Приложение 1. Тесты (использована генерация случайных значений матрицы)

Тест 1.
<<Input rows count:
>>3
<<Input columns count:
>>3
<<4 91 30
<<77 25 5
<<40 12 69
<< Matrix with left cyclic shift:
<<91 30 5
<<4 25 69
<<77 40 12

Тест 2.
<<Input rows count:
>>4
<<Input columns count:
>>4
>>54 69 39 45
>>76 49 44 9
>>66 83 79 52
>>14 27 99 11
>>Matrix with left cyclic shift:
>>69 39 45 9
>>54 44 79 52
>>76 49 83 11
>>66 14 27 99

Тест 3.
<<Input rows count:
>>3
<<Input columns count:
>>7
>>73 44 99 7 51 82 11
>>48 34 93 54 48 60 12
>>23 99 42 6 23 95 43
>>Matrix with left cyclic shift:
>>44 99 7 51 82 11 12
>>73 34 93 54 48 60 43
>>48 23 99 42 6 23 95

Тест 4.

```
<<Input rows count:
>>3
<<Input columns count:
>>7
>>64    87    18
>>74    24    45
>>64    61    55
>>68    94    56
>>35    16    81
>>Matrix with left cyclic shift:
>>87    18    45
>>64    24    55
>>74    61    56
>>64    94    81
>>68    35    16
```

Тест 5.

```
<<Input rows count:
>>1
<<Input columns count:
>>1
>>54
>>Matrix with left cyclic shift:
>>54
```

Приложение 1. Программа

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "Matrix.h"

void delay_ms(unsigned int milliseconds) {
   struct timespec req;
   req.tv_sec = milliseconds / 1000;
   req.tv_nsec = (milliseconds % 1000) * 1000000;
   nanosleep(&req, NULL);
}

void printMatrix(Matrix *matrix) {
  for(uint i = 0; i < matrix->getN(matrix); ++i) {
    for (uint j = 0; j < matrix->getM(matrix); ++j) {
      printf("%d\t", *Matrix_at(matrix, i, j));
    }
    printf("\n");
  }
}

void printWithTargetMatrix(Matrix *matrix, uint n, uint m) {
  printf("\e[1;1H\e[2J");
  for(uint i = 0; i < matrix->getN(matrix); ++i) {
    for (uint j = 0; j < matrix->getM(matrix); ++j) {
      if (i == n && j == m) {
        printf("\e[1;94;103m");
      }
      printf("%d\e[0;0m\t", *Matrix_at(matrix, i, j));
    }
    printf("\n");
  }
}

void printLine(int *line, uint len_line) {
  for(int i = 0; i < len_line; ++i){
    printf("%d ", line[i]);
  }
  printf("\n");
}

void linearizeMatrix(Matrix *matrix) {
  uint ind = 0;
  uint iters = Matrix_min_size(matrix);
  int cur_el = 0;

  uint len_line = matrix->getN(matrix) * matrix->getM(matrix);
  int line[len_line];

  for(int i = 0; i < len_line; ++i){
    line[i] = 0;
  }

  printf("\e[1;1H\e[2J");

  for(uint delta = 0; delta < iters; ++delta) {
    if (delta){
      for(uint j = 0; j < iters - delta; ++j) {
        printWithTargetMatrix(matrix, j, delta + j);
        cur_el = *Matrix_at(matrix, j, delta + j);
        line[ind] = cur_el;
        ++ind;
        printLine(line, len_line);
        delay_ms(50);
      }
    }
    for(int j = (iters - delta - 1); j > -1; --j) {
        printWithTargetMatrix(matrix, delta + j, j);
        cur_el = *Matrix_at(matrix, delta + j, j);
        line[ind] = cur_el;
```

```
      ++ind;
      printLine(line, len_line);
      delay_ms(50);
    }
  }
}

void outputFormatMatrix(uint n, uint m) {
  int base = 0;
  int el_num = 1;
  Matrix * matrix = Matrix_constructor(n, m, &base);
  uint iters = Matrix_min_size(matrix);
  printf("\e[2;33mCorrect output format:\e[0;0m\n");

  for(uint delta = 0; delta < iters; ++delta) {
    if (delta){
      for(uint j = 0; j < iters - delta; ++j) {
        *Matrix_at(matrix, j, delta + j) = el_num;
        ++el_num;
      }
    }
    for(int j = (iters - delta - 1); j > -1; --j) {
      *Matrix_at(matrix, delta + j, j) = el_num;
      ++el_num;
    }
  }
  printMatrix(matrix);
  Matrix_destructor(&matrix);
}

int main() {
  printf("\e[1;1H\e[2J");
  int base = 0;
  uint n, m;

  printf("\e[2;32mInput rows count:\e[0;0m\t");
  scanf("%d", &n);
  printf("\e[2;32mInput columns count:\e[0;0m\t");
  scanf("%d", &m);
  printf("\e[1;1H\e[2J");

  outputFormatMatrix(n, m);

  Matrix * matrix = Matrix_constructor(n, m, &base);
  Matrix_fill_random(matrix);
  linearizeMatrix(matrix);

  // printf("\nMatrix after diagonales replace:\n");
  // Matrix_diagonales_replace(matrix);
  // printMatrix(matrix);

  printf("\nMatrix with left cyclic shift:\n");
  Matrix * copy = Matrix_with_left_cyclic_shift_v2(matrix);
  printMatrix(copy);

  Matrix_destructor(&matrix);
  Matrix_destructor(&copy);
  return 0;
}
```