



东北大学秦皇岛分校
计算机与通信工程学院
数据结构课程设计

设计题目 八皇后问题

专业名称 计算机科学与技术

班级学号 160508

学生姓名 邢鼎威

指导教师 王聪

设计时间 2017 年 12 月 21 日— 2018 年 1
月 4 日

课程设计任务书

专业：计算机科学与技术 学号：20167482 学生姓名（签名）：

设计题目：八皇后问题

一、设计实验条件

工学馆

二、设计任务及要求

1. 求出在一个 $n \times n$ 的棋盘上，放置 n 个不能互相捕捉的国际象棋“皇后”的所有布局。
2. 皇后可以沿着纵横和两条斜线 8 个方向相互捕捉。如图所示，一个皇后放在棋盘的 第 4 行第 3 列位置上，则棋盘上凡打“×”的位置上的皇后就能与这个皇后相互捕捉，也就是下一个皇后不能放的位置。
3. 皇后可以沿着纵横和两条斜线 8 个方向相互捕捉。如图所示，一个皇后放在棋盘的 第 4 行第 3 列位置上，则棋盘上凡打“×”的位置上的皇后就能与这个皇后相互捕捉，也就是下一个皇后不能放的位置。

三、设计报告的内容

1. 设计题目与设计任务（设计任务书）

设计题目：八皇后问题

设计任务书：

求出在一个 $n \times n$ 的棋盘上，放置 n 个不能互相捕捉的国际象棋“皇后”的所有布局。

这是来源于国际象棋的一个问题。皇后可以沿着纵横和两条斜线 8 个方向相互捕捉。如图所示，一个皇后放在棋盘的 第 4 行第 3 列位置上，则棋盘上凡打“×”的位置上的皇后就能与这个皇后相互捕捉，也就是下一个皇后不能放的位置。

表 1 八皇后问题样例表

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

		×			×		
×		×		×			
	×	×	×				
×	×	Q	×	×	×	×	×
	×	×	×				
×		×		×			
		×			×		
		×				×	

从图中可以得到以下启示：一个合适的解应是在每列、每行上只有一个皇后，且一条斜线上也只有一个皇后。

2. 前言（绪论）(设计的目的、意义等)

绪论：

本课程设计是以八皇后问题为主，对八皇后问题进行求解，求解过程中使用回溯法进行程序的设计，在程序设计过程中，用到了 dfs 等具有回溯思想的方法，加深了我对于回溯思想的理解和印象。

3. 设计主体（各部分设计内容、分析、结论等）

3.1 需求分析

程序设计的任务：

本程序设计是在使用回溯法对于八皇后问题进行求解的基础上，拓展至 n 皇后问题，从而能够用回溯法对于 n 皇后问题进行求解，并且通过 Qt 的设计，能够模拟出求解 n 皇后的过程，从而能够更加清晰地理解回溯法的原理。

