中国矿业大学计算机学院

2019 级本科生课程设计报告

课程名称	程序设计综合实践
报告时间	2021.1.20
学生姓名	许万鹏
学　　号	05191643
专　　业	信息安全
任课教师	薛猛

成绩考核

编号	课程教学目标	占比	得分
1	**目标 1：**掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。	20%	
2	**目标 2：**针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。	20%	
3	**目标 3：**在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。	60%	
总成绩			
指导教师		评阅日期	

目 录

实验一 简单计算器

1 系统概述

该程序名为简单计算器，设计思路来源于微软计算器，使用 QT 开发框架配合 C++语言完成。利用栈解决优先级问题、通过中缀转后缀计算数值。在连续四则运算计算器的基础上增加科学运算，支持幂运算、三角函数运算、反三角函数运算、指数运算、对数运算等运算，增加贷款计算、进制转换等功能，优化交互体验，手感良好，可用于生产应用。

2 系统设计

2.1 设计目标

该项目的主要内容是设计开发一个支持连续计算的简单计算器，并在此基础上增加部分科学运算及其他功能，通过点击相应的数字按钮和运算按钮，完成如$1+6*9, \log(100), \sin(\pi \div 6)$等数学运算，并将运算结果显示，应具备退格、清空功能以实现重复运算。

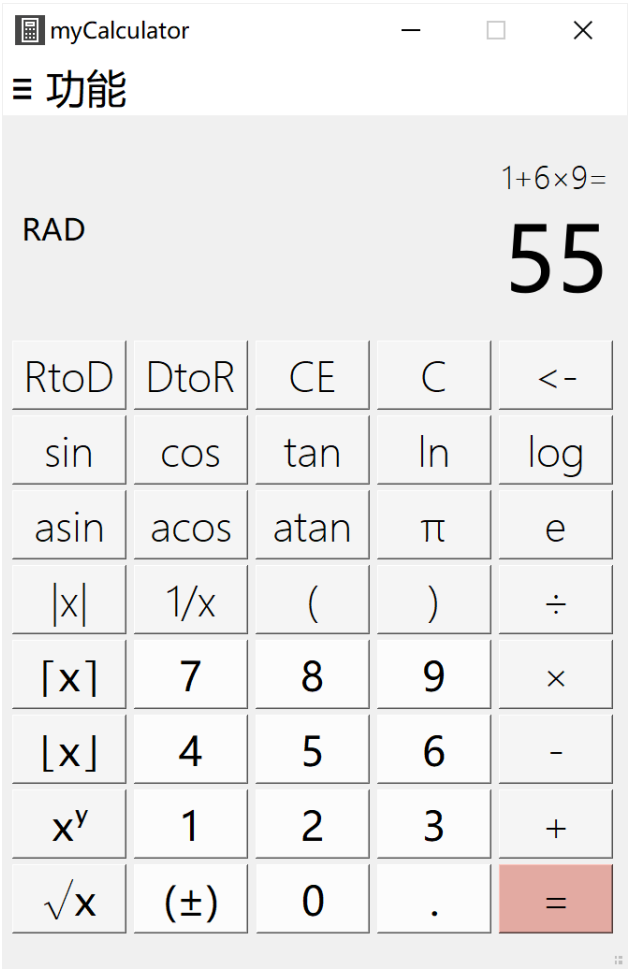


图 1.1 计算器整体布局

2.2 设计分析与算法流程

为实现括号匹配与表达式运算，我们需要利用栈将中缀表示转为后缀表示并求值。

2.2.1 中缀转后缀

首先构造 getpostfix 函数将中缀表达式转换成后缀表达式，代码如下：

```
void getPostFix(char *in, char *post)
{
    stack<char> st;//创建一个字符类的栈，用于存放操作符
    st.push('@');//存一个@进栈底，方便于最后判断栈里的操作符是否已经全部出栈
    int i = 0;
    int j = 0;
    while (in[i] != '#')//'#'为字符数组的最后一个元素
    {
        char ch = in[i];
        if (isNum(ch))
        {
            post[j++] = ch;
            i++;
        }
        else if (isChar(ch))
        {
            i++;
            char ch1 = in[i];
            while (isNum(ch1))
            {
                post[j++] = ch1;
                i++;
                ch1 = in[i];
            }
            post[j++] = ' ';//用空格分开数和函数名
            post[j++] = ch; //将函数名的缩写放进后缀表达式的数组
        }
        else
        {
            post[j++] = ' ';//用空格隔开操作符前后的数
            if (ch == '(')//如果字符是'('，则将字符进栈
            {
                st.push(ch);

            }
            else if (ch == ')')  //若字符是')'，则将栈中的字符输进后缀表达式的字符数组，直至遇到'('
            {
                ch = st.top();
                while (ch != '(')
                {
                    post[j++] = ch;
                    st.pop();
                    ch = st.top();
                }
                st.pop(); //将'('出栈

            }
            else
```

```
50              {
51                  int thisPri = getPri(ch);  //获取操作符的优先级
52                  char prevOpt = st.top();  //获取栈顶操作符
53                  int prevPri = getPri(prevOpt); //获取栈顶操作符的优先级
54                  while (thisPri <= prevPri)//比较两者的优先级，若栈顶操作符的优
55  先级高，则将栈顶操作符放进后缀表达式，并将其出栈
56                  {
57                      post[j++] = prevOpt;//将栈顶操作符放进后缀表达式
58                      st.pop();  //将其出栈
59                      prevOpt = st.top();//取栈中新栈顶的操作符
60                      prevPri = getPri(prevOpt);//同理获取栈中新栈顶的操作符的优
61  先级
62                  }
63                  st.push(ch);  //把新的操作符进栈
64              }
65              i++;
66          }
67      }
68
69      char ch = st.top();
70      while (ch != '@')
71      {
72          post[j++] = ch;
73          st.pop();
74          ch = st.top();
75      }//当栈底元素不是'@'时，将栈顶元素放进后缀表达时的字符数组
76      post[j] = '\0';
77  }
78
79
80
```

其中 isNum()用于判断是否是数字，若是则把其放进后缀字符数组，代码如下：

```
1           int isNum(char c)
2           {
3               switch (c){
4               case '0':
5               case '1':
6               case '2':
7               case '3':
8               case '4':
9               case '5':
10              case '6':
11              case '7':
12              case '8':
13              case '9':
14              case '.':
15                  return 1; break;
16              default:
17                  return 0; break;
18              }
19          }
```

其中 isChar()用于判断字符是否是常见（单目运算）函数的缩写，若是则将函数后面的数放进后缀字符数组，再将函数名缩写放进后缀数组，代码如下：

```
int isChar(char c)
{
    switch (c){
    case 's'://sin 函数
    case 'c'://cos 函数
    case 't'://tan 函数
    case 'a'://abs 函数
    case 'q'://sqrt 函数
    case 'l'://ln 函数
    case 'g'://log10 函数
    case 'u'://ceil 函数
    case 'd'://floor 函数
    case 'i'://asin 函数
    case 'j'://acos 函数
    case 'k'://atan 函数
    case 'm'://RtoD 函数
    case 'n'://DtoR 函数
    case 'r'://rec 函数
    case 'w'://sqrt 函数
    case 'p'://plusorminus 函数
        return 1; break;
    default:
        return 0; break;
    }
}
```

其中 getPri()函数用于获取优先级，代码如下：

```
int getPri(char c)
{
    switch (c) {
    case '+':
        return 1; break;
    case '-':
        return 1; break;//'+'和'-'的优先级为最低级 1
    case '*':
        return 2; break;
    case '/':
        return 2; break;//'*'和'/'的优先级为中级 2
    case '^':
        return 3; break;//'^'的优先级最高，为 3
    default:
        return 0; break;
    }
}
```

2.2.2 对后缀表达式求值

得到后缀表达式后，我们构造 getPAnswer 函数，利用栈和数组对后缀表达式进行求值。

```
1    double getPAnswer(char *dest)
2    {
3        double x[50];//创建一个数组，用于存放数据
4        int len = 0;
5        int i = 0;
6        while (dest[i] != '\0')//当字符数组中的字符不是'\0'时
7        {
8            if (dest[i] >= '0'&&dest[i] <= '9')
9                x[len++] = readNumber(dest, &i);
10           else if (isChar(dest[i]) || isOpr(dest[i]))
11           {
12               switch (dest[i])
13               {
14               case '+'://加法
15                   x[len - 2] = x[len - 2] + x[len - 1];
16                   len--;
17                   i++;
18                   break;
19               case '-'://加法
20                   x[len - 2] = x[len - 2] - x[len - 1];
21                   len--;
22                   i++;
23                   break;
24               case '*'://乘法
25                   x[len - 2] = x[len - 2] * x[len - 1];
26                   len--;
27                   i++;
28                   break;
29               case '/'://除法
30                   x[len - 2] = x[len - 2] / x[len - 1];
31                   len--;
32                   i++;
33                   break;
34               case '^'://幂运算
35                   x[len - 2] = qPow(x[len - 2], x[len - 1]);
36                   len--;
37                   i++;
38                   break;
39               case 's'://sin 函数
40                   x[len - 1] = sin(x[len - 1]);
41                   i++;
42                   break;
43               case 'c'://cos 函数
44                   x[len - 1] = cos(x[len - 1]);
45                   i++;
46                   break;
47               case 't'://tan 函数
48                   x[len - 1] = tan(x[len - 1]);
49                   i++;
50                   break;
51               case 'a'://abs 函数
52                   x[len - 1] = abs(x[len - 1]);
53                   i++;
```

```
54                  break;
55              case 'q'://开方函数
56                  x[len - 1] = sqrt(x[len - 1]);
57                  i++;
58                  break;
59              case 'l'://ln 函数
60                  x[len - 1] = log(x[len - 1]);
61                  i++;
62                  break;
63              case 'g'://log10 函数
64                  x[len - 1] = log10(x[len - 1]);
65                  i++;
66                  break;
67              case 'u'://ceil 函数
68                  x[len - 1] = ceil(x[len - 1]);
69                  i++;
70                  break;
71              case 'd'://floor 函数
72                  x[len - 1] = floor(x[len - 1]);
73                  i++;
74                  break;
75              case 'i'://asin 函数
76                  x[len - 1] = asin(x[len - 1]);
77                  i++;
78                  break;
79              case 'j'://acos 函数
80                  x[len - 1] = acos(x[len - 1]);
81                  i++;
82                  break;
83              case 'k'://atan 函数
84                  x[len - 1] = atan(x[len - 1]);
85                  i++;
86                  break;
87              case 'm'://RtoD 函数
88                  x[len - 1] = RtoD(x[len - 1]);
89                  i++;
90                  break;
91              case 'n'://DtoR 函数
92                  x[len - 1] = DtoR(x[len - 1]);
93                  i++;
94                  break;
95              case 'r'://rec 函数
96                  x[len - 1] = rec(x[len - 1]);
97                  i++;
98                  break;
99              case 'w'://sqrt 函数
100                 x[len - 1] = sqrt(x[len - 1]);
101                 i++;
102                 break;
103             case 'p'://pos 函数
104                 x[len - 1] = plusorminus(x[len - 1]);
105                 i++;
106                 break;
107             }
108         }
109         else i++;
110     }
111     return x[len - 1];
```

```
112  }
```

其中 isOpr()函数用于判断字符是否是二目操作数，代码如下：

```
1          int isOpr(char c)
2          {
3              switch (c){
4              case '+':
5              case '-':
6              case '*':
7              case '/':
8              case '^':
9                  return 1; break;
10             default:
11                 return 0; break;
12             }
13         }
```

其中 readNumber()函数用于将数字从字符型转换为浮点型，代码如下：

```
1      double readNumber(char *dest, int *i)
2      {
3          double x = 0;
4          int num = 0;
5          int j;
6          while (dest[*i]<'0' || dest[*i]>'9')
7              (*i)++;
8          while (dest[*i] >= '0'&&dest[*i] <= '9')
9          {
10             x = x * 10 + (dest[*i] - '0');
11             (*i)++;
12         }
13         if (dest[*i] == '.')
14         {
15             (*i)++;
16             while (dest[*i] >= '0'&&dest[*i] <= '9')
17             {
18                 num++;
19                 x = x * 10 + (dest[*i] - '0');
20                 (*i)++;
21             }
22         }
23         for (j = 0; j < num; j++)
24             x = x / 10;
25         return x;
26     }
```

