**项目说明文档**

**数据结构课程设计**

**——N皇后问题**

作 者 姓 名 刘淑仪

学 号 2251730

指 导 教 师 张 颖

学 院 专 业 软件学院 软件工程



二〇二三 年 十 月 二十五 日

目录

1 项目分析 1

1.1 项目背景分析 1

1.2 项目功能分析 1

2 项目设计 2

2.1 数据结构设计 2

2.2 类结构设计 2

2.3 成员与操作设计 3

2.4 系统设计 3

3 功能设计与项目实现 4

3.1 输入皇后个数功能的实现 4

3.1.1 输入皇后个数功能实现思路 4

3.1.2 输入皇后个数功能核心代码 4

3.1.3 输入皇后个数功能实现示例 5

3.2 N皇后问题的解决 5

3.2.1 N皇后问题解决实现思路 5

3.2.2 N皇后问题解决核心代码 5

3.2.3 N皇后问题解决实现示例 7

4 项目测试 7

4.1 功能测试 7

4.1.1 基本功能测试 7

4.1.2 边界条件测试 8

4.2 错误测试 10

4.2.1 输入验证错误 10

4.2.1.1 输入验证错误判断思路 10

4.2.1.3输入示例 10

4.2.2 异常情况处理 11

4.3 Linux环境测试 12

5 集成开发环境与编译运行环境 12

# 项目分析

## 项目背景分析

八皇后问题源于一个历史悠久的数学挑战，由19世纪的数学家高斯在1850年提出。这个问题要求在8x8的国际象棋棋盘上放置8个皇后，使得它们中的任何一个都不能攻击到其他皇后。在象棋规则中，皇后能攻击到同一行、列或对角线上的任何棋子，因此这个问题变成了确保所有皇后都不共行、列或对角线的挑战。这个问题不仅仅是一个简单的棋盘游戏问题，它在数学和算法设计领域有着重要意义。最初，高斯认为有76种不同的解决方案，但后续研究发现，实际上有92种不同的解法。这个问题的研究推动了图论和算法设计领域的发展，尤其是在探索解决方案时所使用的回溯算法。

解决 N 皇后问题的核心在于高效的回溯算法。这个算法需要不断尝试不同的皇后放置方式，一旦当前方式导致冲突，算法就需要回溯并尝试其他可能的解决方案。关键在于如何高效地检测冲突以及如何智能地减少不必要的检测，这对于提高算法效率至关重要。

## 项目功能分析

1. 回溯算法

回溯算法是解决N皇后问题的核心。它尝试在棋盘的每一行放置一个皇后，并且保证这个皇后不会被其他已放置的皇后攻击。如果当前行没有安全的列可放置皇后，算法会回溯到上一行，移动那一行的皇后到不同的列。

1. 冲突检测

此功能用于检查在棋盘上放置一个新的皇后时是否会导致冲突。冲突发生在当两个皇后处于同一行、同一列或同一对角线。

1. 输入输出处理

项目允许用户输入皇后的数量，从而动态决定棋盘的大小和问题的复杂性。当找到一种满足条件的皇后布局时，程序输出该解决方案的具体布局。

# 项目设计

## 数据结构设计

在 N 皇后问题的项目中，我们采用了二维数组 board[MAXSIZE][MAXSIZE] 来表达棋盘的布局，其中 MAXSIZE 指定了棋盘的最大尺寸。这个二维数组天生适合于描绘棋盘的结构，每个数组元素代表棋盘上的一个单元格，可标识为皇后（QUEEN）或空格（EMPTY）。这样的数据结构便于在棋盘的任意位置添加或移动皇后，从而有效地实现了N皇后布局的生成和求解算法。

为了记录棋盘上每个单元格的安全状态，我们使用了布尔型数组 isSafe[MAXSIZE][MAXSIZE]。这种结构使程序能够迅速检查任意给定位置是否适合放置皇后，这在保证皇后互不攻击的规则下寻找解决方案时至关重要。

在这个项目中，棋盘上不同的元素（如皇后、空格）代表了不同的状态和含义。使用枚举类型 ChessPiece 来区分这些不同的元素，不仅使代码更加易于理解和维护，同时在功能上提供了更强的类型检查，降低了出错的可能性。