功能模块：

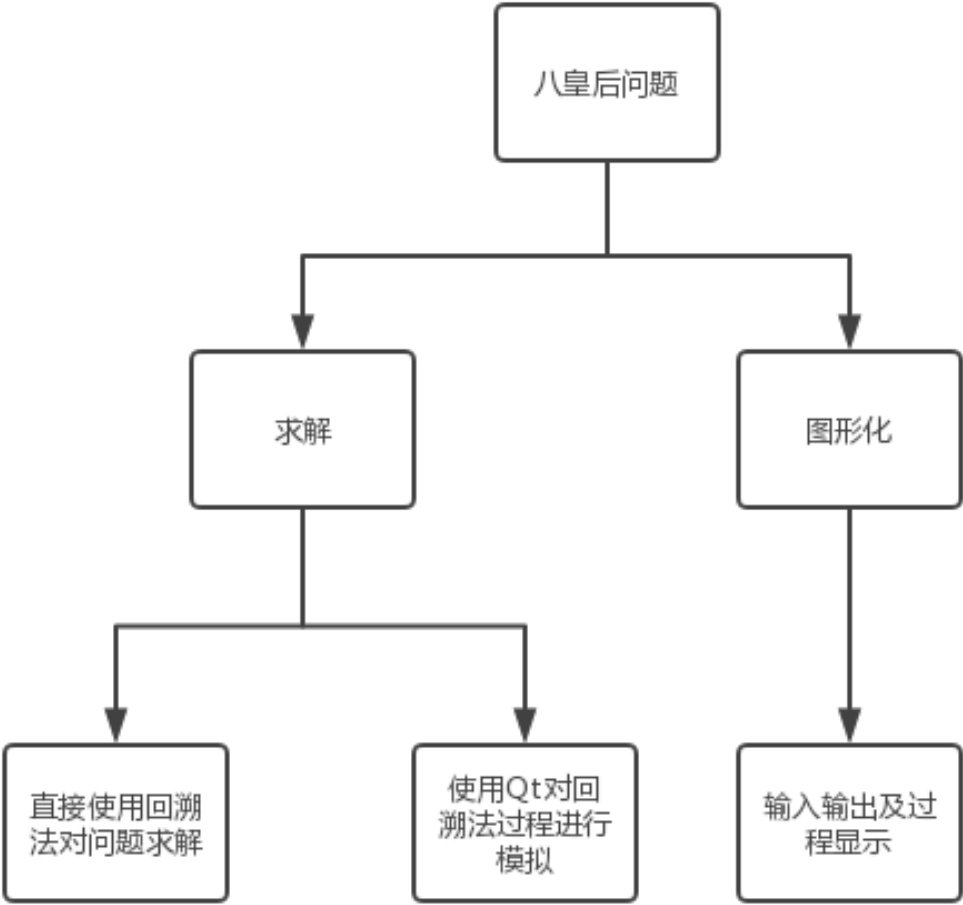


图 1 功能模块

流程图：

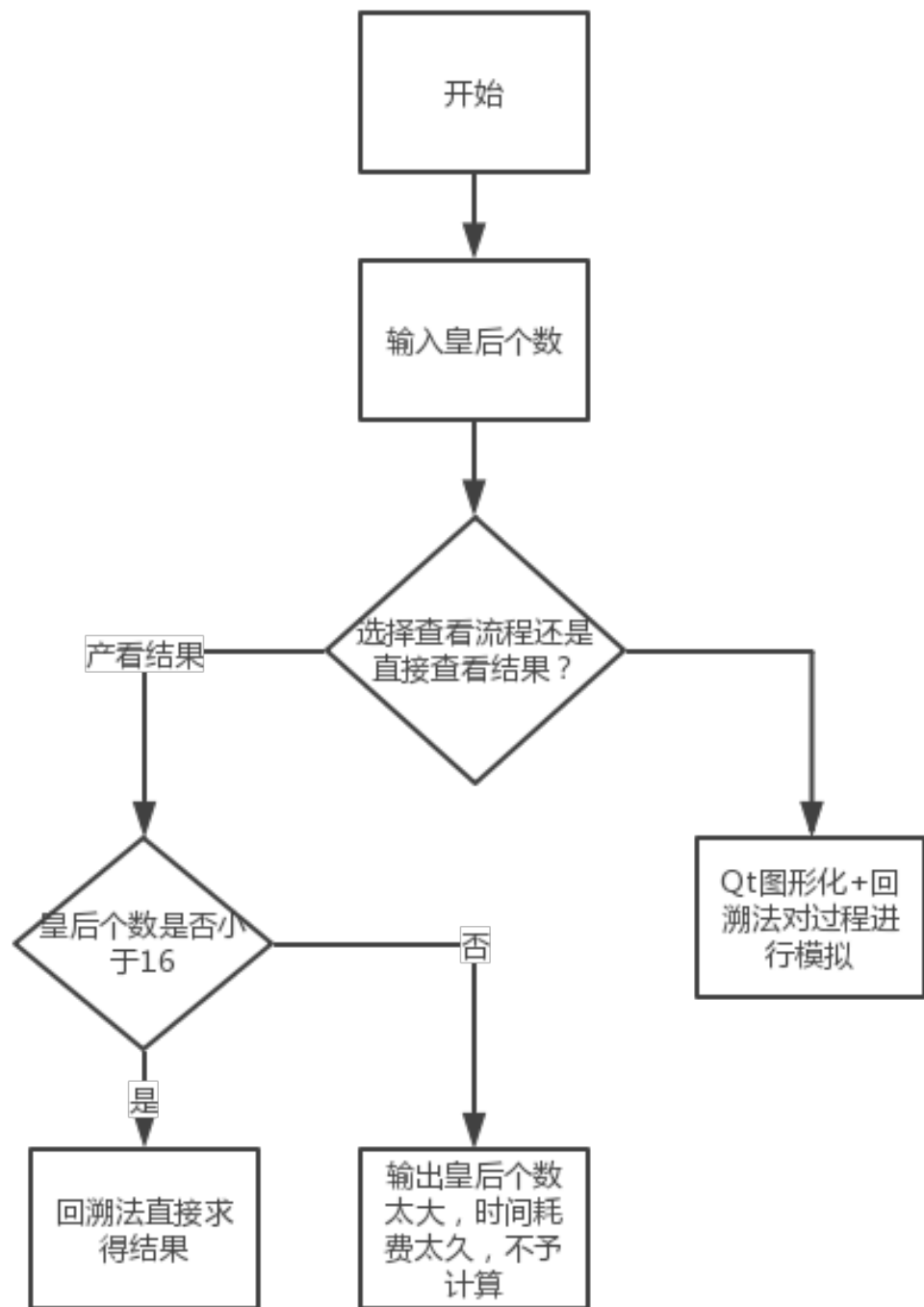


图 2 流程图

输入形式：一个整型数字

输出值形式：

直接输出：n 皇后结 ($n \leq 16$)

