Отчет по лабораторной работе №7 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 №7 (15)".

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

3. Задание: сделать циклический сдвиг (имеется ввиду перемещение элементов матрицы) влево всех элементов квадратной матрицы произвольного размера

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 циклического сдвига влево разработаем алгоритм: заметим, что все элементы изменяют только один из индексов (line or column), а конкретно делают это по правилу:

| ^	<-	<-	<-	<-
| ^	| ^	<-	<-	^ |
| ^	| ^	pass	^ |	^ |
| ^	->	->	^ |	^ |
->	->	->	->	^ |

Заметив закономерность в изменении, опишем её в коде. Главная итерация будет заключаться в постепенном схождении к центру матрицы, далее рассмотрение элементов в области:

```
Matrix * Matrix_with_left_cyclic_shift(Matrix * matrix) {
  Matrix * copy = Matrix_copy(matrix);
  uint lines_count = matrix->getN(matrix);
  uint columns_count = matrix->getM(matrix);
  uint iters;
  uint line;
  uint column;

  if (lines_count > columns_count) {
    iters = columns_count;
  } else {
    iters = lines_count;
  }

  for (uint step = 0; step < iters; ++step){
    for (line = step; line < (lines_count - step); ++line){
      for (column = step; column < (columns_count - step); ++column){
        if (line == step && column != (columns_count - step - 1)) {
          *Matrix_at(copy, line, column) = *Matrix_at(matrix, line, column + 1);
        } else if (line != (lines_count - step - 1) && column == (columns_count - step - 1)) {
          *Matrix_at(copy, line, column) = *Matrix_at(matrix, line + 1, column);
        } else if (line == (lines_count - step - 1) && column != step) {
          *Matrix_at(copy, line, column) = *Matrix_at(matrix, line, column - 1);
        } else if (line != step && column == step) {
          *Matrix_at(copy, line, column) = *Matrix_at(matrix, line - 1, column);
        }
      }
    }
  }
  return copy;
}
```

Но данный алгоритм не является оптимальным, так как при каждой итерации затрагиваются элементы, которые не входят в условия итерации. Именно поэтому была создана вторая версия алгоритма, которая для каждого “схождения” кольца. Также вторая версия алгоритма корректно работает с матрицами произвольного размера, а не только с квадратными. Итоговый алгоритм:

```
158  Matrix * Matrix_with_left_cyclic_shift_v2(Matrix * matrix) {
159    uint iters = Matrix_min_size(matrix) / 2;
160    uint lines_count = matrix->getN(matrix);
161    uint columns_count = matrix->getM(matrix);
162    T base_el = 0;
163    Matrix * copy = Matrix_constructor(lines_count, columns_count, &base_el);
164    uint line;
165    uint column;
166
167
168    for (uint step = 0; step < iters; ++step){
169      for (column = step; column < columns_count - 1 - step; ++column){
170        *Matrix_at(copy, step, column) = *Matrix_at(matrix, step, column + 1);
171        *Matrix_at(copy, lines_count - 1 - step, columns_count - 1 - column) = *Matrix_at(matrix, lines_count - 1 - step, columns_count - 1 - column - 1);
172      }
173      for (line = step; line < lines_count - 1 - step; ++line){
174        *Matrix_at(copy, line + 1, step) = *Matrix_at(matrix, line, step);
175        *Matrix_at(copy, line, columns_count - 1 - step) = *Matrix_at(matrix, line + 1, columns_count - 1 - step);
176      }
177    }
178
179    if ((lines_count > columns_count) && (columns_count % 2)){
180      for (line = columns_count / 2; line < lines_count - columns_count / 2; ++line){
181        *Matrix_at(copy, line, columns_count / 2) = *Matrix_at(matrix, line, columns_count / 2);
182      }
183    } else if ((lines_count < columns_count) && (lines_count % 2)){
184      for (column = lines_count / 2; column < columns_count - lines_count / 2; ++column){
185        *Matrix_at(copy, lines_count / 2, column) = *Matrix_at(matrix, lines_count / 2, column);
186      }
187    } else if ((lines_count == columns_count) && (lines_count % 2)){
188      *Matrix_at(copy, lines_count / 2, columns_count / 2) = *Matrix_at(matrix, lines_count / 2, columns_count / 2);
189    }
190
191    return copy;
192  }
193
```

Подробную реализацию структуры Matrix можно увидеть в “Приложении 2”.

Оценка сложности алгоритма:
Общая сложность первой версии – O(n^2). Неоптимизированный обход.
Общая сложность итогового алгоритма – O(n). Каждый элемент рассматривается единожды.

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: Изначальная версия алгоритма имеет изъян: рассматриваются не только элементы на рассматриваемой для каждой итерации “рамки”. Из-за чего сложность алгоритма увеличивается. В дальнейшем данный недостаток был исправлен, итоговая оптимизированная версия представлена в коде.
11. Выводы: программа успешно написана и оттестирована. В очередной раз удалось убедиться в том, что всегда нужно искать оптимальное решение.

Подпись студента: _____