2.3 界面设计

2.3.1 主界面设计

本程序中按钮较多，我们采用 QT Designer 来进行 UI 设计，最终效果如图 1.2 所示。

图 1.2 计算器主界面

2.3.2 按键绑定

对通过 UI 生成的按钮进行“转到槽”操作建立槽函数，槽函数名称为 on_按钮名称_clicked()，这里给出按钮名称（图 1.3）。

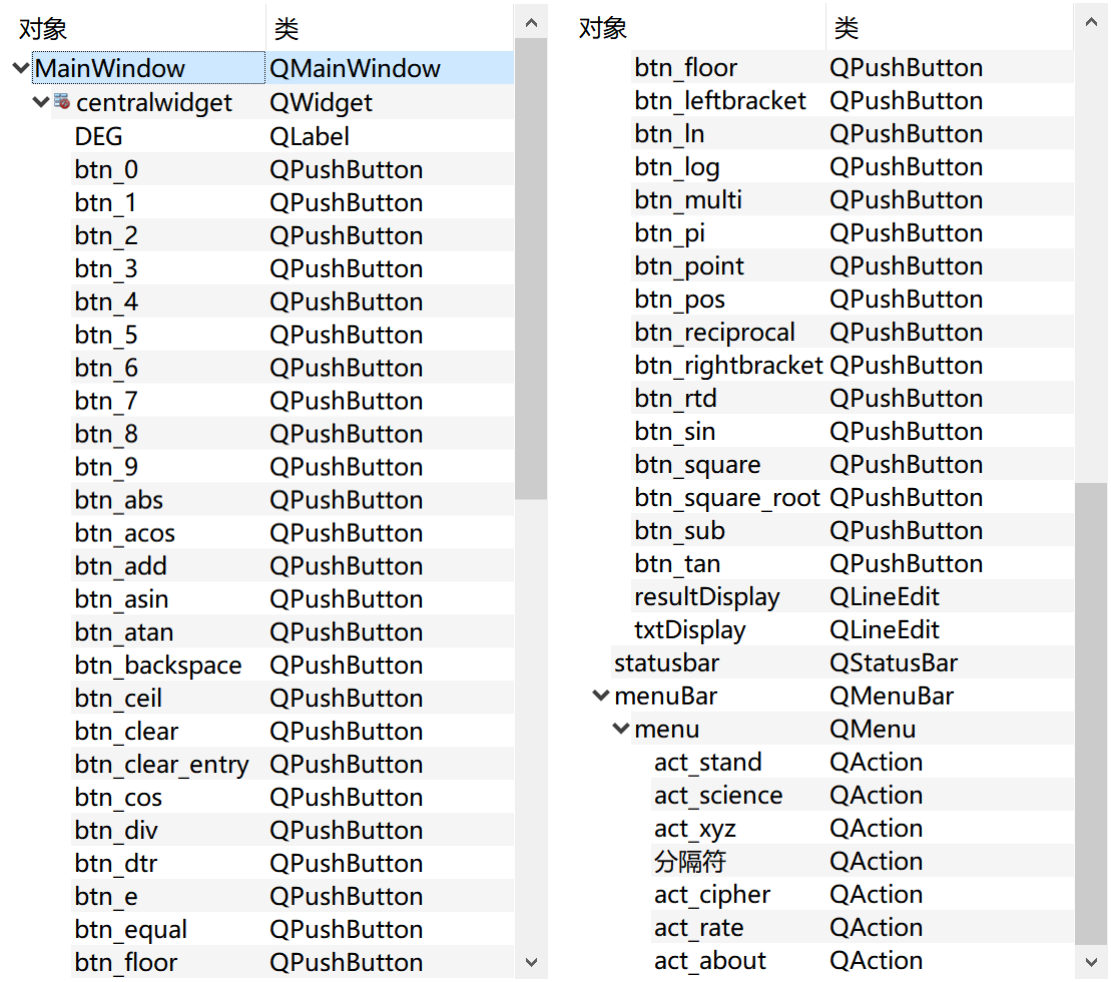


图 1.3 主界面的按键绑定

2.4 关键类图

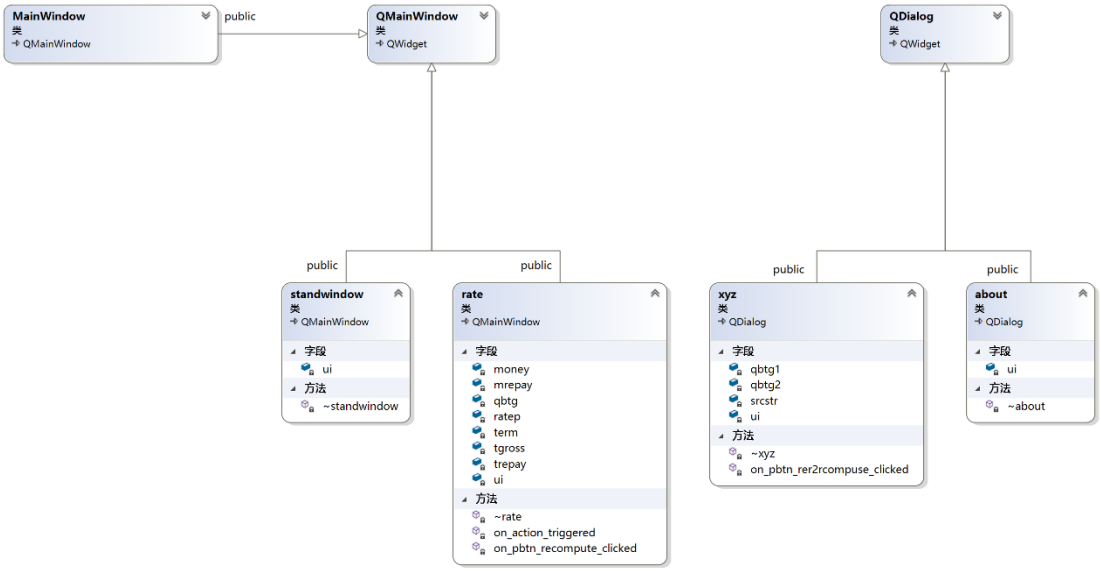


图 1.4 关键类图（MainWindow 折叠）

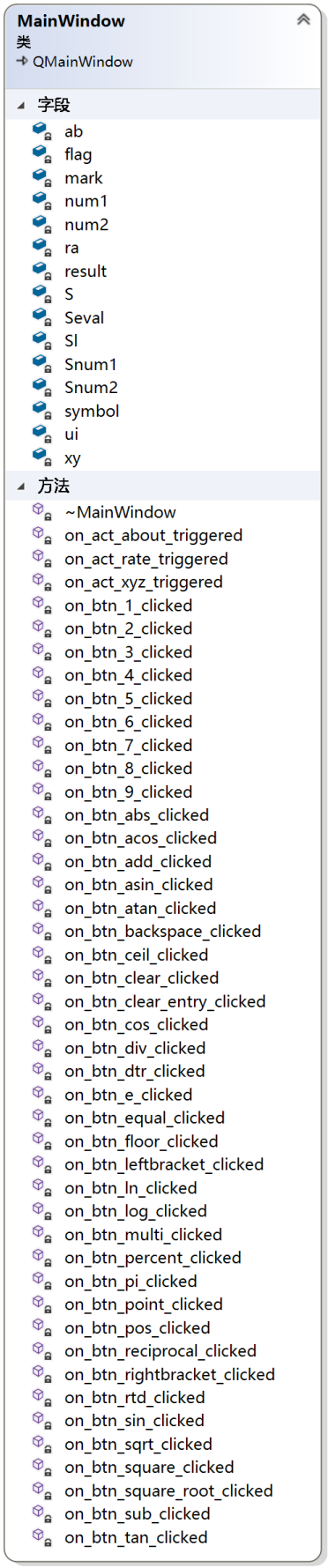


图 1.6 关键类图（MainWindow）

3 系统实现（运行调试）

一些示例操作，见图 1.7-图 1.9

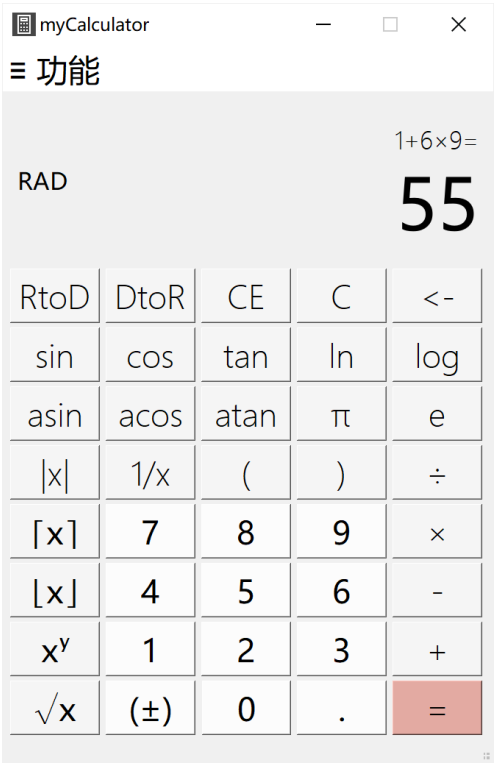


图 1.7 示例操作 1

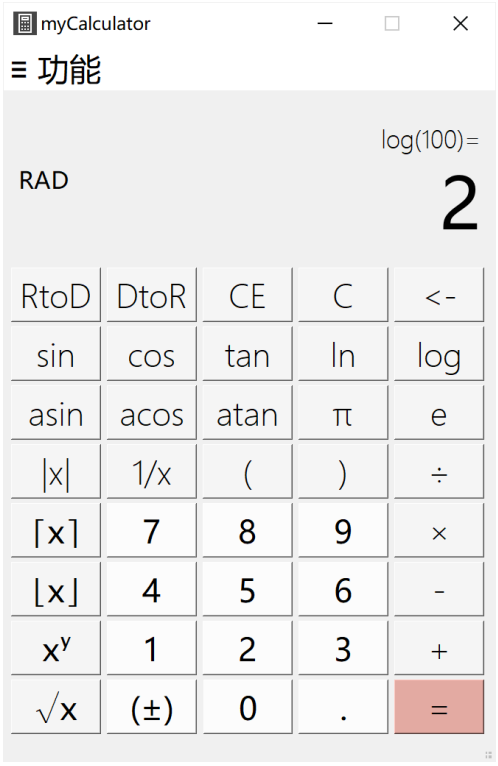


图 1.8 示例操作 2

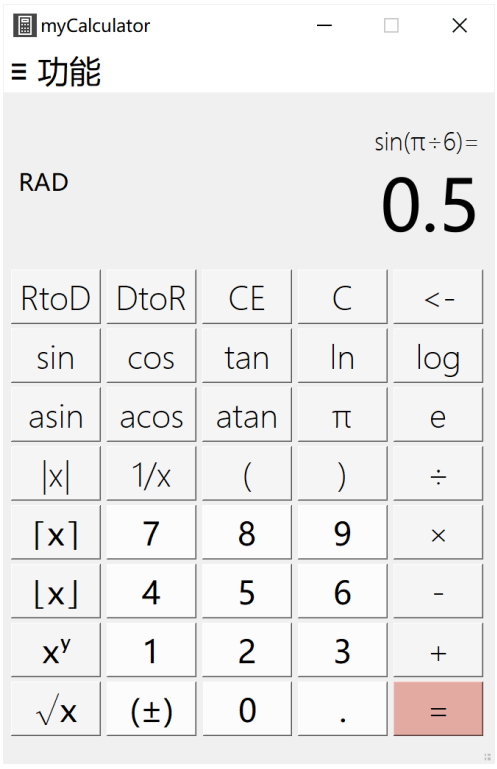


图 1.9 实例操作 3

图 1.10 实例操作 4

4 系统扩展

4.1 进制转换

为了使本计算器对程序员更友好，特添加了进制转换功能，界面如图 1.4。

图 1.4　进制转换功能

该功能通过选中待转换进制与欲转换进制实现，故建立两个 QbuttonGroup 分别作为待转换进制按钮与欲转换进制按钮的容器，代码如下：

```
1     xyz::xyz(QWidget *parent) :
2         QDialog(parent),
3         ui(new Ui::xyz)
4     {
5         ui->setupUi(this);
6         qbtg1 = new QButtonGroup(this);
7
8         qbtg2 = new QButtonGroup(this);
9
10        qbtg1->addButton(ui->rbtn1_2);
11        qbtg1->addButton(ui->rbtn1_8);
12        qbtg1->addButton(ui->rbtn1_10);
13        qbtg1->addButton(ui->rbtn1_16);
14
15        qbtg1->setId(ui->rbtn1_2,2);
16        qbtg1->setId(ui->rbtn1_8,8);
17        qbtg1->setId(ui->rbtn1_10,10);
18        qbtg1->setId(ui->rbtn1_16,16);
19
20        qbtg2->addButton(ui->rbtn2_2);
21        qbtg2->addButton(ui->rbtn2_8);
22        qbtg2->addButton(ui->rbtn2_10);
23        qbtg2->addButton(ui->rbtn2_16);
24
25        qbtg2->setId(ui->rbtn2_2,2);
26        qbtg2->setId(ui->rbtn2_8,8);
27        qbtg2->setId(ui->rbtn2_10,10);
28        qbtg2->setId(ui->rbtn2_16,16);
29
30        ui->le_2r->setAlignment(Qt::AlignRight);
```

```
31        ui->le_r2->setAlignment(Qt::AlignRight);
32
33        ui->rbtn1_10->setChecked(true);
34        ui->rbtn2_2->setChecked(true);
35    }
```

槽函数实现代码如下：

```
1   void xyz::on_pbtn_r2rcompuse_clicked()
2   {
3       int rto=qbtg1->checkedId();
4       QString srcstr0=ui->le_r2->text();
5       QString srcstr;
6       bool ok;
7       switch (rto) {
8       case 2:srcstr=QString::number(srcstr0.toInt(&ok,2));break;
9       case 8:srcstr=QString::number(srcstr0.toInt(&ok,8));break;
10      case 10:srcstr=QString::number(srcstr0.toInt(&ok,10));break;
11      case 16:srcstr=QString::number(srcstr0.toInt(&ok,16));break;
12      }
13      int tor=qbtg2->checkedId();
14      QString rststr;
15      switch (tor) {
16      case 2:rststr=QString::number(srcstr.toInt(),2);break;
17      case 8:rststr=QString::number(srcstr.toInt(),8);break;
18      case 10:rststr=QString::number(srcstr.toInt(),10);break;
19      case 16:rststr=QString::number(srcstr.toInt(),16);break;
20      }
21      ui->le_2r->setText(rststr);
22  }
```

4.2 贷款计算

根据书中提示制作了通过子窗口实现的贷款计算器，界面如图 1.5。

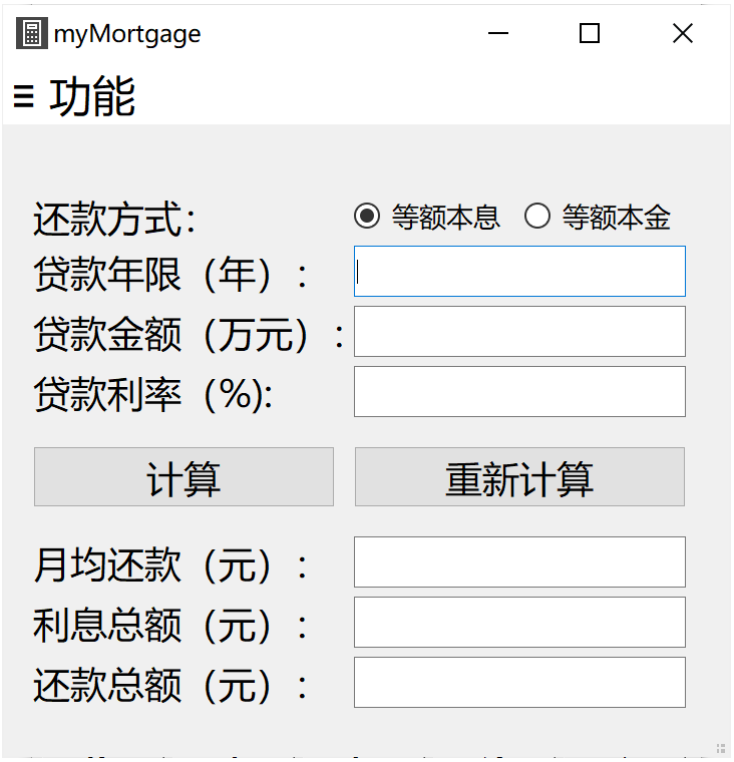


图 1.5　贷款计算器界面

该功能通过如下函数实现：

①等额本息方式

月均还款＝[贷款本金×月利率×(1＋月利率)^还款月数]÷[(1＋月利率)^还款月数－1]