输出过程：以图形变换输出求解过程

测试数据：整型数字

3.2 系统设计

本程序中用到的数据结构：图

图的定义：是研究数据元素之间的多对多的关系。在这种结构中，任意两个元素之间可能存在关系。即结点之间的关系可以是任意的，图中任意元素之间都可能相关。

图的 dfs 伪代码：

```
bool DFS(Node n, int d) {
    if (isEnd(n, d)) { // 一旦搜索深度到达一个结束状态，就返回 true
        return true;
    }

    for (Node nextNode in n) { // 遍历 n 相邻的节点 nextNode
        if (!visit[nextNode]) { //
            visit[nextNode] = true; // 在下一步搜索中，nextNode 不能再次出现

            if (DFS(nextNode, d+1)) { // 如果搜索出有解
                // 做些其他事情，例如记录结果深度等
                return true;
            }

            // 重新设置成 false，因为它有可能出现在下一次搜索的别的路径中

            visit[nextNode] = false;
        }
    }

    return false; // 本次搜索无解
}
```

```
}
```

函数调用关系图：

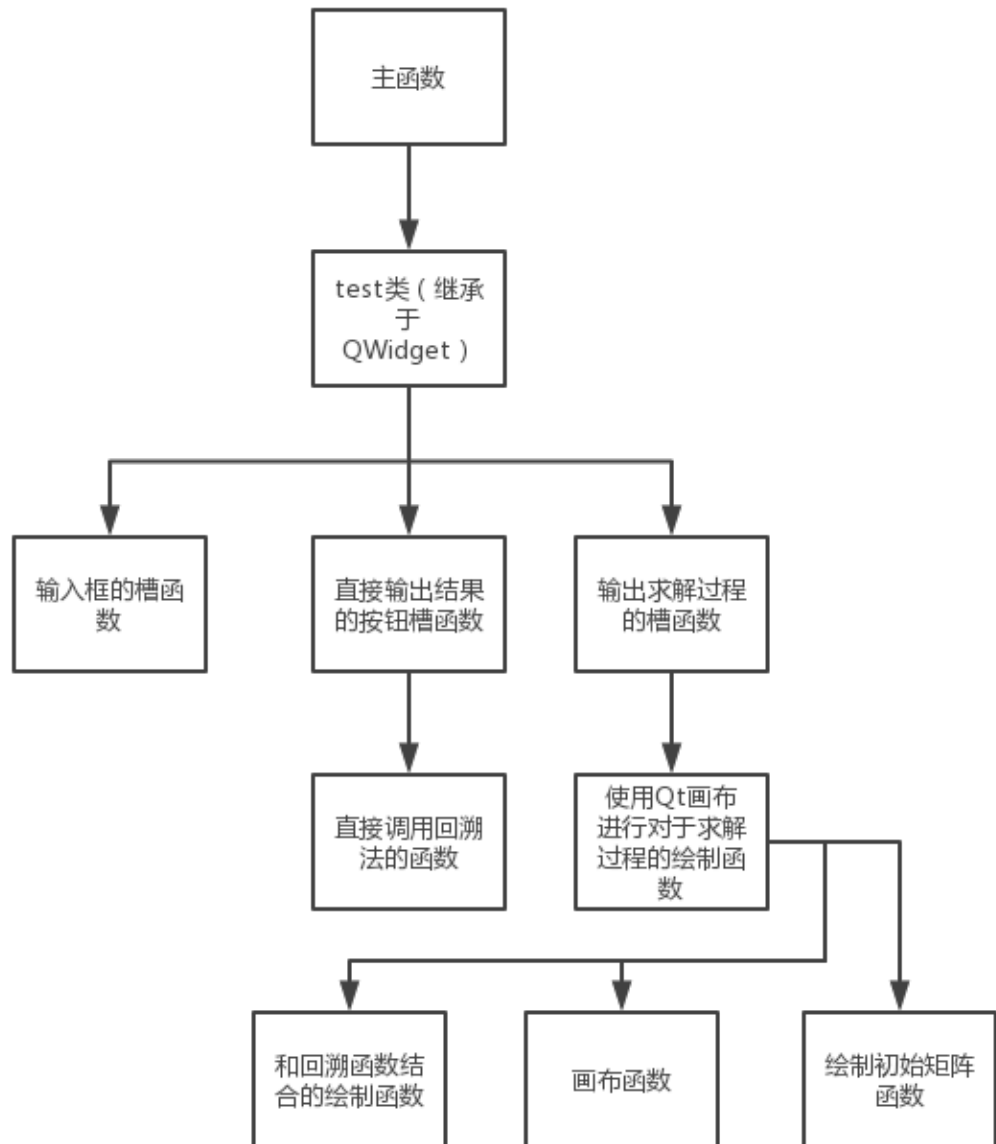


图 3 函数调用关系图

主程序伪码流程：

声明一个 test 类的对象->show 这个对象

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    test ttest;

    ttest.show();
    return a.exec();
}
```

其他模块:

图形化回溯过程模块:

