Пермский национальный исследовательский политехнический университет

Кафедра информационные технологии и автоматизированные системы

Лабораторная работа

Реализация алгоритма для решения задачи о 8 ферзях

|  |  |
| --- | --- |
|  | Выполнил:  студент группы: ИВТ-24-2б  Ваулин Артём Сергеевич  Проверил:  доцент кафедры ИТАС  О.А. Полякова |

Пермь 2025

# Постановка задачи

**Задача:** Разработать программу на языке C++ для решения классической задачи о восьми ферзях. Необходимо разместить восемь ферзей на стандартной шахматной доске размером 8x8 таким образом, чтобы ни один ферзь не угрожал другому. Это означает, что никакие два ферзя не должны находиться на одной горизонтали, вертикали или диагонали.

**Входные данные:**

* Начальное положение первого ферзя. Пользователь должен ввести номер столбца (от 0 до 7) для размещения первого ферзя в первой строке (строка 0).

**Выходные данные:**

* В случае успешного нахождения решения, программа должна вывести на экран шахматную доску, где ферзи обозначены символом "Q", а пустые клетки символом ".". Также должен быть выведен текст "Решение найдено:".
* В процессе поиска решения программа должна пошагово отображать попытки размещения ферзей, выводя текущее состояние доски после каждой установки и отката ферзя. Должны быть выведены сообщения "Размещаем ферзя на позиции (строка, столбец):" и "Убираем ферзя с позиции (строка, столбец):" с соответствующими координатами.
* Если для заданного начального положения первого ферзя решение не найдено, программа должна вывести сообщение "Решение не найдено."

**Цель лабораторной работы:**

* Практическое применение алгоритма возврата (backtracking) для решения задачи с ограничениями.
* Реализация функций для проверки условий и пошагового поиска решения.
* Понимание принципов работы рекурсивных алгоритмов и их использования для перебора вариантов.
* Визуализация процесса решения задачи о 8 ферзях через пошаговый вывод.

# Анализ задачи

**Выбор алгоритма:**

Для решения задачи о 8 ферзях в данной лабораторной работе используется алгоритм **возврата (backtracking)**. Этот алгоритм хорошо подходит для задач поиска решений, где необходимо перебрать множество вариантов, отсекая заведомо неверные пути на ранних этапах.

**Обоснование выбора алгоритма:**

* **Систематический перебор:** Алгоритм возврата обеспечивает систематический перебор всех возможных вариантов размещения ферзей, гарантируя, что будет найдено решение (если оно существует) или доказано его отсутствие.
* **Эффективное отсечение:** Благодаря проверке условий безопасности (IsSafe функция), алгоритм отсекает ветви поиска, которые не могут привести к корректному решению, что значительно сокращает время работы по сравнению с полным перебором всех комбинаций.
* **Простота реализации:** Алгоритм возврата достаточно прост в реализации, особенно с использованием рекурсивных функций, что делает его подходящим для учебных целей.

