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

Кафедра Математического обеспечения электронно-вычислительных машин

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Организация ЭВМ и систем»

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

Вариант 13

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить основные принципы связи Ассемблера с ЯВУ. Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

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 .

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.

Задание, соответствующее варианту:

№	Вид распредения	Число ассем. процедур	Nint $\geq D_x$	Nint < D_x	Lgi $\leq X_{min}$	Lg1 > X_{min}	ПГпосл $\leq X_{max}$	ПГпосл > X_{max}
13	равном.	1	+	-	+	-	-	+

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.

Порядок выполнения работы.

Программа разработана с использованием языка программирования C++. В основном файле *lab6.cpp* организуем ввод исходных данных, удовлетворяющий требованиям поставленной задачи. Внутри будет осуществляться вызов ассемблерной функции для обработки введенных данных и подсчета попаданий сгенерированных чисел в заданные интервалы.

Ассемблерный модуль будет обрабатывать массив сгенерированных псевдослучайных чисел *numbers*. С помощью *loop*-цикла в блоке кода по метке

iter_numbers будут перебираться элементы этого массива. И для каждого элемента будет проверяться, принадлежит ли он интервалу, заданному левой границей из массива *borders,* которые так же будут перебираться в цикле по метке *check_interval:* пока граница не станет больше текущего элемента. Как только граница превысила элемент, значит, интервал, образованный предыдущей границей и текущей, «окутывает» элемент, а значит, элемент попадает в заданный интервал. Следовательно, можно обновлять счетчик попаданий для данного интервала, т.е. переходить по метке *update_counter,* где эти действия и осуществляются. С разработанным кодом можно ознакомиться в приложении. Код сопровождается комментариями.

Тестирование.

При запуске программа выводит и в консоль, и в файл верную результирующую таблицу, что говорит о корректности работы программы. Результаты тестирования см. на рисунке 1.

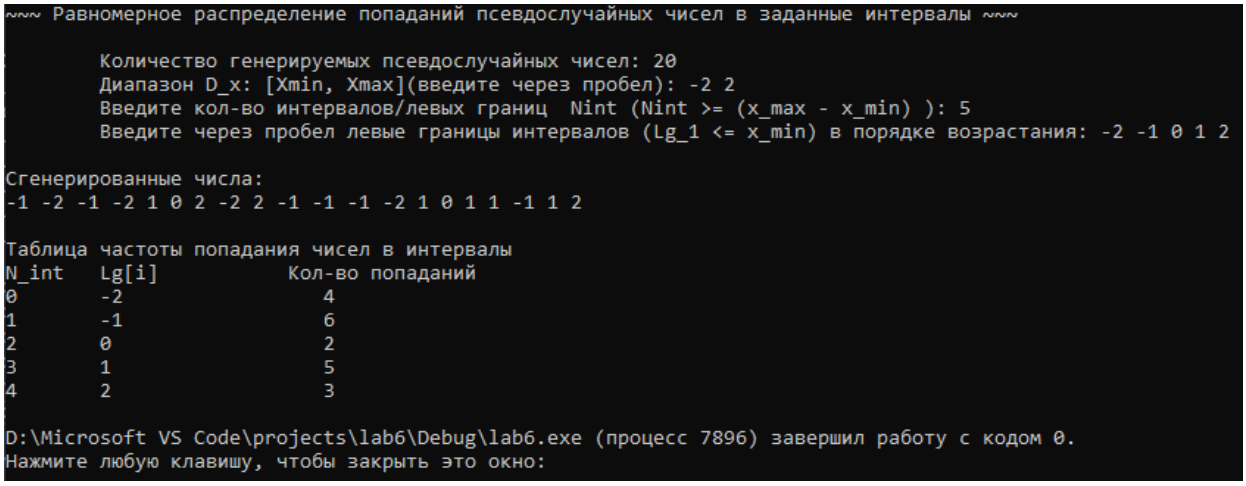

Рисунок 1 – Результаты тестирования

Вывод.