月均还款＝贷款本金×月利率×[(1＋月利率)^还款月数－(1＋月利率)^(还款月序号－1)]÷[(1＋月利率)^还款月数－1]

总利息＝还款月数×每月月供额－贷款本金

②等额本金方式

月均还款＝(贷款本金÷还款月数)＋(贷款本金－已归还本金累计额)×月利率

月均利息＝剩余本金×月利率

＝(贷款本金－已归还本金累计额)×月利率

总利息＝[(总贷款额÷还款月数＋总贷款额×月利率)＋总贷款额÷还款月数×(1＋月利率)]÷2×还款月数－总贷款额

建立 QbuttonGroup 的方法类似，代码如下：

```
1       rate::rate(QWidget *parent) :
2           QMainWindow(parent),
```

```
3           ui(new Ui::rate)
4       {
5           ui->setupUi(this);
6           qbtg = new QButtonGroup(this);
7
8           qbtg->addButton(ui->rbtn_blp);
9           qbtg->addButton(ui->rbtn_bpp);
10
11          qbtg->setId(ui->rbtn_blp, 0);
12          qbtg->setId(ui->rbtn_bpp, 1);
13
14          ui->rbtn_blp->setChecked(true);
15      }
```

槽函数实现代码如下：

```
1    void rate::on_pbtn_compute_clicked()
2    {
3        int a = qbtg->checkedId();
4        term=ui->loan_term->text().toInt();
5        money=ui->loan_amount->text().toDouble()*10000;
6        ratep=ui->loan_rate->text().toFloat()/100/12;
7        switch(a)
8        {
9        case 0:
10           mrepay=money*ratep*qPow((1+ratep),term*12)/(qPow((1+ratep),
                 term*12)-1);
12           trepay=mrepay*12*term;
13           tgross=trepay-money;
14           ui->mon_repay->setText(QString::number(mrepay,'g'));
15           ui->repay_total->setText(QString::number(trepay,'g'));
16           ui->gross_total->setText(QString::number(tgross,'g'));
17           break;
18       case 1:
19          tgross=((money/(term*12)+money*ratep)+money/(term*12)*
                 (1+ratep))/2*(term*12)-money;
21           trepay=tgross+money;
22           mrepay=trepay/(term*12);
23           ui->mon_repay->setText(QString::number(mrepay,'g'));
24           ui->repay_total->setText(QString::number(trepay,'g'));
25           ui->gross_total->setText(QString::number(tgross,'g'));
26           break;
27       default:
28           break;
29       }
30   }
```

5 总结

这是本课程中要求中的第一个任务，虽然粗糙却也花费了很多时间，通过简单计算器的制作，我学会了使用 QT 的信号与槽机制、UI Designer 等，将面向对象编程知识实际应用并加深了理解。

实验二 多文本编辑器

1 系统概述

该程序是一个简单的多文本编辑器，使用 QT 开发框架配合 C++语言完成，设计思路粗略提取自 Micsoft Word，为作为 MDI 应用程序，具有新建、打开、保存、另存为等基础功能，并在此基础上复现相关 office 操作，如：字体、字号、颜色、段落标号等。交互感优，可用于生产应用。

2 系统设计

2.1 设计目标

项目的基本目标是设计开发一个简单的多文档文本编辑器，具有新建、打开、保存一个文本文件，设置字体、字型功能，运行界面见图 2.1

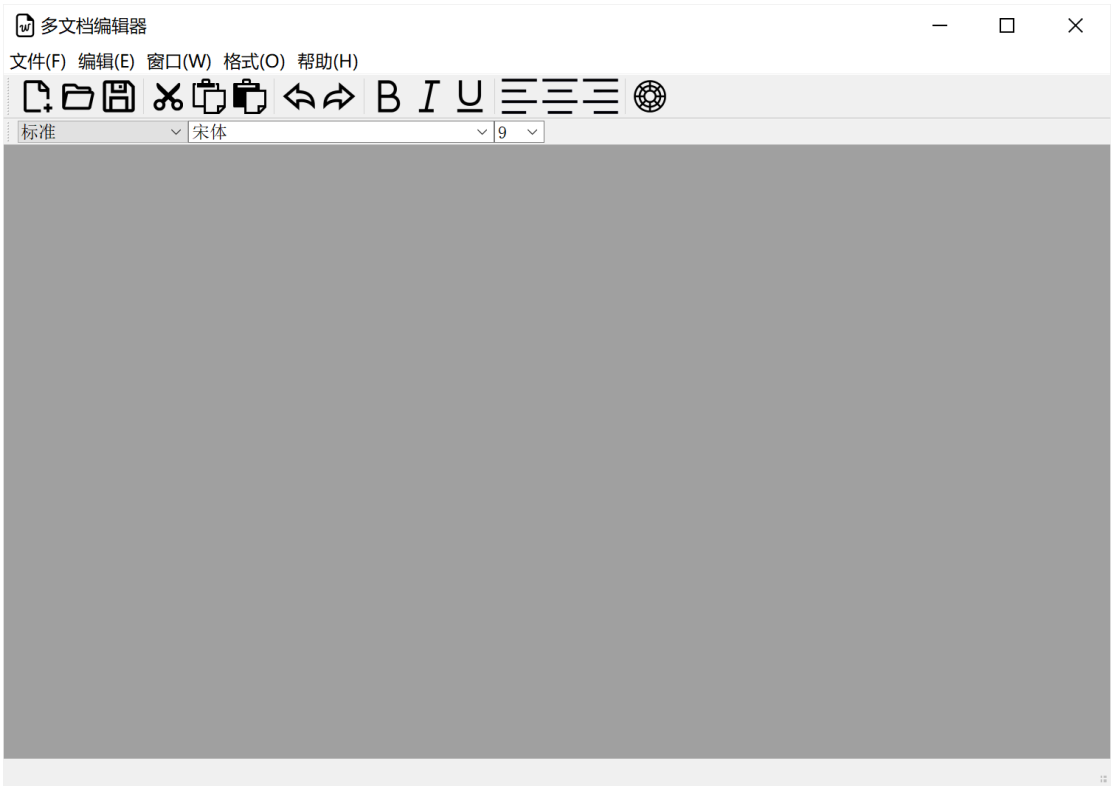


图 2.1 多文档编辑器

2.2 设计分析与算法流程

本程序按钮较少、菜单栏复杂，故采用 UI Designer 图形化编程与代码编程混合设计；

2.2.1 MDI 实现

作为多文档文本编辑器，我们需要以 QmdiArea 类为多文档区域，以

QtextEdit 类为子窗口部件，从而实现一个 MDI 应用程序。为此，我们要在头文件中添加 QmdiArea 类的声明并定义变量，在源文件的构造函数中编写如下代码：

```
1      mdiArea = new QMdiArea;
2      mdiArea->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
3      mdiArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);
4      setCentralWidget(mdiArea);
```

2.2.2 新建文件

设计思路：

① 设置窗口编号

② 设置文件未被保存过，”isUntitled=true;”

③ 保存文件路径，为 curFile 赋初值

④ 设置子窗口标题

主要代码如下：

```
1    void childwindow::newFile()
2    {
3        static int sequenceNumber=1;
4        isUntitled=true;
5        curFile=tr("文档 %1").arg(sequenceNumber++);
6        setWindowTitle(curFile+"[*]"+QString("——notepad--"));
7    }
```

```
1    void MainWindow::on_action_newFile_triggered()
2    {
3        childwindow *child=createChildWindow();
4        child->newFile();
5        child->show();
6        enabledText();
7    }
```

2.2.3 打开文件

设计思路：

① 打开指定的文件，并读取文件内容到编辑器；

② 设置当前文件 setCurrentFile(),该函数可以获取文件路径，完成文件和窗口状态的设置；

③ 关联文档内容改变信号到显示文档更改状态槽 documentWasModified()

主要代码如下：

```
1    bool childwindow::loadFile(const QString &fileName)
```

```
2   {
3       if (!fileName.isEmpty())
4       {
5           if (!QFile::exists(fileName))
6               return false;
7           QFile file(fileName);
8           if (!file.open(QFile::ReadOnly))
9               return false;
10
11          QByteArray data = file.readAll();
12          QTextCodec *codec = Qt::codecForHtml(data);
13          QString str = codec->toUnicode(data);
14          if (Qt::mightBeRichText(str)) {
15              this->setHtml(str);
16          } else {
17              str = QString::fromLocal8Bit(data);
18              this->setPlainText(str);
19          }
20          setCurrentFile(fileName);
21          connect(document(), SIGNAL(contentsChanged()),this,
22  SLOT(documentWasModified()));
23          return true;
24      }
25  }
```

```
1   void MainWindow::on_action_open_triggered()
2   {
3       QString fileName = QFileDialog::getOpenFileName(this, tr("打开
4   "),QString(), tr("HTML 文档 (*.htm *.html);;所有文件 (*.*)"));
5       if (!fileName.isEmpty()) {
6           QMdiSubWindow *existing = findChildWindow(fileName);
7           if (existing) {
8               mdiArea->setActiveSubWindow(existing);
9               return;
10          }
11          childwindow *child = createChildWindow();
12          if (child->loadFile(fileName)) {
13              statusBar()->showMessage(tr("文件已载入"), 2000);
14              child->show();
15          } else {
16              child->close();
17          }
18      }
19  }
```

2.2.4 保存文件

设计思路：

① 如果文件没有被保存过(用 isUntitled 判断)，执行“另存为”操作 saveAs() ；

② 否则直接保存文件 saveFile(),该函数先打开指定文件，然后将编辑器

的内容写入该文件，最后设置当前文件 setCurrentFile()。

主要代码如下：

```
bool childwindow::saveFile(QString fileName)
{
    if (!(fileName.endsWith(".htm", Qt::CaseInsensitive) ||
fileName.endsWith(".html", Qt::CaseInsensitive))) {
        fileName += ".html"; // 默认保存为 HTML 文档
    }
    QTextDocumentWriter writer(fileName);
    bool success = writer.write(this->document());
    if (success)
        setCurrentFile(fileName);
    return success;
}
```

```
bool childwindow::save()
{
    if(isUntitled)
        return saveAs();
    else
        return saveFile(curFile);
}
```

```
void MainWindow::on_action_save_triggered()
{
    if(activeChildWindow()&&activeChildWindow()->save())
        statusBar()->showMessage(tr("保存成功"),2000);
}
```

2.2.5 另存为文件

设计思路：

① 从“文件”对话框获取文件路径

② 如果路径不为空，则保存文件 saveFile()

主要代码如下：

```
bool childwindow::saveAs()
{
    QString fileName=QFileDialog::getSaveFileName(this,tr("另存为
"),curFile,tr("HTML 文档（*.htm *.html）;;所有文件（*.*）"));
    if (fileName.isEmpty())
        return false;
    return saveFile(fileName);
}
```

```
void MainWindow::on_action_saveAs_triggered()
{
    if(activeChildWindow()&&activeChildWindow()->saveAs())
        statusBar()->showMessage(tr("保存成功"),2000);
}
```

2.2.6 设置字体、字号、加粗等操作

设计思路：

① 设计字体格式化函数 fontFormat()，用于接收各功能按钮的参数；

② 设计字体改变函数、字号改变函数、加粗函数等用于向 fontFormat()发送参数；

主要代码如下：

```
void childwindow::fontFormat(const QTextCharFormat &format)
{
    QTextCursor cursor = this->textCursor();
    if (!cursor.hasSelection())
        cursor.select(QTextCursor::WordUnderCursor);
    cursor.mergeCharFormat(format);
    this->mergeCurrentCharFormat(format);
}
```

```
 void MainWindow::textFamily(const QString &f)
 {
     QTextCharFormat fmt;
     fmt.setFontFamily(f);
     if(activeChildWindow())
         activeChildWindow()->fontFormat(fmt);
 }
```

```
  void MainWindow::textSize(const QString &p)
  {
      qreal pointSize = p.toFloat();
      if (p.toFloat() > 0) {
          QTextCharFormat fmt;
          fmt.setFontPointSize(pointSize);
          if(activeChildWindow())
              activeChildWindow()->fontFormat(fmt);
      }
  }
```

```
void MainWindow::on_action_bold_triggered()
{
    QTextCharFormat fmt;
    fmt.setFontWeight(isBold?QFont::Bold:QFont::Normal);
    isBold=isBold?false:true;
    if(activeChildWindow())
        activeChildWindow()->fontFormat(fmt);
}
```

```
 void MainWindow::on_action_italic_triggered()
 {
     QTextCharFormat fmt;
     fmt.setFontItalic(isItalic);
     isItalic=isItalic?false:true;
     if(activeChildWindow())
```

```
7           activeChildWindow()->fontFormat(fmt);
8    }
```

```
1    void MainWindow::on_action_underline_triggered()
2    {
3        QTextCharFormat fmt;
4        fmt.setFontUnderline(isUnderline);
5        isUnderline=isUnderline?false:true;
6        if(activeChildWindow())
7            activeChildWindow()->fontFormat(fmt);
8    }
```

2.2.7 设置窗体排列方式

设计思路：

① 调用 QmdiArea 的 tileSubWindows()和 cascadeSubWindows()函数

主要代码如下：

```
1    void MainWindow::on_action_windowLayer_triggered()
2    {
3        mdiArea->cascadeSubWindows();
4    }
```

```
1    void MainWindow::on_action_tileHorizontal_triggered()
2    {
3        mdiArea->tileSubWindows();
4    }
```