```
void test::ttest(int t)
{
    if(t==testN+1){
        count++;
        ui->count->setText(QString::number(count, 10));
        sleep(500);
    }
    else{
        for(int i=1;i<=testN;i++){
            x[t]=i;
            if(canIncheckerboard(t)){
                transeAnotherColor(t, i);
                sleep(1);
                ttest(t+1);
            }
        }
    }

    this->getPainter()->setBrush(QColor(100, 255, 0));

    this->getPainter()->setPen(QPen(QColor(255, 255, 255), 5));
}
```



```
this->getPainter()->drawRect(1366*1.0/testN*(i-1), 768*1.0/testN*(t-1), 1366*1.0/testN, 768*1.0/testN);  
        this->update();  
        sleep(1);  
    }  
}  
}  
}
```

3.3 系统实现

算法实现:

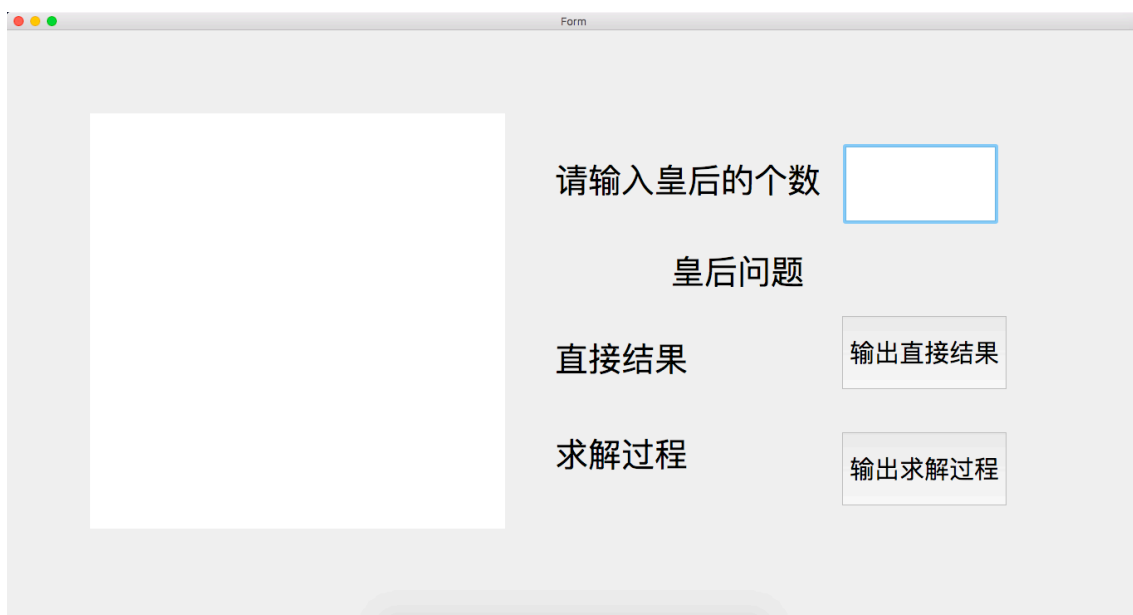
当 $n==t$ 时, 总数+1, 开始回溯, 否则对这一行的所有点进行循环判断, 如果可以放上皇后, 则进入下一行进行递归

问题与回顾分析: 图形化界面的过程显示和递归的过程很那去相结合, 最初的时候曾经想要去模拟把矩阵不能放皇后的地方全部变成特殊的颜色, 比方说黑色, 后来发现这样子做的话很不直观, 会显得很杂乱, 所以每次只展示对应的皇后的位置来展示算法的回溯过程, 通过这种方式, 通过实验可以看出过程。

3.4 用户手册

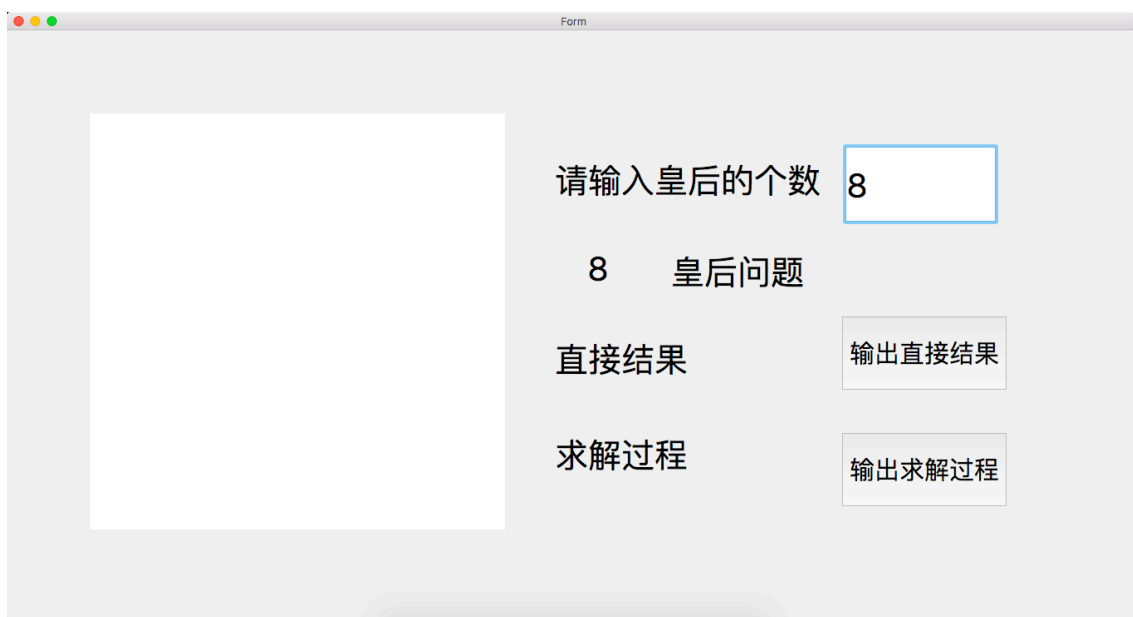
运行程序后, 在输入框中输入对应的数字, 按照对应的数字, 可以点击对应的按钮, 可以输出对应的解和对应的求解过程

3.5 测试



