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 _____ Михайлов Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Формулировка задачи.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND_GEN (при его отсутствии программу датчика получить у преподавателя).

Далее должен вызываться ассемблерный модуль(модули) для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину.

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.

1. Длина массива псевдослучайных целых чисел - NumRanDat (<= 16K, K=1024)

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] , значения могут быть биполярные;

3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)

4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]).

Результаты.

1. Текстовый файл, строка которого содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк равно числу интервалов разбиения.

2. График, отражающий распределение чисел по интервалам (необязательный результат).

Для бригад с четным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами). Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла и, возможно, графика.

Код программы.

1.lab6_1.asm

```
.MODEL flat, C
.CODE
func1 PROC C random_numbers : ptr dword, one_step_intervals_dist : ptr dword,
number_of_random_numbers : dword, x_min : dword
; Данная функция строит единичное распределение.
mov ecx, 0; инициалиируем счетчик
mov esi, random_numbers; Устанавливаем указатель на входные данные
mov edi, one_step_intervals_dist; Устанавливаем указатель на выходные данные
start_func1:
; Начало цикла работы программы
mov eax, dword ptr [esi+ecx*4]; Считываем значение из входной строки
sub eax, x_min; вычисляем адрес, на котором находятся его смещения в массиве
inc dword ptr [edi+eax*4];
inc ecx;
cmp ecx, number_of_random_numbers;
jl start_func1;
ret
func1 ENDP
end
```

2. lab6_2.asm

```
.MODEL flat, C
.CODE
func2 PROC C one_step_intervals_dist : ptr dword, left_borders : ptr dword,
dist : ptr dword, x_min : dword, x_max : dword
```

```
mov eax, x_min; Инициализация переменных
mov ebx, 0;
mov ecx, 0;
mov edi, dist;
mov edx, left_borders;
mov esi, one_step_intervals_dist;
start_func2:
; Начало обработки
cmp eax, dword ptr [edx]; Проверяет, нужно ли перейти на другой интервал.
jge stuff;
mov ebx, dword ptr [esi+ecx*4];
add dword ptr [edi], ebx; Увеличиваем счетчик интервала.
; Увеличиваем переменные цикла
inc ecx;
inc eax;
cmp eax, x_max;
jle start_func2;
jmp to_end;
stuff:
add edi, 4; Меняет интервал
add edx, 4;
jmp start_func2; возвращается
to_end:
ret
func2 ENDP
end
```