2.3 界面设计

2.3.1 UI Designer 部分

如图 2.2

图 2.2 多文档编辑器主界面（UI 设计师部分）

2.3.2 代码部分

设计思路：

① 设计 createToolBars()函数，用于创建工具栏；

② 在程序启动时调用 createToolBars()函数。

主要代码如下：

```
1    void MainWindow::createToolBars()
2    {
3        addToolBarBreak(Qt::TopToolBarArea);
4        comboToolBar = addToolBar(tr("组合选择"));
5        comboStyle = new QComboBox();
6        comboToolBar->addWidget(comboStyle);//加入 combox
7        comboStyle->addItem("标准");
8        comboStyle->addItem("项目符号 (●)");
9        comboStyle->addItem("项目符号 (○)");
10       comboStyle->addItem("项目符号 (■)");
11       comboStyle->addItem("编号 (1.2.3.)");
12       comboStyle->addItem("编号 (a.b.c.)");
13       comboStyle->addItem("编号 (A.B.C.)");
14       comboStyle->addItem("编号 (ⅰ.ⅱ.ⅲ.)");
15       comboStyle->addItem("编号 (Ⅰ.Ⅱ.Ⅲ.)");
16       comboStyle->setStatusTip("段落加标号或编号");
17       connect(comboStyle, SIGNAL(activated(int)), this,
     SLOT(textStyle(int)));
```

```
19
20      comboFont = new QFontComboBox();
21      comboToolBar->addWidget(comboFont);
22      comboFont->setStatusTip("更改字体");
23      connect(comboFont, SIGNAL(activated(QString)), this,
    SLOT(textFamily(QString)));
25
26      comboSize = new QComboBox();
27      comboToolBar->addWidget(comboSize);
28      comboSize->setEditable(true);
29      comboSize->setStatusTip("更改字号");
30
31      QFontDatabase db;
32      foreach(int size, db.standardSizes())
33          comboSize->addItem(QString::number(size));
34
35      connect(comboSize, SIGNAL(activated(QString)), this,
    SLOT(textSize(QString)));
37
38  comboSize->setCurrentIndex(comboSize->findText(QString::number(QAppli
    cation::font().pointSize())));
40  }
```

2.3.2 按键绑定

图 2.3 给出按键名称。

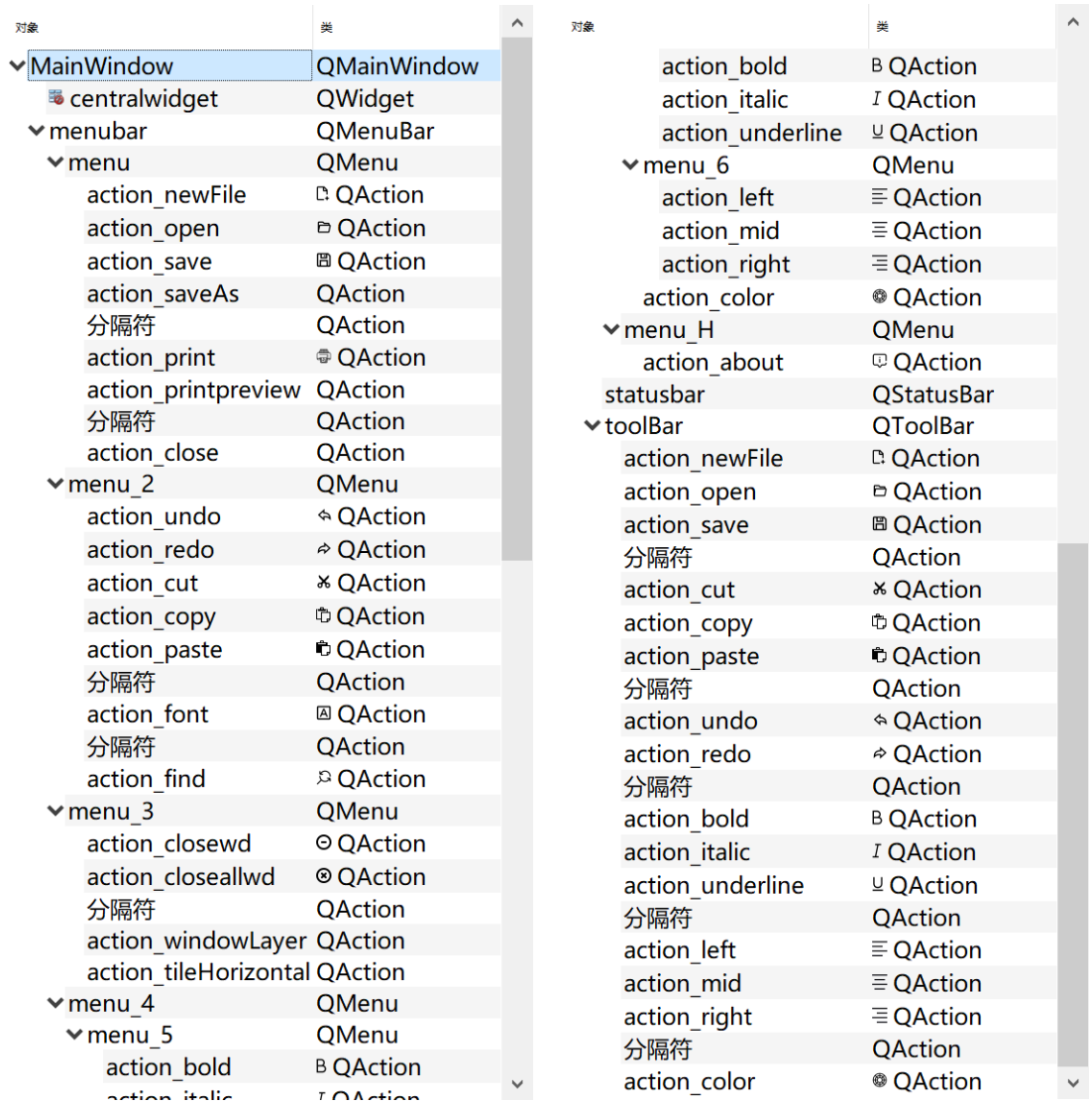


图 2.3 主界面的按键绑定

2.4 关键类图

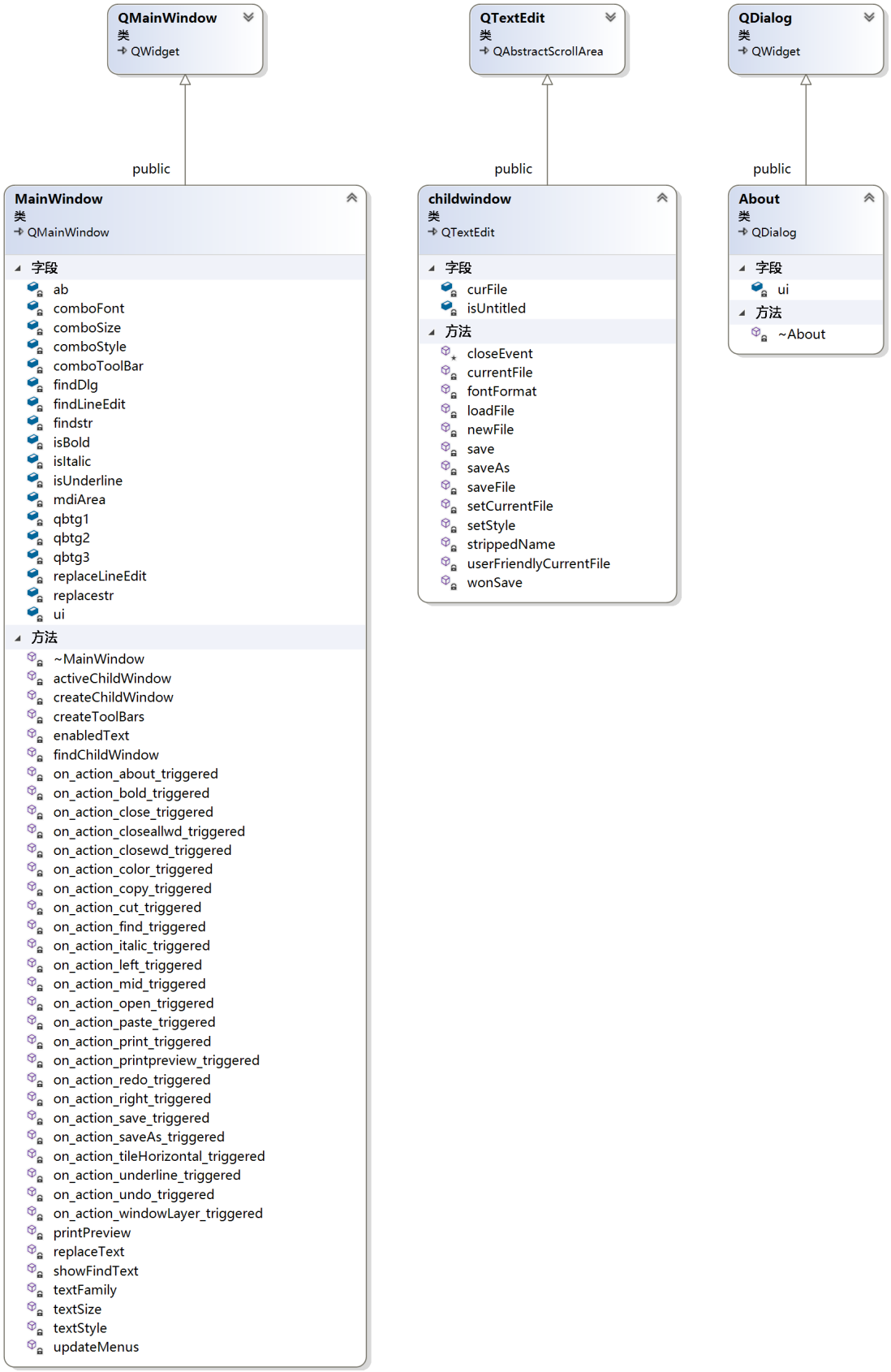


图 2.4 程序类图

3 系统实现（运行调试）

一些集成操作（字体、字号、居中、颜色、段落标号）示例见图 2.5；

打印预览操作示例见图 2.6。

图 2.5 示例操作 1

图 2.6 示例操作 2

4 系统扩展

4.1 简单操作

剪切、复制、粘贴、撤销、重做、打印等功能均通过 QT 库函数实现，详见图 2.6

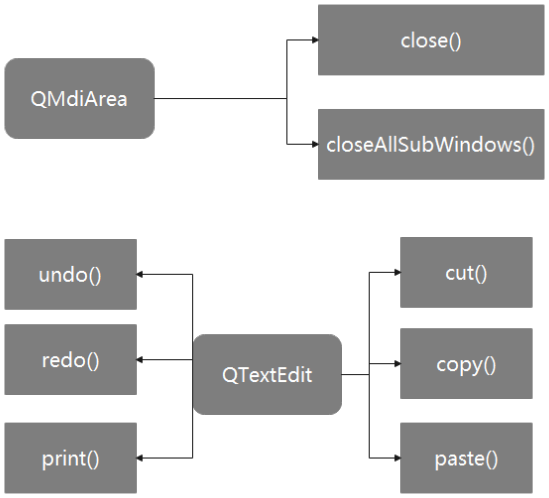


图 2.6 简单操作的函数调用

4.2 查找、替换操作

设计思路：

① 使用代码创建 findDlg 窗口，其本身为主窗口的一个成员，故能进行消息传递；

② 利用成员特性向主窗口传递 LineEdit 中的信息并进行查找、替换操作。

代码如下：

```
1    {
2        findDlg = new QDialog(this);
3        findDlg->setWindowTitle(tr("查找&替换"));
4
5        findLineEdit = new QLineEdit(findDlg);
6        replaceLineEdit = new QLineEdit(findDlg);
7        QPushButton *btn= new QPushButton(tr("查找下一个"), findDlg);
8        QPushButton *btn2= new QPushButton(tr("替换"), findDlg);
9
10       QVBoxLayout *layout= new QVBoxLayout(findDlg);
11       QHBoxLayout *cldlayout0=new QHBoxLayout();
12       QHBoxLayout *cldlayout1=new QHBoxLayout();
13       QHBoxLayout *cldlayout2=new QHBoxLayout();
14       qbtg1 = new QButtonGroup(this);
15       qbtg2 = new QButtonGroup(this);
16       qbtg3 = new QButtonGroup(this);
17
18       QRadioButton *Button01 = new QRadioButton(this);
19       Button01->setText(tr("非大小写匹配"));
20       cldlayout0->addWidget(Button01);
21       QRadioButton *Button02 = new QRadioButton(this);
22       Button02->setText(tr("大小写匹配"));
23       cldlayout0->addWidget(Button02);
24
```

```
25        qbtg1->addButton(Button01);
26        qbtg1->addButton(Button02);
27        qbtg1->setId(Button01, 0);
28        qbtg1->setId(Button02, 1);
29        Button01->setChecked(true);
30
31        QRadioButton *Button11 = new QRadioButton(this);
32        Button11->setText(tr("单字匹配"));
33        cldlayout1->addWidget(Button11);
34        QRadioButton *Button12 = new QRadioButton(this);
35        Button12->setText(tr("全字匹配"));
36        cldlayout1->addWidget(Button12);
37
38        qbtg2->addButton(Button11);
39        qbtg2->addButton(Button12);
40        qbtg2->setId(Button11, 0);
41        qbtg2->setId(Button12, 1);
42        Button11->setChecked(true);
43
44        QRadioButton *Button21 = new QRadioButton(this);
45        Button21->setText(tr("单个替换"));
46        cldlayout2->addWidget(Button21);
47        QRadioButton *Button22 = new QRadioButton(this);
48        Button22->setText(tr("全部替换"));
49        cldlayout2->addWidget(Button22);
50
51        qbtg3->addButton(Button21);
52        qbtg3->addButton(Button22);
53        qbtg3->setId(Button21, 0);
54        qbtg3->setId(Button22, 1);
55        Button21->setChecked(true);
56
57        layout->addLayout(cldlayout0);
58        layout->addLayout(cldlayout1);
59        layout->addWidget(findLineEdit);
60        layout->addWidget(btn);
61        layout->addLayout(cldlayout2);
62        layout->addWidget(replaceLineEdit);
63        layout->addWidget(btn2);
64
65        connect(btn, SIGNAL(clicked()), this, SLOT(showFindText()));
66        connect(btn2, SIGNAL(clicked()), this, SLOT(replaceText()));
67    }
```