项目中的每一个棋盘位置都可以用一个坐标来表示。定义了结构体 Position 来存储这些坐标。这种方法不仅使代码更加清晰有序，同时也为处理棋盘上的位置信息提供了一种灵活且精确的手段。

## 类结构设计

NQueen 类的设计恰当地反映了解决 N 皇后问题的教育和实践目的，同时展示了计算机算法在解决经典问题中的应用。该类通过模块化的方法和有效的数据结构实现了用户友好的交互界面和 N 皇后问题的求解功能。这种设计不仅使项目易于使用和理解，而且在代码的可维护性、扩展性和效率方面表现出色，为解决复杂的逻辑和算法问题提供了坚实的基础。

NQueen 类结构设计：用于存储棋盘状态和实现其核心功能。

在主要的 cpp 文件（假设为 MainNQueenGame.cpp）中：

创建 NQueenGame 类（如果需要）：用于存储 NQueen 实例和游戏相关的参数作为私有成员，例如棋盘的尺寸或其他设置。公有成员包括游戏流程控制函数：用于初始化游戏、开始游戏循环、处理用户输入和展示游戏结果。

NQueen 类中定义的方法和变量，如 solveNQueens 和 isSafe，以及棋盘数组 board，都直接关联到 N 皇后问题的求解。这些方法和变量提供了项目的核心功能，如回溯算法的实现、冲突检测和用户交互处理。

通过清晰的类结构，NQueen 类不仅在解决 N 皇后问题方面提供了高效的机制，同时也为学习和理解复杂算法提供了一个实用的平台。

## 成员与操作设计

class NQueen {

private:

bool board[MAXSIZE][MAXSIZE] = { 0 };

int count = 0;

int N = 0;

public:

void menu();

void input();

void solveNQueens(int row);

bool isSafe(int row, int col);

void Exit() {

std::cout << "按任意键继续...\n";

// 清除输入缓冲区

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

// 等待用户按键

std::cin.get();

}

};

## 系统设计

模块化：代码被组织成不同的函数，每个函数负责特定的任务，如初始化、生成棋盘布局并求解问题、检查皇后的放置是否会导致冲突，使得代码易于理解和维护。

算法驱动：N 皇后问题的解决完全依赖于实现的算法，特别是回溯算法的应用。

用户友好性：虽然项目的用户界面相对简单，但它提供了充分的信息和反馈，使用户可以轻松地与系统交互。通过明确的提示和指导，用户可以轻松输入皇后的数量并观察到不同解决方案的输出。

可扩展性：系统设计允许在未来对迷宫的功能进行扩展，如更高级的算法优化、用户界面改进或其他用户交互特性。

# 功能设计与项目实现

## 输入皇后个数功能的实现

### 输入皇后个数功能实现思路

该函数通过一个循环结构，函数不断请求用户输入，直至接收到一个有效且合理的数值。它细致地检查输入的有效性，处理了各种可能的错误输入情况，例如非数字输入和超出预定范围的数值。若输入无效，函数会清除错误状态，忽略非法输入，并指引用户重新输入，直到接收到合法的数据。

### 输入皇后个数功能核心代码

void NQueen::input()

{

while (1) {

std::cin >> N;

if (std::cin.fail() || N <= 3 || N > MAXSIZE) {

std::cin.clear();

std::cin.ignore(65536, '\n');

std::cout << "输入数据有误，请重新输入\n";

}

else {

if (std::cin.get() == '\n')

break;

else {

std::cin.clear();

std::cin.ignore(65536, '\n');

std::cout << "输入数据有误，请重新输入\n";

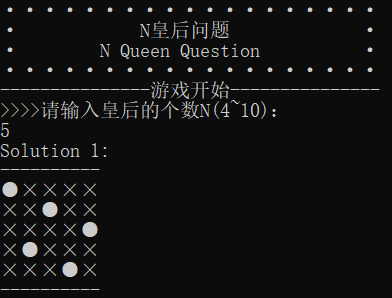
}

}

}

}

### 输入皇后个数功能实现示例



## N皇后问题的解决

### N皇后问题解决实现思路

isSafe 函数的目的是检查在给定的 row 和 col（行和列）位置上放置皇后是否安全。函数遍历当前列的所有行（从0到row-1），如果发现任何行已经有皇后（board[i][col] == 1），则返回 false，表示放置不安全。从当前位置向左上方向遍历，如果在对角线上发现皇后，则同样返回 false。从当前位置向右上方向遍历，同样检查对角线上是否有皇后。如果上述检查都通过，返回 true，表示在该位置放置皇后是安全的。

