Приложение 1. Тесты

<<Input rows count:

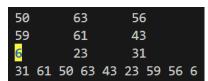
>>3

<<Input columns count:

>>3

>>50 63 56 59 61 43 6 23 31

<<31 61 50 63 43 23 59 56 6



<<Input rows count:

>>4

<<Input columns count:

>>4

>>6 91 38 20 69 73 83 62 55 61 69 55 94 11 1 15

<<15 69 73 6 91 83 55 1 61 69 38 62 11 55 20 94

```
6 91 38 20
69 73 83 62
55 61 69 55
94 11 1 15 15
15 69 73 6 91 83 55 1 61 69 38 62 11 55 20 94
```

## 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 len_line = matrix->getN(matrix) * matrix->getM(matrix);
 int line[len_line];
 for(int i = 0; i < len_line; ++i){
  line[i] = 0;
 }
 uint ind = 0;
 int cur_el = 0;
 printf("\e[1;1H\e[2J");
 for(uint delta = 0; delta < matrix->getN(matrix); ++delta) {
   for(uint j = 0; j < matrix->getN(matrix) - 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 = (matrix->getN(matrix) - 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;
 Matrix * matrix = Matrix_constructor(n, m, &base);
 int el_num = 1;
 printf("\e[2;33mCorrect output format:\e[0;0m\n");
 for(uint delta = 0; delta < matrix->getN(matrix); ++delta) {
  if (delta){
   for(uint j = 0; j < matrix->getN(matrix) - delta; ++j) {
    *Matrix_at(matrix, j, delta + j) = el_num;
    ++el_num;
   }
  }
  for(int j = (matrix - sgetN(matrix) - 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_from_console(matrix);
 linearizeMatrix(matrix);
```

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