查找操作共有四种情况，这里采用双重 switch 语句去判断，流程图见图 2.7。

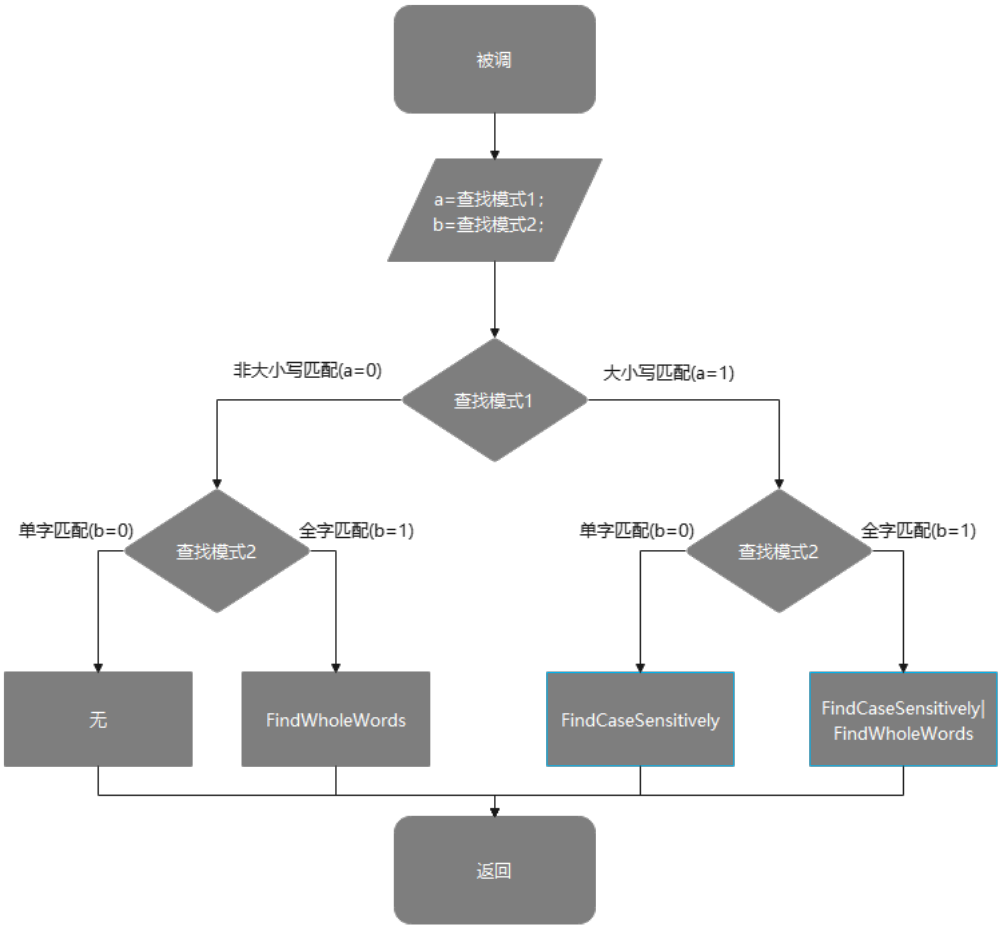


图 2.7 查找操作流程图

代码如下：

```
1    void MainWindow::showFindText()
2    {
3        if(!findstr)
4        {
5            activeChildWindow()->moveCursor(QTextCursor::Start);
6            findstr=true;
7        }
8        int a = qbtg1->checkedId();
9        int b = qbtg2->checkedId();
10       QString str = findLineEdit->text();
11       if(activeChildWindow())
12           {
13           switch (a)
14           {
15           case 0:
16           {
17               switch (b)
18               {
19               case 0:
20               {
21                   qDebug()<<"00";
```

```
22                 if (!activeChildWindow()->find(str))
23                 {
24                    QMessageBox::warning(this, tr("查找"),
25                           tr("找不到%1").arg(str));
26                 }
27                 break;
28             }
29             case 1:
30             {
31                 qDebug()<<"01";
32                 if
33    (!activeChildWindow()->find(str,QTextDocument::FindWholeWords))
34                 {
35                    QMessageBox::warning(this, tr("查找"),
36                           tr("找不到%1").arg(str));
37                 }
38                 break;
39             }
40             }
41             break;
42         }
43         case 1:
44             {
45                 switch (b)
46                 {
47                 case 0:
48                 {
49                     qDebug()<<"10";
50                     if
51    (!activeChildWindow()->find(str,QTextDocument::FindCaseSensitively))
52                     {
53                        QMessageBox::warning(this, tr("查找"),
54                               tr("找不到%1").arg(str));
55                     }
56                     break;
57                 }
58                 case 1:
59                 {
60                     qDebug()<<"11";
61                     if
62    (!activeChildWindow()->find(str,QTextDocument::FindCaseSensitively|QT
63    extDocument::FindWholeWords))
64                     {
65                        QMessageBox::warning(this, tr("查找"),
66                               tr("找不到%1").arg(str));
67                     }
68                     break;
69                 }
70                 }
71              break;
72             }
73         }
74
75         }
76    }
```

对于替换操作，流程图见图 2.8。

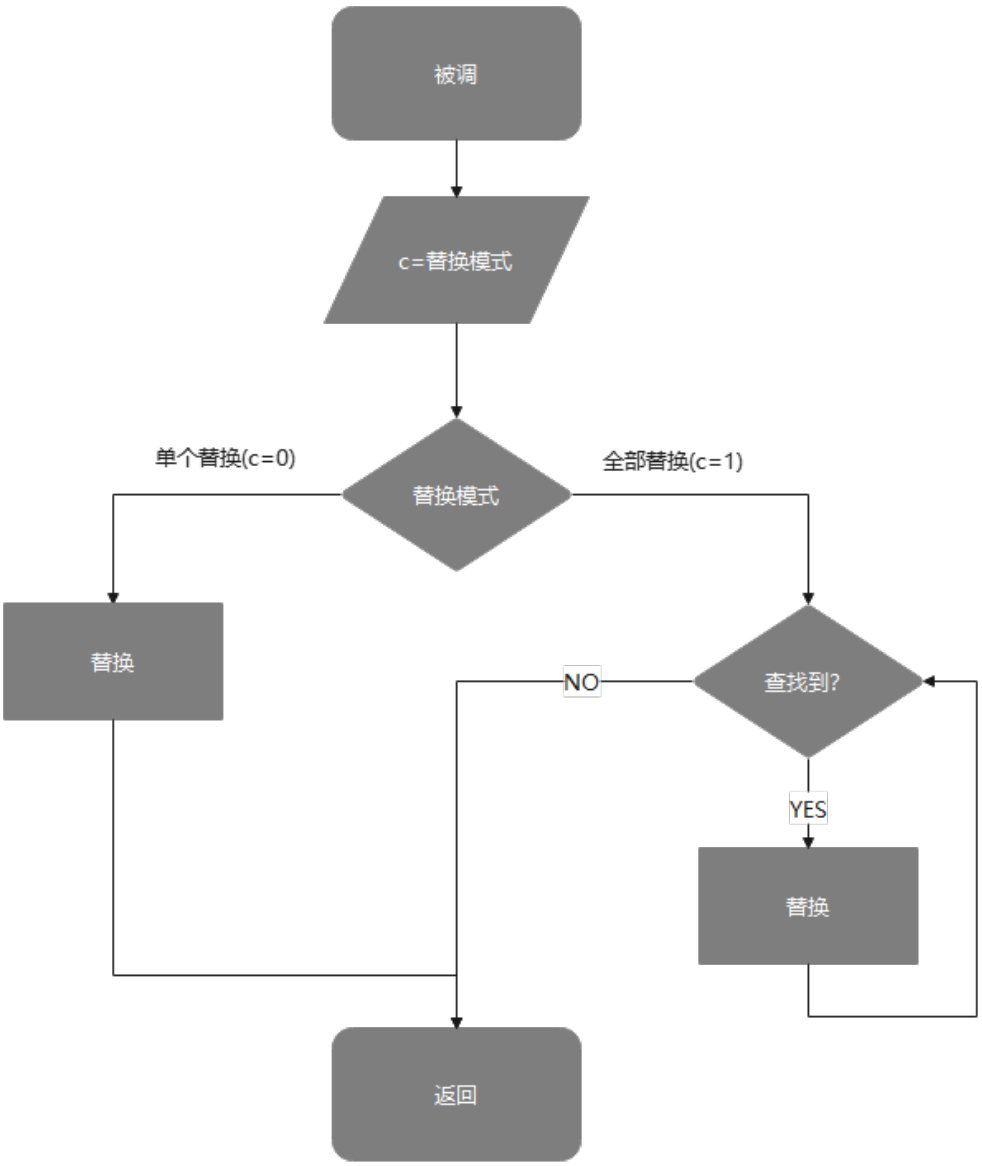


图 2.8 替换操作的流程图

其代码如下：

```
1   void MainWindow::replaceText()
2   {
3       if(!replacestr)
4       {
5           activeChildWindow()->moveCursor(QTextCursor::Start);
6           replacestr=true;
7       }
8       int c = qbtg3->checkedId();
9       QString str = findLineEdit->text();
10      QString str2=replaceLineEdit->text();
11
```

```
12      if(activeChildWindow())
13          {
14          switch (c) {
15          case 0:
16          {
17              if (!activeChildWindow()->find(str))
18              {
19                 QMessageBox::warning(this, tr("替换"),
20                        tr("不存在%1").arg(str));
21              }
22              else
23              {
24                  activeChildWindow()->insertPlainText(str2);
25              }
26              break;
27          }
28          case 1:
29          {
30              while(activeChildWindow()->find(str))
31                  activeChildWindow()->insertPlainText(str2);
32              break;
33          }
34          }
35      }
36  }
```

5. 总结

本项目通过较为传统的方式（代码编程）建立了一个 MDI 程序，接触到了大量 QT 库，如 QmdiArea、QtextCodec 等，掌握了更多 QT 框架下 C++的操作，并学习了包括继承、图形化、消息传递等面向对象编程的知识。

实验三 2048 游戏

1 系统概述

本章原定开发内容为拼图游戏，因趣味性和个人兴趣，该项目实际开发为 2048 游戏。为了获得更好的表现力，采用 QML 语言进行图形界面的编程。项目采用二维数组作为存储形式，重难点在于算法部分。

注：游戏算法不易优化，故本项目借鉴了很多帖子，但程序是由本人在读懂各贴算法后从不同语言改造、对不同功能拼接而成。以下是引用内容的超链接：

GitHub - gabrielecirulli/2048

小游戏 2048 最佳算法怎么实现？思路全解析！

vue 实现 2048 小游戏功能思路详解

GitHub - sogilis/qt2048

QML 自定义 Button

2 系统设计

2.1 设计目标

本项目目标是设计开发一个支持键盘、鼠标操作的游戏，游戏可以对内置磁贴移动，在碰撞到面板边缘时停止并合并相同磁贴（数值相加），达到2048即为获胜，未达到并无空磁贴并不可合并时，游戏失败。为了使游戏重新开始，还需要一个多用重启按钮。