The image shows a software window titled "Form" with a light gray background. On the left side, there is a large, empty white rectangular area. On the right side, the text "请输入皇后的个数" (Please enter the number of queens) is followed by a text input field. Below this, the text "皇后问题" (N-Queens problem) is displayed. Further down, there are two rows of controls: the first row has the text "直接结果" (Direct result) and a button labeled "输出直接结果" (Output direct result); the second row has the text "求解过程" (Solving process) and a button labeled "输出求解过程" (Output solving process).

图 4 初始界面



The image shows the same software window "Form" as in Figure 4, but with the number "8" entered into the text input field. The text "8 皇后问题" (8 N-Queens problem) now appears. The buttons "输出直接结果" and "输出求解过程" remain visible and unchanged.

图 5 输入数字后界面



图 6 点击直接输出结果后界面

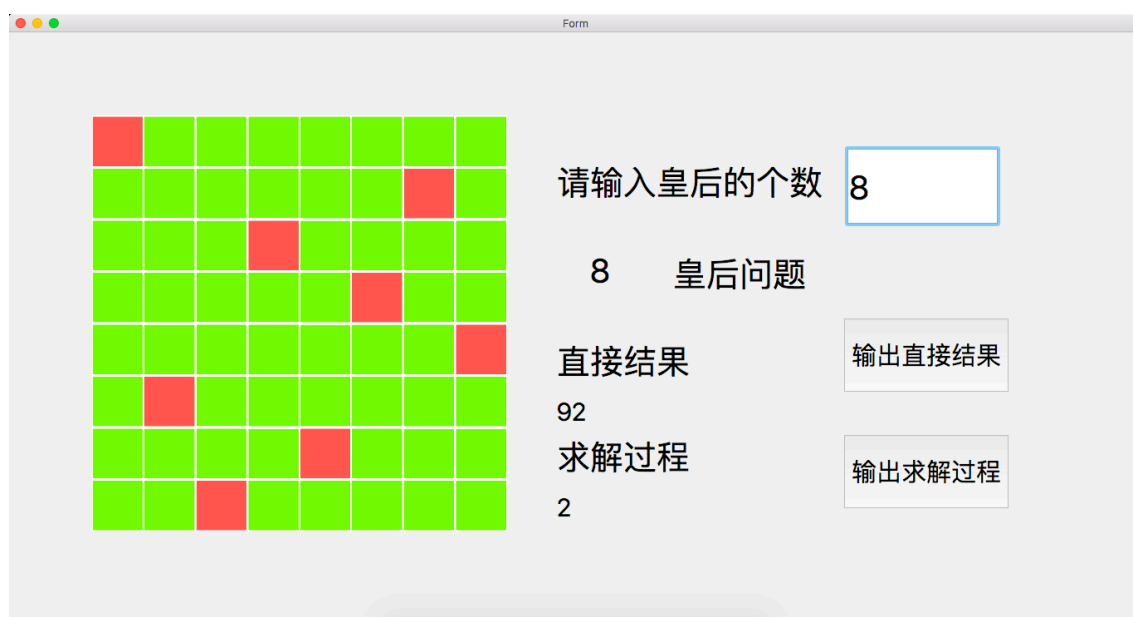


图 7 点击输出求解过程中界面

4. 结束语（设计的成果，展望等）

本次课程设计的程序可以使用于正常的求解八皇后和 n 皇后问题，可以通过图形化界面的过程来体现回溯法的内部逻辑，希望以后能把界面更加美化，效率更加提高。

5. 参考资料

[1] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社. 2017

页

[2]啊哈磊 . 啊哈! 算法[M]. 北京: 人民邮电出版社出版发行. 2016

[3]谭浩强. C++程序设计 (第 3 版) [M]. 北京:清华大学出版社. 2016

6. 附录

main.cpp 文件:

