RND: Случайные числа

Код. Шаги 0-3

```
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include <math.h>
//////////////////////////////////////////////////////////////////////////////
/**
* Возвращает вещественное случайное число, равномерно распределённое
* на полуоткрытом интервале [a, b).
*
* @param a нижняя граница
* @param b верхняя граница
*
* @return Случайное число от a до b (не включая верхнюю границу).
*/
double Random(double a, double b) {
    return ((double)rand() / RAND_MAX) * (b - a) + a;
}

/**
* Заполняет массив равномерно распределёнными случайными числами.
*
* @param arr массив
* @param size количество элементов в массиве
* @param a нижняя граница
* @param b верхняя граница (не включается в интервал)
*
* @see Random()
*/
void FillRandom(double arr[], int size, double a, double b) {
    for(int i = 0; i < size; i++){
        arr[i] = Random(a, b);
    }
}

/**
* Печатает элементы массива через запятую в фигурных скобках.
*
* @param arr массив
* @param size количество элементов в массиве
*/
void Print(double const arr[], int size) {
    printf("{");

    for(int i = 0; i < size - 1; i++){
        printf("%lf, ", arr[i]);
    }
    printf("%lf", arr[size - 1]);

    printf("}\n");
}
//////////////////////////////////////////////////////////////////////////////
/**
* Строит гистограмму значений элементов массива.
```

```
*
* Заполняет массив counters[] на основании значений элементов массива arr[],
* подсчитывая, сколько их попало в соответствующий подынтервал
* полного интервала [a, b). Элементы массива, не попадающие в указанный
* интервал от a до b игнорируются.
*
* @param arr массив
* @param size количество элементов в массиве
* @param a нижняя граница
* @param b верхняя граница (не включается в интервал)
* @param counters заполняемый массив
* @param numCounters количество подынтервалов подсчёта
*/
void BuildHistogram(double const arr[], int size,
double left, double right, int counters[], int numCounters) {
    for(int i = 0; i < numCounters; i++){
        counters[i] = 0;
    }

    for(int i = 0; i < size; i++){
        int interval = (int)(numCounters * (arr[i] - left) / (right - left));
        counters[interval]++;
    }
}

/**
* Печатает элементы массива через запятую в фигурных скобках.
*
* @param counters массив
* @param numCounters количество элементов в массиве
*/
void PrintHistogram(int counters[], int numCounters) {
    printf("{");

    for(int i = 0; i < numCounters - 1; i++){
        printf("%d, ", counters[i]);
    }
    printf("%d", counters[numCounters - 1]);

    printf("}\n");
}
#define HIST_CHAR_BAR 'o'
#define HIST_CHAR_SPACE 183
/**
* Печатает горизонтальную гистограмму значений элементов массива.
*
* @param counters массив интервалов, содержащий количество элементов,
* попавших в соответствующий подынтервал
* @param numCounters количество подынтервалов подсчёта
* @param width полная ширина поля вывода гистограммы, в знакоместах
*/
void DrawHistogram(int counters[], int numCounters, int width) {
    int max_count = -1;
    for(int i = 0; i < numCounters; i++){
        if(counters[i] > max_count){
            max_count = counters[i];
        }
    }

    for(int i = 0; i < numCounters; i++){
        int n_symbols = (int)(width * counters[i] / max_count);
```

```
        int n_points = width - n_symbols;
        printf("%d ", i);
        for(int j = 0; j < n_symbols; j++){
            printf("o");
        }
        for(int j = 0; j < n_points; j++){
            printf(".");
        }

        printf("\n");
    }
}
////////////////////////////////////////////////////////////////////////////////
int main(void) {
    const int PRINT_MAX = 20;
    const double RANDOM_MIN = 0.3;
    const double RANDOM_MAX = 0.75;
    const double HIST_MIN = 0.0;
    const double HIST_MAX = 1.0;
    const int HIST_LINES = 5;
    const int HIST_LENGTH = 16;
    int size = 10;
    double *numbers = NULL;
    int *hist = NULL;
    setlocale(LC_CTYPE, "Russian");
    // TODO: 0 Взляните на случайные числа. Напечатайте несколько в цикле.
    // При желании можно рандомизировать ГПСЧ текущим числом секунд
    // srand((unsigned)time(0));
    // ...
    // Подготовьтесь к эксперименту
    printf("\nВведите количество чисел: ");
    scanf("%i", &size);
    // Выделите память
    numbers = malloc(size * sizeof(double));
    hist = malloc(HIST_LINES * sizeof(int));
    // Заполните массивы
    printf("\nРавномерная СВ (первые %i чисел):\n", PRINT_MAX);
    FillRandom(numbers, size, RANDOM_MIN, RANDOM_MAX);
    Print(numbers, size < PRINT_MAX ? size : PRINT_MAX);
    // Постройте гистограмму равномерной случайной величины
    BuildHistogram(numbers, size, HIST_MIN, HIST_MAX, hist, HIST_LINES);
    PrintHistogram(hist, HIST_LINES);
    DrawHistogram(hist, HIST_LINES, HIST_LENGTH);
    // Освободите память
    free(hist);
    free(numbers);
    return 0;
}
```

Пример работы программы.