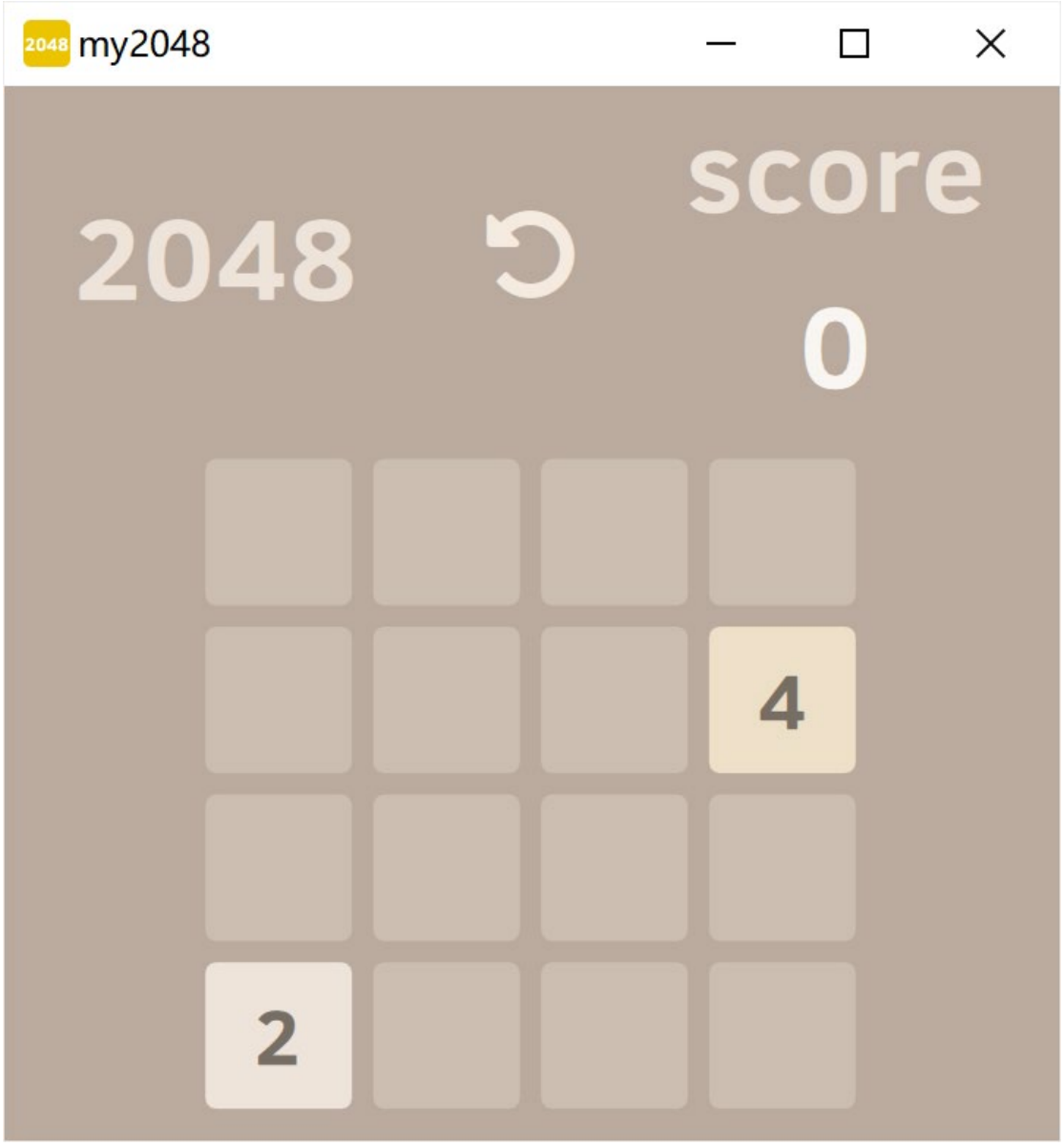


图 3.1 2048 游戏

2.2 设计分析与算法流程

游戏的设计思路见图 3.2。

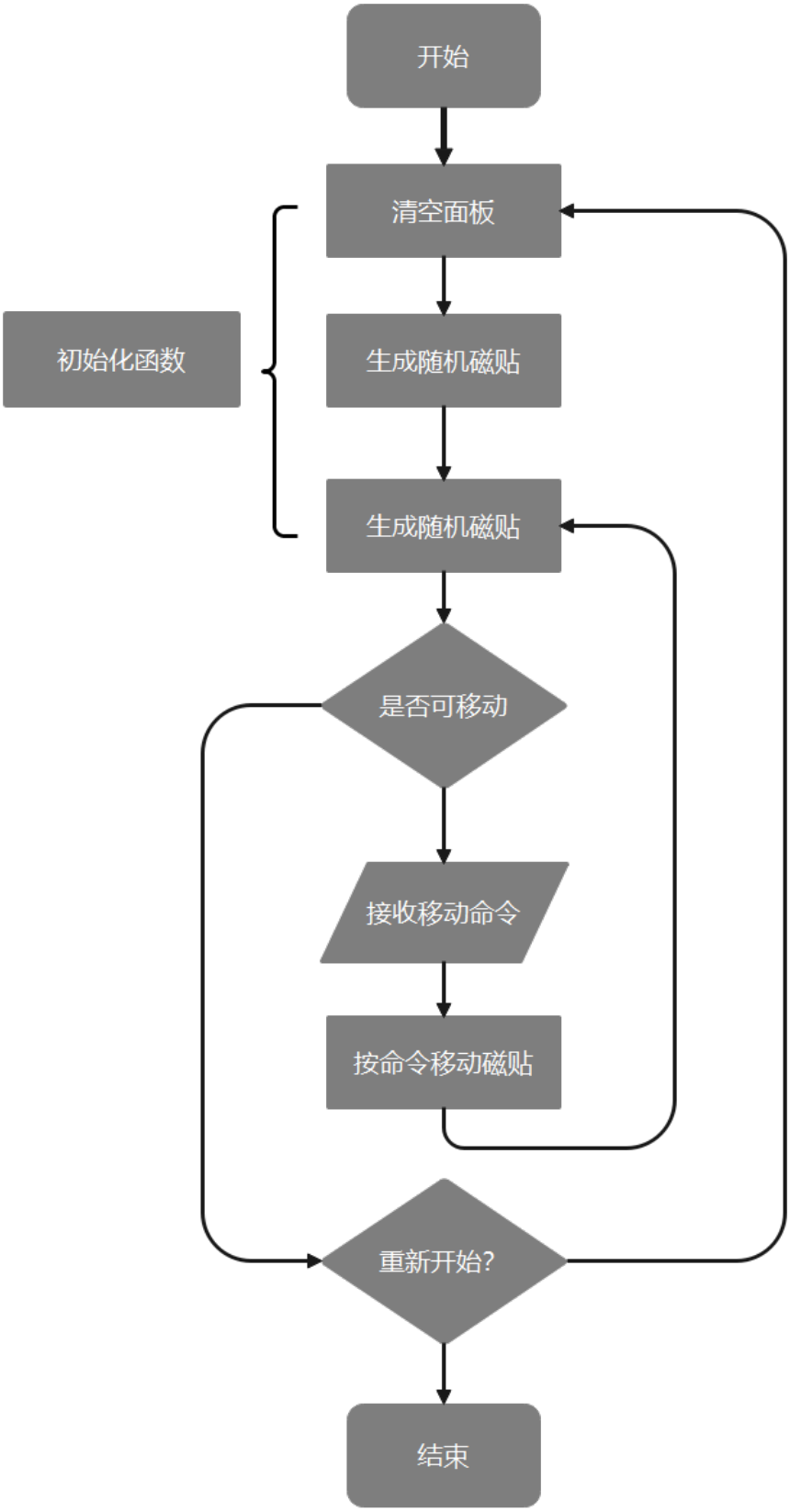


图 3.2 2048 设计基本思路

下面我们自上而下地分析游戏算法

首先，为了完成游戏，我们需定义几个常量：

kSize 游戏面板的尺寸，我们完成的是经典 2048，故 kSize=4;

kValueOfLastTile 游戏胜利需要的最新磁贴值，同理，应 kValueOfLastTile=2048;但为了测试胜利方便，我们可以调整它的大小。

2.2.1 初始化函数

初始化函数应分为两个：

① 初始化随机数函数；

② 初始化游戏函数；

这里思路比较简单，流程图反而繁琐，下面是这两个函数的代码：

```
1    void Panel::initRand() {
2        auto time = QTime::currentTime();
3        qsrand((uint) time.msec());
4    }
```

```
1    void Panel::initGame() {
2        memset(panel_, 0, sizeof(panel_));
3        score_ = 0;
4        higherTile_ = 0;
5        addRandomTile();
6        addRandomTile();
7    }
```

2.2.2 增加磁贴函数

这里也可以用一句话说明：首先创建一个 len*2 的坐标数组（len 为空磁贴数），然后用随机数%len 获取某行的坐标，在此处调用新磁贴求值函数，将其赋给二维数组的坐标所在磁贴处。

代码如下：

```
1     void Panel::addRandomTile() {
2         index_t x, y;
3         value_t list[kSize * kSize][2];
4         value_t len = 0;
5         value_t r;
6
7         for(x = 0; x < kSize; x++) {//生成坐标数组
8             for(y = 0; y < kSize; y++) {
9                 if(panel_[x][y] == 0) {
10                    list[len][0] = x;
11                    list[len][1] = y;
12                    len++;
13                }
14            }
15        }
16
17        if (len > 0) {
18            r = qrand() % len;
19            x = list[r][0];
```

```
            y = list[r][1];
            panel_[x][y] = nextTileValue();
        }
    }
```

```
value_t Panel::nextTileValue() const {
    return ((qrand() % 10) / 9 + 1) * 2;
}
```

这个函数中:

%10 代表取 0-9,

/9 代表取 0 或 1,

+1 代表取 1 或 2,

*2 代表取 2 或 4.

2.2.4 是否可移动

这里通过 gameEnd()函数来判断是否可移动（游戏是否结束），流程图见图 3.3

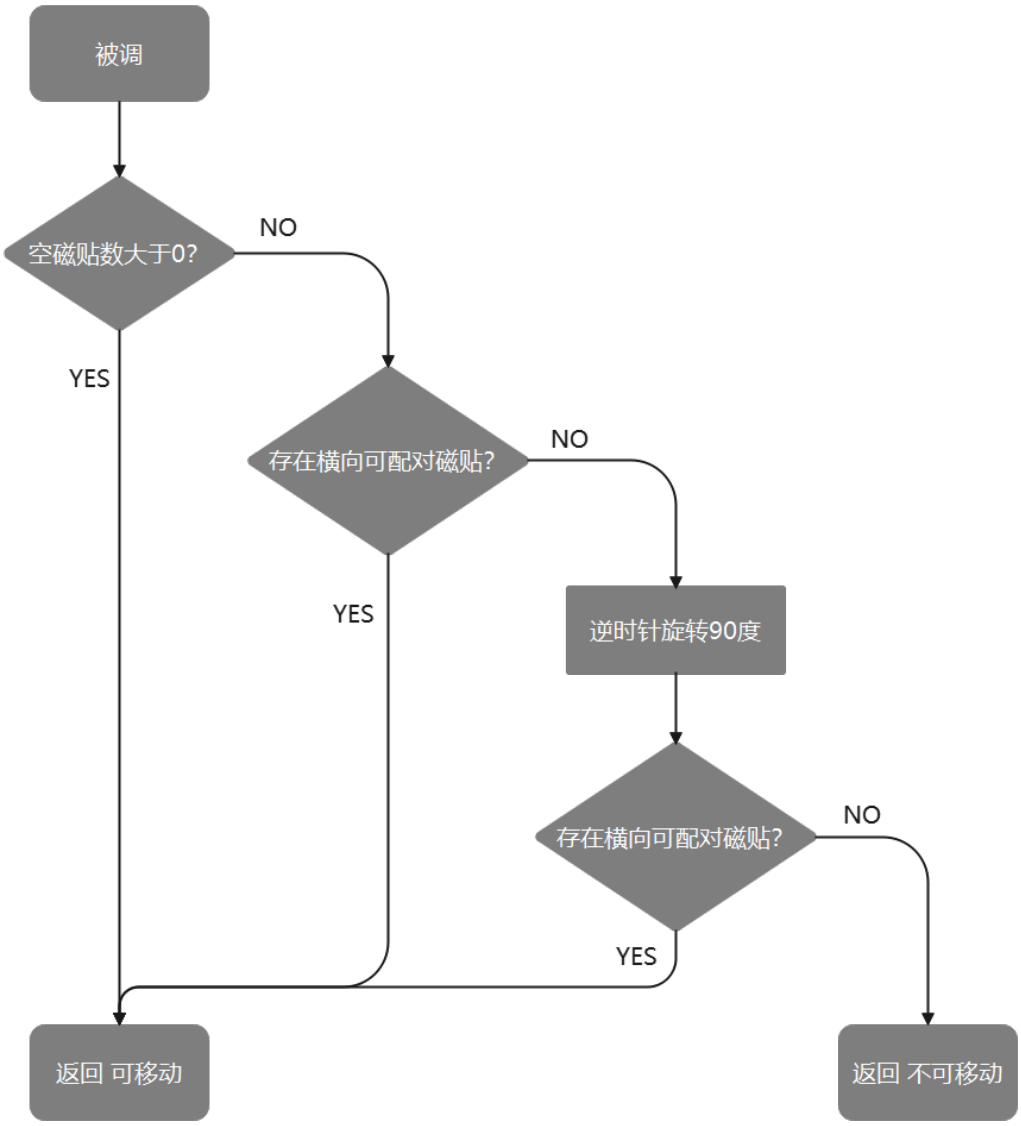


图 3.3 gameEnd()函数简化流程图

代码如下：

```
1         bool Panel::gameEnded() {
2             bool ended = true;
3             if (countEmpty() > 0) {
4                 return false;
5             }
6             if (findPairDown()) {
7                 return false;
8             }
9             rotateToRight();
10            if (findPairDown()) {
11                ended = false;
12            }
13            rotateToRight();
14            rotateToRight();
15            rotateToRight();
```

```
16            return ended;
17        }
```

这里注意纵向匹配（旋转后横向匹配）后要将面板旋转回原方向。

此处涉及到了空磁贴计数函数 countEmpty()、横向匹配函数 findPairDown()和旋转函数 rotateToRight()，下面将一一介绍。

2.2.5 空磁贴计数函数 countEmpty()

原理简单，代码如下：

```
1       value_t Panel::countEmpty() const {
2           index_t x, y;
3           value_t count = 0;
4           for (x = 0; x < kSize; x++) {
5               for (y = 0; y < kSize; y++) {
6                   if (panel_[x][y] == 0) {
7                       count++;
8                   }
9               }
10          }
11          return count;
12      }
```

2.2.6 横向匹配函数 findPairDown()

原理简单，代码如下：

```
1     bool Panel::findPairDown() const {
2         bool success = false;
3         index_t x, y;
4         for (x = 0; x < kSize; x++) {
5             for (y = 0; y < kSize - 1; y++) {
6                 if (panel_[x][y] == panel_[x][y+1]) {
7                     return true;
8                 }
9             }
10        }
11
12        return success;
13    }
```

2.2.7 旋转函数 rotateToRight()

因该函数可统一操作，故在本项目中频繁使用。

对二维数组的操作总可分为横向和纵向，如上移下移为纵向，左移右移为横向。使用此函数便可以不再区分方向，统一向上移即可。

原理见下图 3.4（只标出了内外两条方向，实际上同颜色磁贴都要逆时针替换一次）：

图 3.4 rotateToRight()原理图

代码如下：

```
1   void Panel::rotateToRight() {//逆时针旋转 90 度
2       index_t i, j;
3       value_t tmp;
4       for (i = 0; i < kSize / 2; i++) {
5           for (j = i; j < kSize - i - 1; j++) {
6               tmp                       = panel_[i][j];
7               panel_[i][j]                = panel_[j][kSize-i-1];
8               panel_[j][kSize-i-1]        = panel_[kSize-i-1][kSize-j-1];
9               panel_[kSize-i-1][kSize-j-1] = panel_[kSize-j-1][i];
10              panel_[kSize-j-1][i]        = tmp;
11          }
12      }
13  }
14
```

2.2.8 移动磁贴命令

原理在上一条中已详述，即使用此函数，将面板旋转，始终向上移动（相对性方向不变），移动后将面板旋转回原方向，原理如图 3.5。

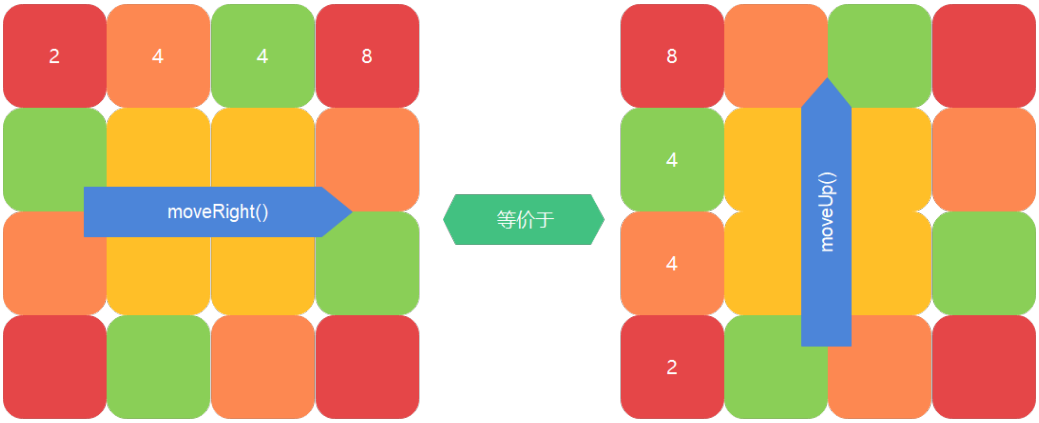


图 3.5 move()原理图

其代码如下：

```
bool Panel::move(Directions direction) {
    bool success = false;

    int i;
    for(i = Up; i < direction; i++) {//转过去
        rotateToRight();
    }
    success = moveTilesUpwards();//始终对转好的矩阵向上移动
    for(i = direction; i <= Right; i++) {//转回来
        rotateToRight();
    }
    if (success) {
        addRandomTile();
        emit panelChangedAfterMovement();
    }

    if(win() || gameEnded()) {
        emit gameStatusChanged();
        return false;
    }

    return success;
}
```