```
#include "mainwindow.h"
#include "paintwidget.h"
#include "test.h"
#include <QApplication>
#include <QPainter>
#include <iostream>
int n;
using namespace std;
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    test ttest;

    ttest.show();
    // ttest.tt();
    return a.exec();
}
```

test.h 文件:

```
#ifndef TEST_H
#define TEST_H

#include <QPushButton>
#include <QWidget>

namespace Ui {
class test;
}

class test : public QWidget
{
    Q_OBJECT

public:
    explicit test(QWidget *parent = 0, Qt::WindowFlags f = 0);
    ~test();
}
```

页

```
int count;
int x[100];
int numBer;
long sum , upperlim ;

QPushButton qbutton;

void resizeEvent(QResizeEvent *event);

void paintEvent(QPaintEvent *event);

QPainter* getPainter();
QPixmap* getPixmap();
void initBackground(int n);
void setBackgroundWhite();
void runResult();

void transeAnotherColor(int x,int y);

bool canIncheckerboard(int k);

void ttest(int t);

void getNumber(long row, long ld, long rd);

private slots:
    void on_lineEdit_cursorPositionChanged(int arg1, int arg2);

    void on_pushButton_2_clicked();
    void on_pushButton_clicked();

private:
    Ui::test *ui;
    int testN;
    QPixmap* mPixmap;
    QPainter* mPainter;
};

#endif // TEST_H
```

test.cpp 文件:

页

```
#include "test.h"
#include "ui_test.h"

#include <QPainter>
#include <QResizeEvent>
#include<QColor>
#include<iostream>
#include <stdio.h>
#include <QTime>
#include <sys/select.h>
#include<cstring>
using namespace std;

int a[100][100];
test::test(QWidget *parent, Qt::WindowFlags f) :
    QWidget(parent, f),
    ui(new Ui::test)
{
    ui->setupUi(this);
    mPixmap = new QPixmap(1366, 768);
    mPainter = new QPainter(mPixmap);
    resize(1366, 768);
    mPixmap->fill(QColor(255, 255, 255));
    count=0;
    numBer=0;
    sum=0;
    upperlim=1;
}

void sleep(unsigned int msec)
{
    QTime dieTime = QTime::currentTime().addMSecs(msec);

    while( QTime::currentTime() < dieTime )

        QCoreApplication::processEvents(QEventLoop::AllEvents,
100);
}

test::~~test()
```

页

```
{
    delete ui;
    delete mPainter;
    delete mPixmap;
}

void test::resizeEvent(QResizeEvent *event)
{
    if(event->size().width() > mPixmap->width() ||
event->size().height() > mPixmap->height())
    {
        QPixmap->fill(QColor(255, 255, 255));
        QPixmap* pixmap = new
QPixmap((std::max)(event->size().width(), mPixmap->width()),
(std::max)(event->size().height(), mPixmap->height()));
        delete mPainter;
        mPainter = new QPainter(pixmap);
        mPainter->drawPixmap(0, 0, *mPixmap);
        delete mPixmap;
        mPixmap = pixmap;
    }
}

void test::paintEvent(QPaintEvent *event)
{
    QPainter painter(this);

    painter.drawPixmap(100, 100, 500, 500, *mPixmap);
}

QPainter *test::getPainter()
{
    return mPainter;
}

QPixmap *test::getPixmap()
{
    return mPixmap;
}

//初始化原始矩阵方块
void test::initBackground(int n)
{
    count=0;
    this->getPainter()->setRenderHint(QPainter::Antialiasing,
```

页