**Структура программы и ключевые компоненты:**

1. **Представление шахматной доски:**
   * Используется двумерный массив Board[N][N] типа int, где N = 8.
   * Board[Row][Col] = 1 означает, что в клетке (Row, Col) находится ферзь.
   * Board[Row][Col] = 0 означает, что клетка (Row, Col) пуста.
2. **Функция IsSafe(int Row, int Col):**
   * Проверяет, безопасно ли размещать ферзя в позиции (Row, Col).
   * Проверяет наличие ферзей в предыдущих строках в том же столбце и на обеих диагоналях.
   * Возвращает true, если позиция безопасна, и false в противном случае.
3. **Функция SolveNQueens(int Row):**
   * Реализует рекурсивный алгоритм возврата.
   * **Базовый случай:** Если Row == N (все 8 строк заполнены), решение найдено, функция выводит доску и возвращает true.
   * **Рекурсивный шаг:**
     + Перебирает столбцы в текущей строке Row.
     + Для каждого столбца проверяет безопасность позиции с помощью IsSafe(Row, Col).
     + Если позиция безопасна:
       - Размещает ферзя на доске (Board[Row][Col] = 1).
       - Выводит сообщение о размещении и текущую доску.
       - Рекурсивно вызывает SolveNQueens(Row + 1) для следующей строки.
       - Если рекурсивный вызов возвращает true (решение найдено), текущая функция также возвращает true.
       - Если рекурсивный вызов возвращает false (решение не найдено в подзадаче), выполняется откат: ферзь убирается с доски (Board[Row][Col] = 0), выводится сообщение об откате и текущая доска, и алгоритм переходит к следующему столбцу в текущей строке.
   * Если перебраны все столбцы в текущей строке и решение не найдено, функция возвращает false.
4. **Функции InitializeBoard() и PrintBoard():**
   * InitializeBoard() инициализирует доску, заполняя все клетки нулями.
   * PrintBoard() выводит текущее состояние доски в консоль.
5. **Функция main():**
   * Запрашивает у пользователя начальный столбец для первого ферзя.
   * Проверяет корректность ввода пользователя.
   * Размещает первого ферзя в указанной позиции (если она безопасна, хотя для первой строки это всегда безопасно).
   * Вызывает SolveNQueens(1) для начала поиска решения со второй строки.
   * Выводит сообщение о результате поиска (решение найдено или нет).

**Пошаговый вывод:**

Программа обеспечивает пошаговый вывод процесса решения, что позволяет наглядно отслеживать работу алгоритма возврата. Вывод сообщений "Размещаем ферзя..." и "Убираем ферзя..." вместе с визуализацией доски на каждом шаге помогает понять, как алгоритм исследует пространство решений и выполняет откат при обнаружении тупиковых ситуаций.

# Блок-схема

# 

# Код программы

#include <iostream>

#include <limits>

using namespace std;

const int N = 8;

int Board[N][N];

// Инициализация доски

void InitializeBoard()

{

for (int Row = 0; Row < N; ++Row)

{

for (int Col = 0; Col < N; ++Col)

{

Board[Row][Col] = 0;

}

}

}

// Вывод доски

void PrintBoard()

{

for (int Row = 0; Row < N; ++Row)

{

for (int Col = 0; Col < N; ++Col)

{

cout << (Board[Row][Col] ? "Q " : ". ");

}

cout << endl;

}

cout << endl;

}

// Проверка безопасности позиции (Row, Col)

bool IsSafe(int Row, int Col)

{

for (int i = 0; i < Row; ++i)

{

if (Board[i][Col])

{

return false; // Проверка столбца

}

if (Col - (Row - i) >= 0 && Board[i][Col - (Row - i)])

{

return false; // Проверка диагонали (\)

}

if (Col + (Row - i) < N && Board[i][Col + (Row - i)])

{

return false; // Проверка диагонали (/)

}

}

return true;

}

// Решение задачи о N ферзях с начальным положением

bool SolveNQueens(int Row)

{

if (Row == N)

{

cout << "Решение найдено:" << endl;

PrintBoard();

return true;

}

for (int Col = 0; Col < N; ++Col)

{

if (IsSafe(Row, Col))

{

Board[Row][Col] = 1;

cout << "Размещаем ферзя на позиции (" << Row << ", " << Col << "):" << endl;

PrintBoard();

if (SolveNQueens(Row + 1))

{

return true;

}

Board[Row][Col] = 0; // Откат

cout << "Убираем ферзя с позиции (" << Row << ", " << Col << "):" << endl;

PrintBoard();

}

}

return false;

}

int main()

{

setlocale(LC\_ALL, "Rus");

InitializeBoard();

int initialCol;

cout << "Введите столбец (0-" << N - 1 << ") для первого ферзя: ";

cin >> initialCol;

if (cin.fail() || initialCol < 0 || initialCol >= N)

{

cout << "Неверный ввод." << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return 1;

}

if (IsSafe(0, initialCol))

{

Board[0][initialCol] = 1;

cout << "Начальное положение ферзя на позиции (0, " << initialCol << "):" << endl;

PrintBoard();

if (!SolveNQueens(1))

{

cout << "Решение не найдено." << endl;

}

}

else

{

cout << "Начальное положение небезопасно (невозможно в первой строке)." << endl;

}

return 0;

}

# Результат работы

