Отчет по лабораторной работе №6 по курсу 1
Студент группы М80-111БВ-24, № по списку 15
Контакты e-mail: specapa@yandex.ru
Работа выполнена: «5» ноября 2024 г.
Преподаватель: каф. 806 Бучкин Т. А.
Входной контроль знаний с оценкой _ _ _
Отчет сдан «24» октября 2024 г., итоговая оценка _ _ _
Подпись преподавателя _ _ _

1. Тема: "Отчет по заданию курсового проекта №6 (14)".

2. Цель работы: освоение навыков работы с многомерными массивами, заголовочными файлами и библиотеками в языке Си.

3. Задание: линеаризовать квадратную матрицу по определенному правилу:

16

4	**5**	**11**	**15**
10	**3**	**6**	**12**
14	**9**	**2**	**7**
16	**13**	**8**	**1**

4. Оборудование ПЭВМ студента, если использовалось: 1,3 GHz 12-ядерный процессор Intel Core Ultra 5. Монитор: Универсальный монитор PnP.

5. Программное обеспечение ЭВМ студента, если использовалось: Операционная система семейства: Windows, наименование: Windows 11.
   Система программирования: нет.
   Редактор текстов: Notepad++.
   Компилятор: gcc.

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями):
Для линеаризации матрицы заметим следующее: количество шагов всегда нечётное, есть чередование направления движения, выше главное диагонали движение направлено вправо-вниз, ниже и включительно на главной диагонали влево-вверх.
Из всего этого вытекает следующий код:

```
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) {
    if (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);
    }
  }
}
```

На каждом шаге я печатаю матрицу с выделенным текущим элементом и линеаризованную матрицу, с написанными элементами до текущего включительно. Для человеческого восприятия между итерациями происходят паузы, что позволяет наблюдать “трасер” обхода. Подробную реализацию структуры Matrix можно увидеть в “Приложении 2”.

Оценка сложности алгоритма:
Общая сложность – O(n^2). Сложность алгоритма зависит от размеров исходной матрицы.
Сложность линеаризации – O(n). Сложность обусловлена обходом каждого элемента.
Сложность вывода матрицы – O(n). Выводим все элементы матрицы.

7. ~~Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].~~

8. Окончательное решение и тесты: “Приложение 1”

9. ~~Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.~~

10. Замечания автора по существу работы: Мной реализован ориентированный на зрительное восприятие интерфейс, позволяющий увидеть ход работы программы и шаги во время обхода матрицы.

11. Выводы: программа успешно написана и оттестирована.

Недочёты при выполнении задания могут быть устранены следующим образом: Недочётов нет.

Подпись студента: _____