```
true);
    this->getPainter()->setPen(QPen(QColor(255, 255, 255), 5));
    this->getPainter()->setBrush(QColor(100, 255, 0));
    int x=0, y=0;
    for(int i=0; i<n; i++) {
        x=0;
        for(int j=0; j<n; j++) {

            this->getPainter()->drawRect(1366*1.0/testN*(i), 768*1.0/testN*(j),
            1366*1.0/n, 768*1.0/n);
                sleep(1);
                this->update();
            }
            this->update();
        }
    }

void test::setBackgroundWhite()
{
    this->getPainter()->setBrush(QColor(255, 255, 255));
    this->getPainter()->setPen(QPen(QColor(255, 255, 255), 5));
    this->getPainter()->drawRect(0, 0, 3000, 3000);
}

void test::transeAnotherColor(int y, int x)
{
    this->getPainter()->setBrush(QColor(255, 64, 64));
    this->getPainter()->setPen(QPen(QColor(255, 255, 255), 5));

    this->getPainter()->drawRect(1366*1.0/testN*(x-1), 768*1.0/testN*(y-1),
    1366*1.0/testN, 768*1.0/testN);
        this->update();
    }

bool test::canIncheckerboard(int k)
{
    for(int i=1; i<k; i++) {
        if(abs(x[k] - x[i]) == abs(k-i) || x[i] == x[k])
            return false;
    }
}
```


页

```
    }
    return true;
}

void test::ttest(int t)
{
    if(t==testN+1){
        count++;
        ui->count->setText(QString::number(count, 10));
        sleep(500);
    }
    else{
        for(int i=1;i<=testN;i++){
            x[t]=i;
            if(canIncheckerboard(t)){
                transeAnotherColor(t, i);
                sleep(1);
                ttest(t+1);
                this->getPainter()->setBrush(QColor(100, 255, 0));

                this->getPainter()->setPen(QPen(QColor(255, 255, 255), 5));

                this->getPainter()->drawRect(1366*1.0/testN*(i-1), 768*1.0/testN*
                (t-1), 1366*1.0/testN, 768*1.0/testN);
                this->update();
                sleep(1);
            }
        }
    }
}

void test::getNumber(long row, long ld, long rd)
{
    if (row != upperlim)
    {
        // row, ld, rd 进行“或”运算，求得所有可以放置皇后的
        列, 对应位为 0,
        // 然后再取反后“与”上全 1 的数，来求得当前所有可以放
        置皇后的位置，对应列改为 1
        // 也就是求取当前哪些列可以放置皇后
        long pos = upperlim & ~(row | ld | rd);
        while (pos)    // 0 -- 皇后没有地方可放，回溯
        {
            // 拷贝 pos 最右边为 1 的 bit，其余 bit 置 0
```

页

```

        // 也就是取得可以放皇后的最右边的列
        long p = pos & -pos;

        // 将 pos 最右边为 1 的 bit 清零
        // 也就是为获取下一次的最右可用列使用做准备,
        // 程序将来会回溯到这个位置继续试探
        pos -= p;

        // row + p, 将当前列置 1, 表示记录这次皇后放置的
        列。

        // (ld + p) << 1, 标记当前皇后左边相邻的列不允许
        下一个皇后放置。
        // (ld + p) >> 1, 标记当前皇后右边相邻的列不允许
        下一个皇后放置。
        // 此处的移位操作实际上是记录对角线上的限制, 只是
        因为问题都化归
        // 到一行网格上来解决, 所以表示为列的限制就可以
        了。显然, 随着移位
        // 在每次选择列之前进行, 原来 N×N 网格中某个已放
        置的皇后针对其对角线
        // 上产生的限制都被记录下来了
        getNumber(row + p, (ld + p) << 1, (rd + p) >> 1);
    }
}
else
{
    // row 的所有位都为 1, 即找到了一个成功的布局, 回溯
    numBer++;
}
}

//求得输入框的信号槽
void test::on_lineEdit_cursorPositionChanged(int arg1, int arg2)
{
    testN=ui->lineEdit->text().toInt();
    ui->testNNumber->setText(ui->lineEdit->text());
    ui->count->setText(QString(""));
    ui->number->setText(QString(""));
    setBackgroundWhite();
    ui->number->setText(QString(""));
    this->update();
}

```

页

```
//test2Button 的信号槽
void test::on_pushButton_2_clicked()
{
    memset(x, 0, sizeof(x));
    count=0;
    ui->count->setText(QString::number(count, 10));
    setBackgroundWhite();
    initBackground(testN);
    ttest(1);
    ui->count->setText(QString::number(count, 10));
}

void test::on_pushButton_clicked()
{
    if(numBer==0&&testN<17) {
        sum=0;
        upperlim=1;
        upperlim = (upperlim << testN) - 1;
        memset(x, 0, sizeof(x));
        ui->number->setText(QString("请稍后"));
        this->update();
        sleep(10);
        numBer=0;
        getNumber(0, 0, 0);
        ui->number->setText(QString::number(numBer, 10));
        numBer=0;
    }
    else if(testN>=17) {
        ui->number->setText(QString("您输入的数字太大, 计算时间过长"));
    }
}
```

test.ui 文件:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>test</class>
    <widget class="QWidget" name="test">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>2000</width>
```