Были изучены принципы организации связи Ассемблера с ЯВУ, а также разработана программа, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lab6.cpp

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

extern "C" void count_hits_in_interval(int* numbers, int n, int*
borders, int N_int, int* counters);

int main()
{
	setlocale(LC_ALL, "Russian");
	int n, x_min, x_max, N_int;
	std::cout << "~~~ Равномерное распределение попаданий
псевдослучайных чисел в заданные интервалы ~~~\n\n";
	std::cout << "\tКоличество генерируемых псевдослучайных чисел: ";
	std::cin >> n;
	std::cout << "\tДиапазон D_x: [Xmin, Xmax](введите через пробел):
";
	std::cin >> x_min >> x_max;
	std::cout << "\tВведите кол-во интервалов/левых границ  Nint
(Nint >= (x_max - x_min) ): ";
	std::cin >> N_int;
	if (N_int < (x_max - x_min)) {
		std::cout << "\tДолжно выполняться условие Nint >= D_x !\n";
		return 0;
	}

	std::cout << "\tВведите через пробел левые границы интервалов
(Lg_1 <= x_min) в порядке возрастания: ";
	int* borders = new int[N_int+1]; // N_int+1, так как в последнюю
ячейку будем дублировать X_max
	for (int i = 0; i < N_int; i++) {
		std::cin >> borders[i];
	}
	if (borders[0] > x_min) {
		std::cout << "\tДолжно выполняться условие: Lg1 <=
X_min !\n";
		return 0;
	}
	borders[N_int] = x_max;

	std::random_device rnd;
	std::mt19937 gen(rnd());
	std::uniform_int_distribution<int>  distribution(x_min, x_max);

	std::cout << std::endl;
	//генерируем рандомные числа
	int* numbers = new int[n];
	std::cout << "Сгенерированные числа: \n";
	for (int i = 0; i < n; i++) {
		numbers[i] = distribution(gen);
```

```
            std::cout << numbers[i] << " ";
      }
      std::cout << std::endl;

      //подготовка к подсчету распределения и вызов самой ассемблерной
процедуры
      int* counters = new int[N_int];
      for (int i = 0; i < N_int; i++)
            counters[i] = 0;

      count_hits_in_interval(numbers, n, borders, N_int, counters);

      //запись результата
      std::ofstream fres("result_table.txt");
      fres << "Таблица частотного распределения чисел по интервалам\n"
<< "N_int\tLg[i]\t\tКол-во попаданий" << '\n';
      std::cout << "\nТаблица частоты попадания чисел в интервалы\n" <<
"N_int\tLg[i]\t\tКол-во попаданий" << '\n';
      for(int i = 0; i < N_int; i++){
            auto str_res = std::to_string(i) + "\t" +
std::to_string(borders[i]) + "\t\t   " + std::to_string(counters[i]) +
"\n";
            std::cout << str_res;
            fres << str_res;
      }
      return 0;
}
```

Название файла: module.asm

```
.MODEL FLAT, C

.CODE

PUBLIC C count_hits_in_interval

count_hits_in_interval PROC C  numbers: dword, n: dword, borders: dword, N_int: dword,  counters: dword

    ;remember changable registers

    push esi

    push edi

    push eax

    push ebx

    push ecx


    mov esi, numbers ; esi ~ for moving in 'numbers' array

    mov ecx, n       ; ecx ~ for loop-cycle in 'numbers' array

    mov edi, borders ; edi ~ for moving in 'borders' array

    mov eax, 0        ; eax ~ index for moving inside 'numbers' array

iter_numbers:

    mov ebx, 0      ; ebx ~ index of interval for moving inside 'borders' array


    check_interval:

        cmp ebx, N_int                  ; check if it's last "extra" interval [a,a] and Lgr=borders[N_int]("extra" x_max)

        je update_counter

        push eax                        ; remember eax ~ index of element from 'numbers'

        mov eax, [esi + 4 * eax]     ; put into eax an element from 'numbers'

        cmp eax, [edi + 4 * ebx]    ; cmp element from 'numbers' with Lgr from 'borders'
```

```
                pop eax                                 ; restore eax ~
index of element from 'numbers'

                jl update_counter               ; update counter if
element enter in previous interval

                inc ebx                                 ; increment an
index of interval

                jmp check_interval               ; go to check the next
interval


          update_counter:

                dec ebx                                 ; return to the
index of previous interval

                mov edi, counters               ; load to edi 'counters'
array's offset

                push eax                         ; remember eax ~ index
of element from 'numbers'

                mov eax, [edi + 4 * ebx]      ; put into eax current
counter for this interval

                inc eax                                 ; counter++

                mov [edi + 4 * ebx], eax   ; update the counter for this
interval

                pop eax                                 ; restore eax ~
index of element from 'numbers'

                mov edi, borders           ; return edi before next
iteration

                inc eax                                 ; increment an
index of element from 'numbers'


          loop iter_numbers                     ; go to count hits for
the next element from 'numbers'


          ;restore saved registers

          pop ecx

          pop ebx

          pop eax

          pop edi
```

```
    pop esi
ret
count_hits_in_interval ENDP
END
```