solveNQueens 函数采用回溯算法解决 N 皇后问题。如果 row 等于 N，表示所有皇后都已安全放置，这时打印当前的棋盘布局作为一个解决方案，并返回。对于棋盘的每一行，遍历所有列。对于每个单元格（row, col），使用 isSafe 函数检查是否可以安全放置皇后。如果在某个位置放置皇后是安全的，将该位置标记为有皇后（board[row][col] = 1），然后递归地调用 solveNQueens 函数处理下一行。在回溯步骤中，将当前位置的皇后移除（board[row][col] = 0），以便尝试在当前行的其他列放置皇后。

### N皇后问题解决核心代码

bool NQueen::isSafe(int row, int col) {

// 检查同一列

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

if (board[i][col] == 1)

return false;

// 检查左上对角线

for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j] == 1)

return false;

// 检查右上对角线

for (int i = row, j = col; i >= 0 && j < N; i--, j++)

if (board[i][j] == 1)

return false;

return true;

}

void NQueen::solveNQueens(int row) {

if (row == N) {

// 找到一个解，打印它

count++;

std::cout << "Solution " << count << ":\n";

for (int i = 0; i < 2 \* N; i++)

std::cout << '-';

std::cout << '\n';

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

std::cout << (board[i][j] == 1 ? "●" : "×");

}

std::cout << std::endl;

}

for (int i = 0; i < 2 \* N; i++)

std::cout << '-';

std::cout << '\n';

return;

}

for (int col = 0; col < N; col++) {

if (isSafe(row, col)) {

board[row][col] = 1;

solveNQueens(row + 1);

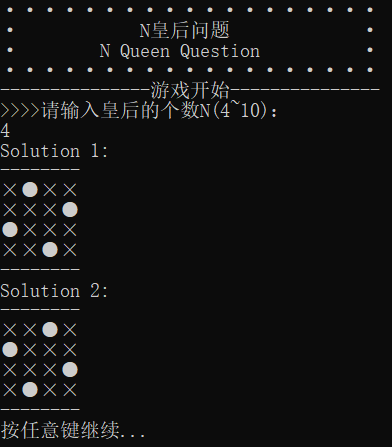
board[row][col] = 0; // 回溯

}

}

}

### N皇后问题解决实现示例



# 项目测试

## 功能测试

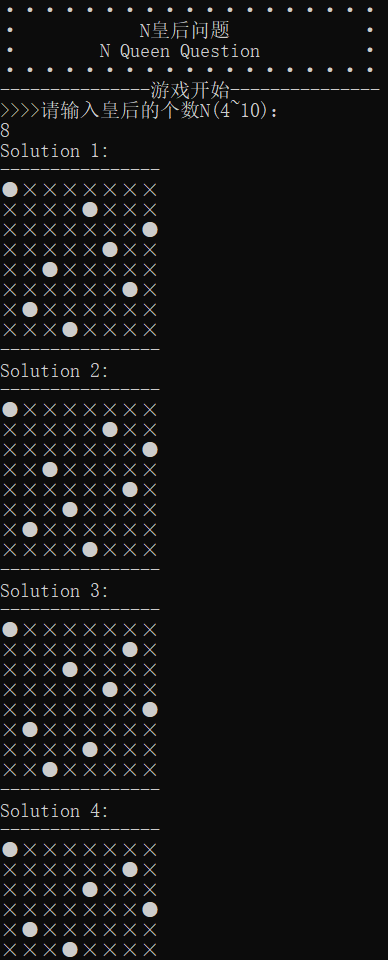
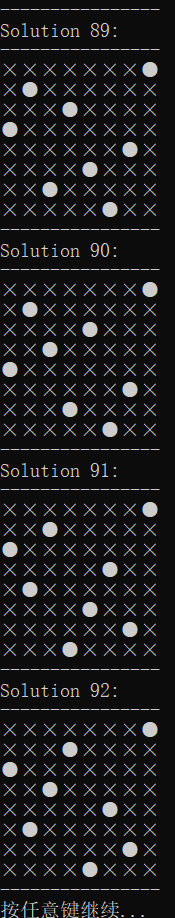
### 基本功能测试