页

```
<height>1000</height>
</rect>
</property>
<property name="windowTitle">
  <string>Form</string>
</property>
<widget class="QPushButton" name="pushButton">
  <property name="geometry">
    <rect>
      <x>1000</x>
      <y>340</y>
      <width>210</width>
      <height>100</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>30</pointsize>
    </font>
  </property>
  <property name="text">
    <string>输出直接结果</string>
  </property>
</widget>
<widget class="QPushButton" name="pushButton_2">
  <property name="enabled">
    <bool>true</bool>
  </property>
  <property name="geometry">
    <rect>
      <x>1000</x>
      <y>480</y>
      <width>210</width>
      <height>100</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>30</pointsize>
      <strikeout>>false</strikeout>
      <stylestrategy>PreferDefault</stylestrategy>
      <kerning>true</kerning>
    </font>
  </property>
```

页

```
<property name="styleSheet">
  <string notr="true"/>
</property>
<property name="text">
  <string>输出求解过程</string>
</property>
<property name="iconSize">
  <size>
    <width>16</width>
    <height>16</height>
  </size>
</property>
<property name="checkable">
  <bool>false</bool>
</property>
<property name="autoDefault">
  <bool>false</bool>
</property>
</widget>
<widget class="QLineEdit" name="lineEdit">
  <property name="geometry">
    <rect>
      <x>1010</x>
      <y>140</y>
      <width>181</width>
      <height>90</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>40</pointsize>
    </font>
  </property>
  <property name="inputMethodHints">
    <set>Qt::ImhNone</set>
  </property>
  <property name="text">
    <string/>
  </property>
</widget>
<widget class="QLabel" name="label">
  <property name="geometry">
    <rect>
      <x>660</x>
```

页

```
<y>140</y>
<width>341</width>
<height>81</height>
</rect>
</property>
<property name="font">
  <font>
    <pointsize>40</pointsize>
  </font>
</property>
<property name="text">
  <string>请输入皇后的个数</string>
</property>
</widget>
<widget class="QLabel" name="label_2">
  <property name="geometry">
    <rect>
      <x>800</x>
      <y>260</y>
      <width>161</width>
      <height>61</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>40</pointsize>
    </font>
  </property>
  <property name="text">
    <string>皇后问题</string>
  </property>
</widget>
<widget class="QLabel" name="testNNumber">
  <property name="geometry">
    <rect>
      <x>700</x>
      <y>250</y>
      <width>71</width>
      <height>71</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>40</pointsize>
```

页

```
</font>
</property>
<property name="text">
  <string/>
</property>
</widget>
<widget class="QLabel" name="label_4">
  <property name="geometry">
    <rect>
      <x>660</x>
      <y>370</y>
      <width>161</width>
      <height>51</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>40</pointsize>
    </font>
  </property>
  <property name="text">
    <string>直接结果</string>
  </property>
</widget>
<widget class="QLabel" name="number">
  <property name="geometry">
    <rect>
      <x>660</x>
      <y>440</y>
      <width>541</width>
      <height>31</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>30</pointsize>
    </font>
  </property>
  <property name="text">
    <string/>
  </property>
</widget>
<widget class="QLabel" name="label_3">
  <property name="geometry">
```

页

```
<rect>
  <x>660</x>
  <y>480</y>
  <width>251</width>
  <height>61</height>
</rect>
</property>
<property name="font">
  <font>
    <pointsize>40</pointsize>
  </font>
</property>
<property name="text">
  <string>求解过程</string>
</property>
</widget>
<widget class="QLabel" name="count">
  <property name="geometry">
    <rect>
      <x>660</x>
      <y>550</y>
      <width>321</width>
      <height>41</height>
    </rect>
  </property>
  <property name="font">
    <font>
      <pointsize>30</pointsize>
    </font>
  </property>
  <property name="text">
    <string/>
  </property>
</widget>
</widget>
<resources/>
<connections/>
</ui>
```

test.pro 文件:

```
#-----
#
# Project created by QtCreator 2017-12-29T18:11:26
```


页

```
#
#-----

QT      += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = test
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact
# warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from
# it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated
# APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain
# version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables
# all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    test.cpp

HEADERS += \
    test.h

FORMS += \
    test.ui
```

四、设计时间与安排

- 1、设计时间： 1 周
- 2、设计时间安排：

熟悉实验设备、收集资料：3 天

设计图纸、实验、计算、程序编写调试：1 天

编写课程设计报告：2 天

答辩：1 天