这里涉及到了 moveTileUpwards()函数，下面将介绍

2.2.9 moveTileUpwards()

其原理为：将面板矩阵分为 4 列，循环调用 slide()进行向上滑动，代码如下：

```
bool Panel::moveTilesUpwards() {
    if (win() || gameEnded()) {
        return false;
    }
    bool success = false;
    index_t x;
    for (x = 0; x < kSize; x++) {
```

```
            success |= slide(panel_[x]);
        }
        return success;
    }
```

这里涉及到了 slide()函数，下面介绍。

2.2.10 slide()

该函数通过设置当前位(pos)和目标位(target)进行合并(merge)检测，若 pos 与 target 相等，代表可以合并，slide()函数将把 pos 位的值加到 target 上并清空 pos 位。停止标志(stop)用于判断该列能否合并，若成功合并，则更新分数，代码如下：

```
bool Panel::slide(value_t array[kSize]) {
    bool success = false;
    index_t pos, target, stop = 0;
    value_t mergeValue;

    for (pos = 0; pos < kSize; pos++) {
        if (array[pos] != 0) {
            target = findTarget(array, pos, stop);
            if (target != pos) {
                // need to move (0) or merge (!0)
                if (array[target] != 0) {
                    stop = target + 1; // don't cascade merges
                    mergeValue = array[target] + array[pos];
                    score_ += mergeValue;
                    emit scoreChanged();
                    higherTile_ =
higherTile_>mergeValue?higherTile_:mergeValue;//max(higherTile_,
mergeValue);
                }
                array[target] += array[pos]; // merge or move (target is
0)
                array[pos] = 0;
                success = true;
            }
        }
    }

    return success;
}
```

这里涉及到了 findTarget()函数，下面介绍

2.2.11 findTarget()

该函数用于寻找可以合并的 target 位置，一个一重循环，若 pos 与 target 相等，代表可以合并，将此 target 位置返回给 slide()，若不能合并，则返回下一个位置。

```
index_t Panel::findTarget(value_t array[kSize], const index_t pos,
const index_t stop) const {
    index_t target;
    if (pos == 0) {
        return pos;
```

```
    }
    for (target = pos - 1; target >= 0; target--) {
        if (array[target] != 0) {
            if (array[target] != array[pos]) {
                // unable to merge or move
                return target + 1;
            }
            // merge is possible
            return target;
        }
        else {
            if (target == stop) {
                return target;
            }
        }
    }
    return pos;
}
```

至此，2048 算法部分完成。

2.3 界面设计

为尽量达到原作者 JavaScript 的表现水准，这里采用 QML 语言进行图形界面的编程，其框架如下（仅保留主面板，代码不全无法高亮）。

```
Window {
    Window{
    Rectangle {
        anchors.fill: parent
        ColumnLayout {
            anchors.fill: parent
            Text {
            }
            }
    }
    visible: false
    }
    Rectangle {
         }
        }
        ColumnLayout {
            anchors.fill: parent
            RowLayout {
                Text {
                    }
                }
                UserButton{
                    }
                }
                ColumnLayout {
                Text {
                }
                Text {
                }

                }

```

```
33                }
34            Rectangle {
35                id: mainPanel
36                Layout.fillHeight: true
37                Layout.fillWidth: true
38
39                function step() {
40                    return Math.min(mainPanel.width, mainPanel.height) /
41  33;
42                }
43
44                width: 330
45                height: 330
46                color: "#baaa9e"
47                MouseArea {
48                    }
49                        }
50                    }
51                }
52
53                Grid {
54                    Repeater {
55                        }
56                    }
57                }
58            }
59        }
60    }
61
62
63  }
```

2.4 关键类图

见图 3.6

其中 PanleMod 类是给 QML 调用的接口

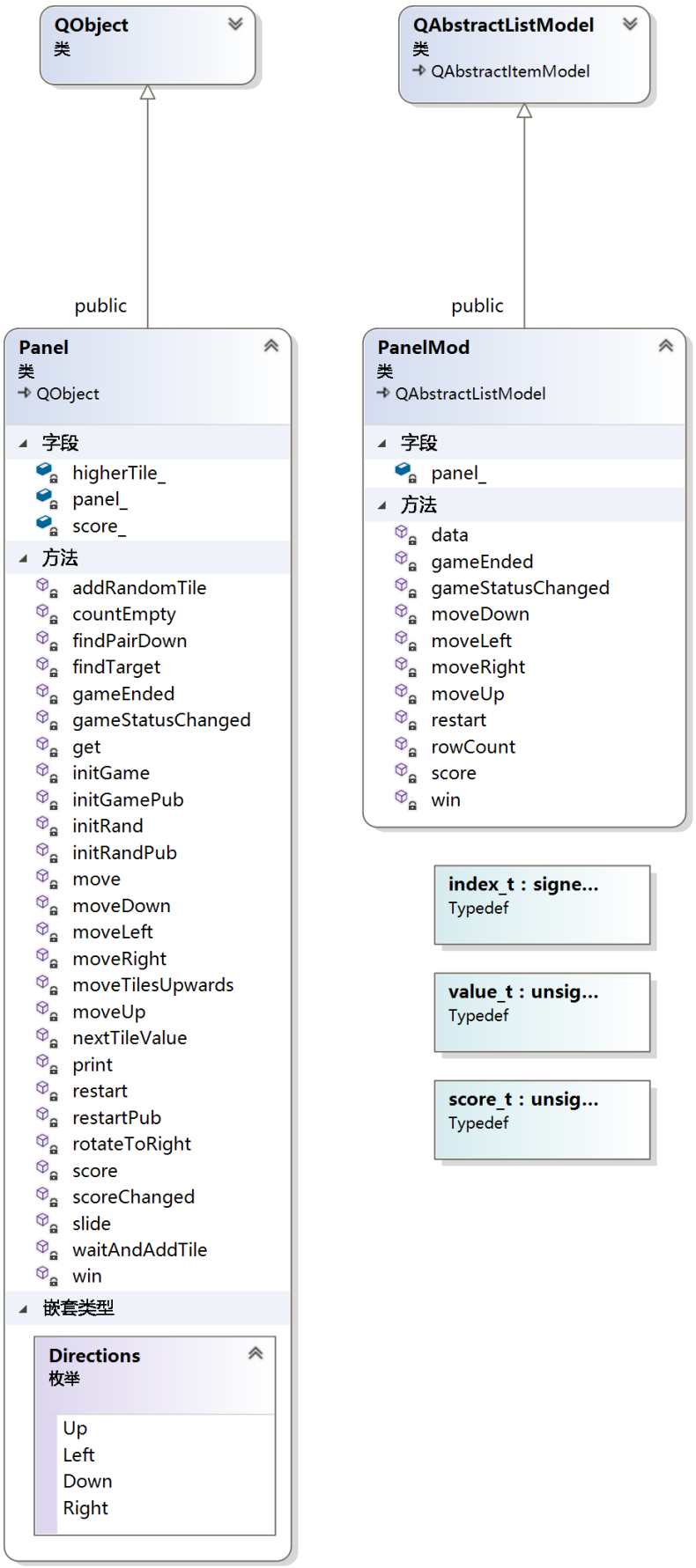


图 3.6 2048 关键类图

3 系统实现（运行调试）

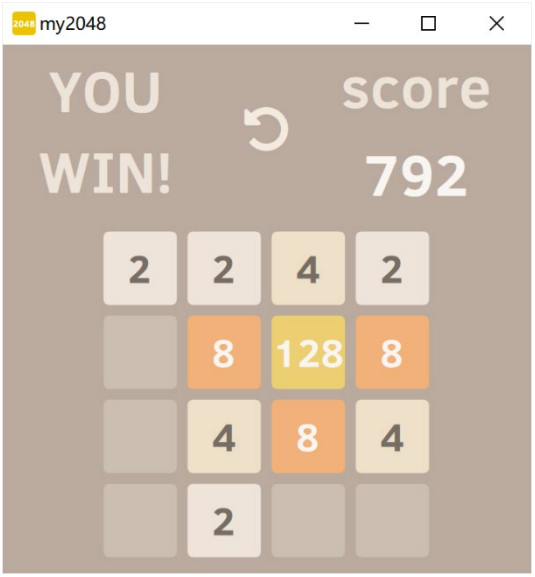


图 3.7 操作示例 1（胜利条件 128）

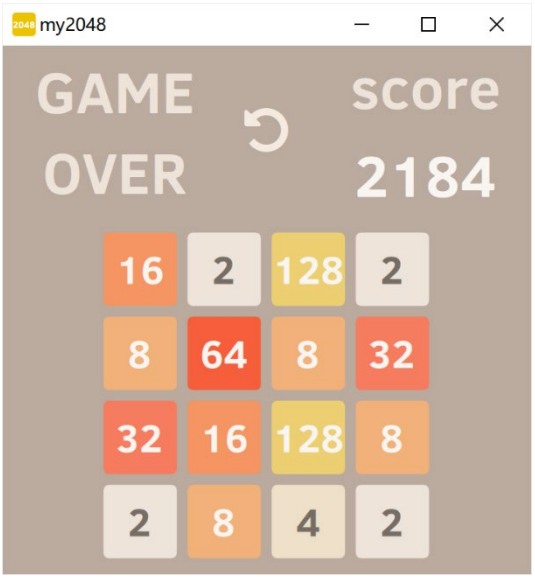


图 3.8 操作示例 2（胜利条件 2048）

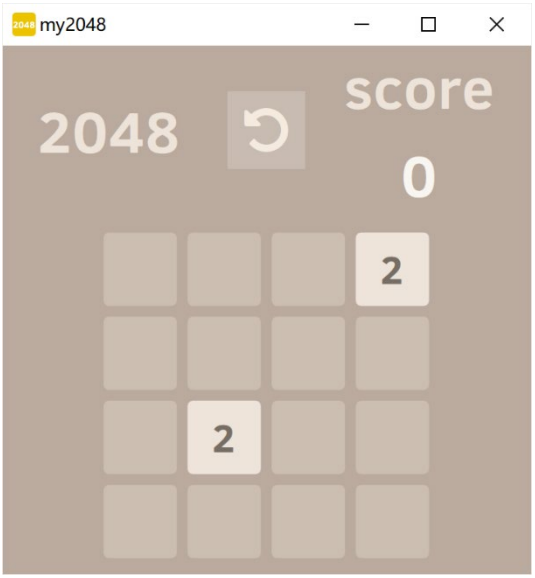


图 3.9 操作示例 3

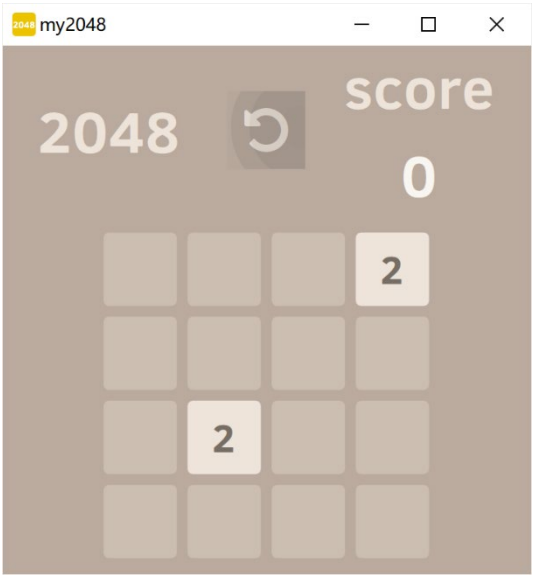


图 3.10 操作示例 4

4 系统扩展

这两段代码是从网络上复制的，我认为会用就可以了，具体原理的研究因本人能力有限未给出，这里只贴出代码。

4.1 鼠标拖动

代码如下：

```
1   function getGesture(startX, startY, endX, endY, areaWidth, areaHeight)
2   {
3           var deltaX = endX - startX;
4           var right = deltaX > 0;
```

```
5           var moveX = Math.abs(deltaX);
6
7           var deltaY = endY - startY;
8           var down = deltaY > 0;
9           var moveY = Math.abs(deltaY);
10
11          var minimumFactor = 0.25;
12
13          var relativeHorizontal = moveX / areaWidth;
14          var relativeVertical = moveY / areaHeight;
15
16          if (relativeHorizontal < minimumFactor &&
17                  relativeVertical < minimumFactor) {
18              return;
19          }
20
21          var horizontal = relativeHorizontal > relativeVertical;
22
23          if (horizontal) {
24              if (right) {
25                  return gesture_swipeRight;
26              }
27              return gesture_swipeLeft;
28          }
29          if (down) {
30              return gesture_swipeDown;
31          }
32          return gesture_swipeUp;
33      }
```

```
1   MouseArea {
2                   id: mouseArea
3                   anchors.fill: parent
4
5                   property int startX: 0
6                   property int startY: 0
7
8                   onPressed: {
9                       startX = mouseX;
10                      startY = mouseY;
11                  }
12
13                  onReleased: {
14                      var gesture = getGesture(startX, startY,
15                                               mouseX, mouseY,
16                                               mouseArea.width,
17  mouseArea.height);
18                      switch(gesture) {
19                      case gesture_swipeUp:
20                          panel.moveUp();
21                          break;
22                      case gesture_swipeRight:
23                          panel.moveRight();
24                          break;
25                      case gesture_swipeDown:
26                          panel.moveDown();
```

```
                        break;
                    case gesture_swipeLeft:
                        panel.moveLeft();
                        break;
                    }
                }
            }
```

4.2 水滴效果按钮

代码如下：

```
UserButton{
                    width: parent.height * 0.5 - 5
                    height: parent.height * 0.5 - 5
                    anchors.horizontalCenter: parent.horizontalCenter
                    icon: "\uf0e2"
                    backColor:"#baaa9e"
                    //toolTipStr:"重开"
                    family: "FontAwesome" //这里使用的是 FontAwesome 字体库
                    btnColor: "#f5ebe0"
                    flat: true
                    onClicked:{
                        console.log("Button was clicked");
                        panel.restart();
                        }
                    onDoubleClicked: {
                        console.log("Button was doubleclicked")
                    }
                    onPressAndHold: {
                        console.log("Button was press");
                        about.show()
                    }
                }
```

5 总结

这个项目是以 QT Creator 为环境，C++与 QML 混合编程写成，在本程序中我学会了除 UI Designer、C++代码构建图形界面的另一种强大方法——QML，这种语言相对于传统编程，支持十分强大的动画效果，在这里我学会了简单布局及水滴按钮的使用。因其算法为游戏算法，故从网络上学习的较多，本人已掌握算法并能复现，该程序是我从这门课程中获得知识最多的程序。