**测试用例：**8

**用例说明：**标准输入

**预期结果：**应显示92种可能的解决方案，这是八皇后问题的已知解。

**实验结果：**

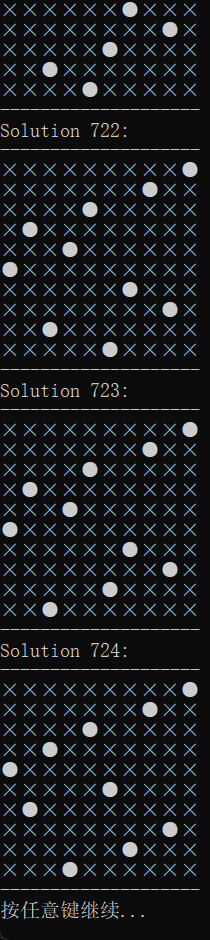
### 4.1.2 边界条件测试

**测试用例：**10

**用例说明：**进行边界测试.

**预期结果：**程序应该在极端情况下也能正确运行，不崩溃。应显示所有10皇后布局的解决方案。虽然数量可能很大，但程序应能正确处理。

**实验结果：**



**[注：由于屏幕缓冲区的问题，截图没有显示完整，但能够正确生成所有布局]**

## 错误测试

### 输入验证错误

#### 4.2.1.1 输入验证错误判断思路

输入非法值（如负数、非数字字符、超出范围的数字）来测试程序对错误输入的处理能力。检查程序是否能妥善处理并提示用户重新输入。

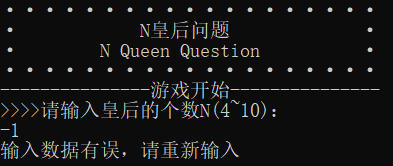
#### 4.2.1.3输入示例

**测试用例：**-1

**用例说明：**测试非正数

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

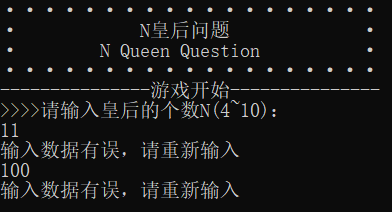


**测试用例：**11/100

**用例说明：**测试超过最大数的数

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

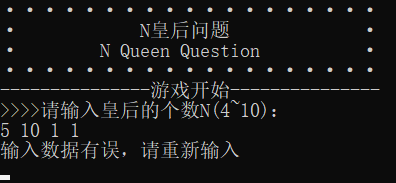


**测试用例：**5 10 1 1；

**用例说明：**测试多个输入

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

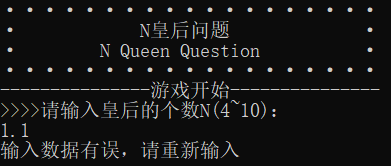


**测试用例：**1.1；

**用例说明：**测试小数；

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



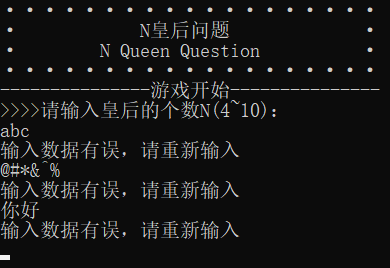
### 异常情况处理

**测试用例：**abc/ @#￥ / 五

**用例说明：**测试非法输入（如字母）

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

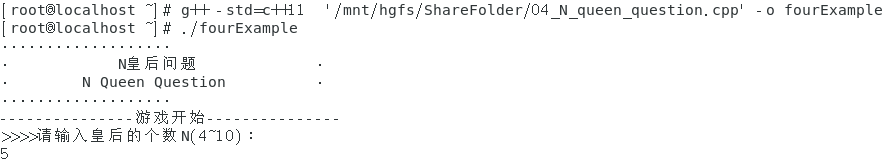


## Linux环境测试

**编译命令：** g++ -std=c++11 '/mnt/hgfs/ShareFolder/ 04\_N\_queen\_question' -o fourExample

**运行命令：** ./fourExample

**实验结果：**



1. **集成开发环境与编译运行环境**

Windows系统：Windows 11 x64

Windows集成开发环境：Microsoft Visual Studio 2022 (Release模式)

Windows编译运行环境：本项目适用于x86架构和x64架构

Linux系统：CentOS 7 x64