loop start_loop



ret
distribution_2 ENDP
END
```