3.Source.cpp

```
#include <ctime>
#include <iostream>
#include <windows.h>
#include <fstream>
#include <algorithm>

using namespace std;
ofstream output("output.txt", ios_base::app);

extern "C"
{
	void func1(int *, int*, int, int);
	void func2(int*, int*, int*, int, int);
}

int main()
{
	setlocale(LC_ALL, "Russian");
	srand(time(NULL));
	int random_amount = 0, random_numbers[16384], x_min = 0, x_max = 0, bor-
ders_amount = 0, left_borders[30];
	cout << "Введите минимальное число: " << endl;
	cin >> x_min;
	x_max = x_min - 1;
	while (x_max <= x_min) {
		cout << "Введите максимальное число: " << endl;
		cin >> x_max;
		if (x_max <= x_min)
			cout << "Максимальное значение не можеть быть меньше минималь-
ного!" << endl;
	}
	cout << "Вы задали интервал: [" << x_min << ", " << x_max << "]" << endl;
	cout << "Введите кол-во чисел, которое требуется сгенерировать(<= 16384):"
<< endl;
	while (random_amount > 16384 || random_amount < 1) {
		cin >> random_amount;
```

```
		if (random_amount < 1 || random_amount > 16384)
			cout << "Введеное количество чисел должно лежать в интервале от
1 до 16000! Повторите попытку:" << endl;
	}
	for (int i = 0; i < random_amount; i++)
		random_numbers[i] = rand() % (x_max - x_min + 1) + x_min;
	cout << "Введите кол-во интервалов, на которые разбивается диапазон измене-
ния массива псевдослучайных целых чисел [1, 24]:" << endl;
	while (borders_amount > 24 || borders_amount < 1) {
		cin >> borders_amount;
		if (borders_amount < 1 || borders_amount > 24)
			cout << "Введеное количество интервалов должно лежать в ин-
тервале [1; 24]! Повторите попытку:" << endl;
	}
	if (borders_amount >(x_max - x_min + 1)) {
		cout << "Ошибка.Количество интервалов не может быть больше количества
различных цифр" << endl;
		system("pause");
		exit(1);
	}
	for (int i = 0; i < borders_amount; i++)
		left_borders[i] = x_max + 1;
	if (borders_amount == 1)
		goto stop;
	cout << "Введите следующее количество границ интервалов - " <<
borders_amount - 1 << " ." << endl;

	for (int i = 0; i < borders_amount - 1; i++) {
		cout << "Введите " << i + 1 << "-ую границу: ";
		cin >> left_borders[i];
		if (left_borders[i] < x_min || left_borders[i] > x_max) {
			cout << "Граница должна быть > " << x_min + 1 << " и " << " < "
<< x_max - 1 << "!" << endl;
			i--;
		}
		if (left_borders[i] == x_min) {
			cout << "Крайнее левое значение отрезка уже является левой гра-
ницей!Повторите попытку:" << endl;
			i--;
		}
		for (int j = 0; j < i; ++j) {
			if (left_borders[i] == left_borders[j]) {
				cout << "Граница не должна повторяться! Повторите попыт-
ку:" << endl;
				i--;
			}
		}
	}
	sort(left_borders, left_borders + borders_amount - 1);
	cout << "Отсортированые значения левых границ: "; output << "Отсортирован-
ные значения левых границ: ";
	for (int i = 0; i < borders_amount - 1; i++) {
		cout << left_borders[i]; output << left_borders[i];
		if (i < borders_amount - 2) {
			cout << ", "; output << ", ";
		}
	}
	cout << "." << endl; output << "." << endl;
stop:
	int *one_step_intervals_dist = new int[x_max - x_min + 1]{};
	func1(random_numbers, one_step_intervals_dist, random_amount, x_min);
	cout << "Сгенерированная выборка: " << endl; output << "Сгенерированная вы-
борка: " << endl;
	for (int i = 0; i < random_amount; i++) {
		cout << random_numbers[i]; output << random_numbers[i];
```

```
            if (i < random_amount - 1) {
                  cout << ", "; output << ", ";
            }
      }
      cout << "." << endl; output << "." << endl;
      cout << "Значения еденичных распределений для чисел от минимума до мак-
симума: " << endl; output << "Значения единичных распределений для чисел от ми-
нимума до максимума: " << endl;
      for (int i = 0; i <= x_max - x_min; i++) {
            cout << i + x_min << " -> " << one_step_intervals_dist[i]; output <<
i + x_min << " -> " << one_step_intervals_dist[i];
            if (i < x_max - x_min) {
                  cout << ";" << endl; output << ";" << endl;
            }
      }
      cout << "." << endl; output << "." << endl;
      int *dist = new int[borders_amount] {};
      func2(one_step_intervals_dist, left_borders, dist, x_min, x_max);
      cout << "Значения распределений: "; output << "Значения распределений: ";
      for (int i = 0; i < borders_amount; i++) {
            cout << dist[i]; output << dist[i];
            if (i < borders_amount - 1) {
                  cout << ", "; output << ", ";
            }
      }
      cout << "." << endl; output << "." << endl;
      cout << endl << "***Таблица результатов***" << endl << "№ Левая граница
Количество чисел, попавших в интервал" << endl;
      cout << "1\t" << x_min << "\t\t\t" << dist[0] << endl;
      for (int i = 0; i < borders_amount - 1; i++) {
            cout << i + 2 << "\t" << left_borders[i] << "\t\t\t" << dist[i + 1]
<< endl;
      }
      delete[] dist; // очищаем память
      delete[] one_step_intervals_dist;
      system("PAUSE");
      return 0;
      }
```

Тестирование.

Размер массива псевдо-сл. чисел	Диапазон [Xmin,Xmax]	Массив распределения по интервалам единичной длины	Масси в ле-вых границ	Результирующая таблица частотного распределения
12	[-14, 0]	-14 -> 1; -13 -> 1; -12 -> 1; -11 -> 1; -10 -> 0; -9 -> 0; -8 -> 1; -7 -> 0; -6 -> 0; -5 -> 1; -4 -> 2; -3 -> 2; -2 -> 0; -1 -> 1;	1) -10 2) -7 3)-2	1 -14 4 2 -10 1 3 -7 5 4 -2 2

		0 -> 1.		
5	[-5, 12]	-5 -> 0; -4 -> 0; -3 -> 0; -2 -> 0; -1 -> 0; 0 -> 1; 1 -> 0; 2 -> 0; 3 -> 0; 4 -> 0; 5 -> 0; 6 -> 0; 7 -> 0; 8 -> 2; 9 -> 1; 10 -> 0; 11 -> 0; 12 -> 1.	1) -3 2) 0 3) 6 4) 9	1 -5 0 2 -3 0 3 0 1 4 6 2 5 9 2
1	[-2, 2]	-2 -> 0; -1 -> 0; 0 -> 0; 1 -> 1; 2 -> 0.	1) -2	1 -2 1

Вывод.

В ходе выполнения лабораторной работы была изучена организация связи ассемблерных модулей с ЯВУ на примере программы написанной на языке C++. Программа генерирует псевдослучайные числа и передает их в ассемблерные модули для их дальнейшей обработки. В итоге программа распределяет числа по отрезкам.