实验四 学生通讯录

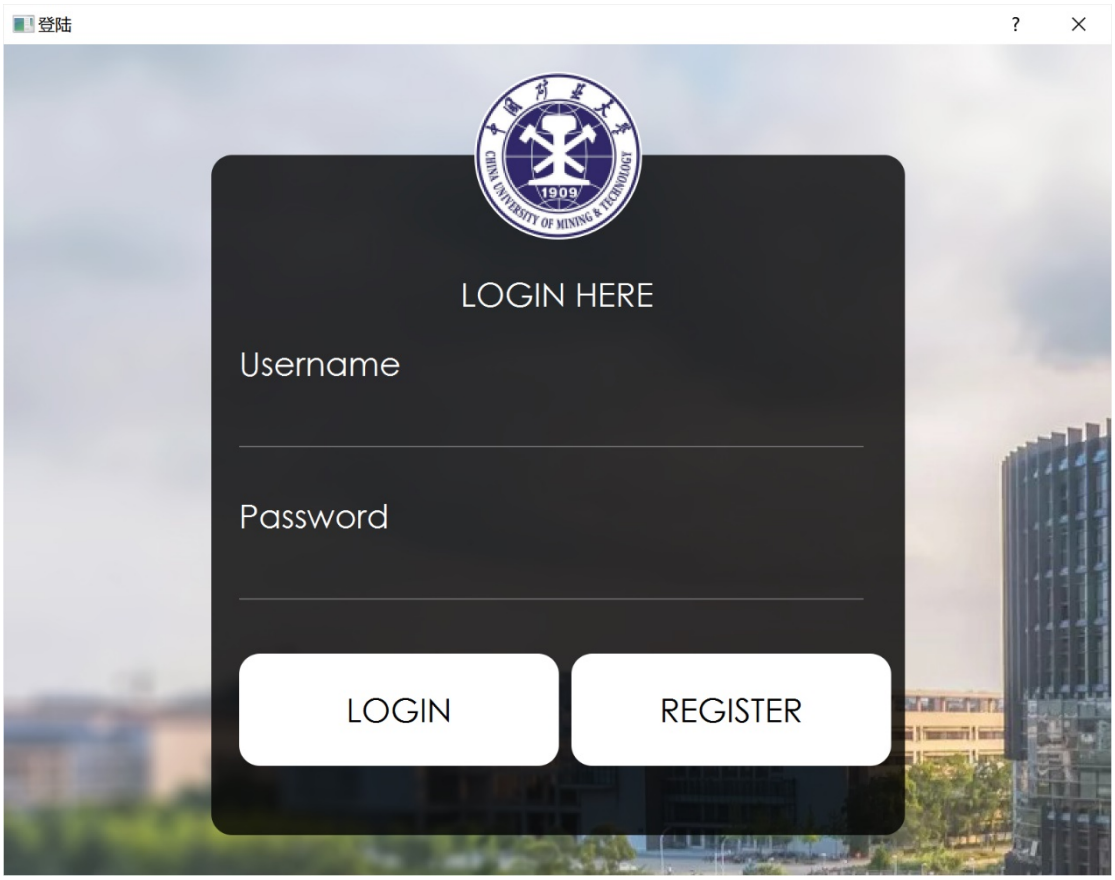
1 系统概述

本项目名为学生通讯录，个人认为叫做学生信息管理系统更为贴切，遂改为此名称。其主要内容为各种学生信息，程序采用 SQL 数据库进行数据存储，具备学生信息添加、编辑、删除、查找等功能，并在此基础上设计登陆界面，所用知识前三个项目基本涵盖。

2 系统设计

2.1 设计目标

设计开发一个学生信息管理系统软件，采用 SQL 数据库存储数据，可用特定软件(如 SQLiteSpy)打开，具备学生信息添加、编辑、删除、查找等功能。



界面如图 4.1、图 4.2 所示。

图 4.1 登陆界面设计

图 4.2 学生管理系统主界面

2.2 设计分析与算法流程

2.2.1 添加信息

设计思路：

① 从 QLineEdit 获取数据；

② 利用 query.exec()调用 SQL 语句 insert into student values()写入数据库。

代码如下：

```
1    void insertDialog::on_insert_pushButton_clicked()
2    {
3        if(ui->studentname_lineEdit->text().isEmpty()    ||
4           ui->studentnumber_ineEdit->text().isEmpty()   ||
5           ui->studentcollege_lineEdit->text().isEmpty() ||
6           ui->studentage_lineEdit->text().isEmpty()     ){
7            QMessageBox::information(this,"提示","请输入完整");
8        }
9        else{
10           QString student_name    = ui->studentname_lineEdit->text();
11           QString student_number  = ui->studentnumber_ineEdit->text();
12           QString student_college = ui->studentcollege_lineEdit->text();
13           QString student_age     = ui->studentage_lineEdit->text();
14           QString student_sex;
```

```
15          int rto=qbtg1->checkedId();
16          switch (rto) {
17          case 1:student_sex="男";break;
18          case 0:student_sex="女";break;
19          }
20          //QString student_sex    = ui->studentsex_lineEdit->text();
21
22          QSqlQuery query;
23          int index_delete = 0;
24          query.exec("select * from student");
25          while(query.next()){
26              if(student_number == query.value(1)){
27                  index_delete = 1;
28                  break;
29              }
30          }
31          if(index_delete == 1){
32              QMessageBox::information(this,"提示","学号已存在");
33          }
34          else{
35              QString insert_student = QString("insert into student
36   values(\"%1\",\"%2\",\"%3\",\"%4\",\"%5\")")
37                      .arg(student_name,student_number,student_college,stu
38   dent_age,student_sex);
39              qDebug()<<insert_student;
40              if(query.exec(insert_student)){
41                  QMessageBox::information(this,"成功","插入成功");
42              }
43              else{
44                  QMessageBox::information(this,"失败","插入失败");
45              }
46          }
47      }
48      this->close();
49  }
```

2.2.2 修改信息

设计思路：

① 通过 QlineEdit 获取信息；

② 通过原学号查找学生；

③ 利用 query.exec()调用 SQL 语句 update student set 写入数据库。

代码如下：

```
1    void updateDialog::on_update_pushButton_clicked()
2    {
3        if(ui->number_lineEdit->text().isEmpty()       ||
4           ui->numberchange_lineEdit->text().isEmpty()  ||
5           ui->namechange_lineEdit->text().isEmpty()    ||
6           ui->collegechange_lineEdit->text().isEmpty() ||
7           ui->agechange_lineEdit->text().isEmpty()     ||
8           ui->sexchange_lineEdit->text().isEmpty()){
9            QMessageBox::information(this,"提示","请输入完整内容");
10       }
11       else{
12           QString number = ui->number_lineEdit->text();
```

```
13          QString namechange =ui->namechange_lineEdit->text();
14          QString numberchange = ui->numberchange_lineEdit->text();
15          QString collegechange = ui->collegechange_lineEdit->text();
16          QString agechange =ui->agechange_lineEdit->text();
17          QString sexchange =ui->sexchange_lineEdit->text();
18          QString telchange =ui->sexchange_lineEdit->text();
19          QString zhuzichange =ui->sexchange_lineEdit->text();
20          QSqlQuery query;
21          query.exec("select * from student");
22          int t =0;
23          while(query.next()){
24              if(number == query.value(1)){
25                  t=1;
26                  break;
27              }
28          }
29          if(t==1){
30              QString update_student =
31                      QString("update student set
32    StudentName=\"%1\" ,StudentNumber = \"%2\",
33    StudentCollege =\"%3\" ,StudentAge =\"%4\",
34    StudentSex  =\"%5\" where StudentNumber =\"%6\"")
35                      .arg(namechange,numberchange,collegechange,agechang
36    e,sexchange,number,telchange,zhuzichange);
37              qDebug()<<update_student;
38              if(!query.exec(update_student)){
39                  QMessageBox::information(this,"失败","修改失败");
40              }
41              else{
42                  QMessageBox::information(this,"成功","修改成功");
43              }
44          }
45          else
46              QMessageBox::information(this,"失败","学号不存在");
47      }
48  }
49
```

2.2.3 查询信息

设计思路：

① 提供两种搜索办法，管理员选择其中一种；

② 从 QLineEdit 中获取数据；

③ 利用 query.value(0).toString()==student_name 或 query.value(1).toString()== student_number 在数据库中查找。

代码如下：

```
1   void selectDialog::on_select_pushButton_clicked()
2   {
3
4       if(ui->scanf_lineEdit->text().isEmpty()){
5           QMessageBox::information(NULL,"提示","请输入查找内容");
6       }
7       else{
8           if(ui->studentname_radioButton->isChecked()){
```

```
9             QString student_name = ui->scanf_lineEdit->text();
10            QSqlQuery query;
11            query.exec("select * from student");
12            int t =0;
13            int check =0;
14            while(query.next()){
15                if(query.value(0).toString()==student_name){
16                    int index_row =ui->select_tableWidget->rowCount();
17                    ui->select_tableWidget->setRowCount(index_row+1);
18                    ui->select_tableWidget->setItem(t,0,new
19  QTableWidgetItem(query.value(0).toString()));
20                    ui->select_tableWidget->setItem(t,1,new
21  QTableWidgetItem(query.value(1).toString()));
22                    ui->select_tableWidget->setItem(t,2,new
23  QTableWidgetItem(query.value(2).toString()));
24                    ui->select_tableWidget->setItem(t,3,new
25  QTableWidgetItem(query.value(3).toString()));
26                    ui->select_tableWidget->setItem(t,4,new
27  QTableWidgetItem(query.value(4).toString()));
28                    ui->select_tableWidget->setItem(t,5,new
29  QTableWidgetItem(query.value(5).toString()));
30                    ui->select_tableWidget->setItem(t,6,new
31  QTableWidgetItem(query.value(6).toString()));
32                    ui->select_tableWidget->setItem(t,7,new
33  QTableWidgetItem(query.value(7).toString()));
34                    check =1;
35                    qDebug()<<query.value(0).toString()
36                           <<","<<query.value(1).toString()
37                           <<","<<query.value(2).toString()
38                           <<","<<query.value(3).toString()
39                           <<","<<query.value(4).toString();
40                }
41                t++;
42            }
43            if(check !=1){
44                QMessageBox::information(NULL,"提示","输入姓名不存在");
45            }
46        }
47        else if(ui->studentnumber_radioButton->isChecked()){
48            QString student_number = ui->scanf_lineEdit->text();
49            QSqlQuery query;
50            query.exec("select * from student");
51            int t =0;
52            int check =0;
53            while(query.next()){
54                if(query.value(1).toString()== student_number){
55                    int index_row =ui->select_tableWidget->rowCount();
56                    ui->select_tableWidget->setRowCount(index_row+1);
57                    ui->select_tableWidget->setItem(t,0,new
58  QTableWidgetItem(query.value(0).toString()));
59                    ui->select_tableWidget->setItem(t,1,new
60  QTableWidgetItem(query.value(1).toString()));
61                    ui->select_tableWidget->setItem(t,2,new
62  QTableWidgetItem(query.value(2).toString()));
63                    ui->select_tableWidget->setItem(t,3,new
64  QTableWidgetItem(query.value(3).toString()));
65                    ui->select_tableWidget->setItem(t,4,new
66  QTableWidgetItem(query.value(4).toString()));
```

```
                    ui->select_tableWidget->setItem(t,5,new
QTableWidgetItem(query.value(5).toString()));
                    ui->select_tableWidget->setItem(t,6,new
QTableWidgetItem(query.value(6).toString()));
                    ui->select_tableWidget->setItem(t,7,new
QTableWidgetItem(query.value(7).toString()));
                    check =1;
                    qDebug()<<query.value(0).toString()
                           <<","<<query.value(1).toString()
                           <<","<<query.value(2).toString()
                           <<","<<query.value(3).toString()
                           <<","<<query.value(4).toString();
                }
                t++;
            }
            if(check!=1){
                QMessageBox::information(NULL,"提示","学号不存在");
            }
        }
        else{
            QMessageBox::information(NULL,"提示","请选择姓名或学号查询");
        }
    }
}

```

2.2.4 删除信息

设计思路：

①②③同上；

增加④利用 query.exec()调用 SQL 语句 delete from student where StudentName 或 delete from student where StudentNumber 在数据库中删除该行。

代码如下：

```
1    void deleteDialog::on_delete_pushButton_clicked()
2    {
3        if(ui->scanf_lineEdit->text().isEmpty()){
4            QMessageBox::information(this,"提示","请输入内容");
5        }
6        else{
7            if(ui->studentname_radioButton->isChecked()){
8                QString student_name;
9                student_name = ui->scanf_lineEdit->text();
10               QSqlQuery query;
11               query.exec("select * from student");
12               int t =0;
13               while(query.next()){
14                   if(student_name == query.value(0).toString()){
15                       t= 1;
16                       break;
17                   }
18               }
19               if(t == 1){
```

```
20                QString delete_name = QString("delete from student
21    where StudentName = \"%1\"").arg(student_name);
22                qDebug()<<delete_name;
23                if(query.exec(delete_name)){
24                    QMessageBox::information(this,"成功","删除成功");
25                }
26                else{
27                    QMessageBox::information(this,"失败","删除失败");
28                }
29            }
30            else{
31                QMessageBox::information(this,"提示","不存在的姓名");
32            }
33        }
34        else if(ui->studentnumber_radioButton->isChecked()){
35            QString student_number;
36            student_number = ui->scanf_lineEdit->text();
37            QSqlQuery query;
38            query.exec("select * from student");
39            int t =0;
40            while(query.next()){
41                if(student_number == query.value(1).toString()){
42                    t= 1;
43                    break;
44                }
45            }
46            if( t== 1){
47                QString delete_student = QString("delete from student
48    where StudentNumber = \"%1\"").arg(student_number);
49                if(query.exec(delete_student)){
50                    QMessageBox::information(this,"成功","删除成功");
51                }
52                else{
53                    QMessageBox::information(this,"失败","删除失败");
54                }
55            }
56            else{
57                QMessageBox::information(this,"提示","不存在的学号");
58            }
59        }
60        else{
61            QMessageBox::information(this,"提示","请选择姓名或学号");
62        }
63    }
64    }
```

2.3 界面设计

本项目窗口和按钮多，故采用 UI Designer 和 CSS 语言设计，见图 4.

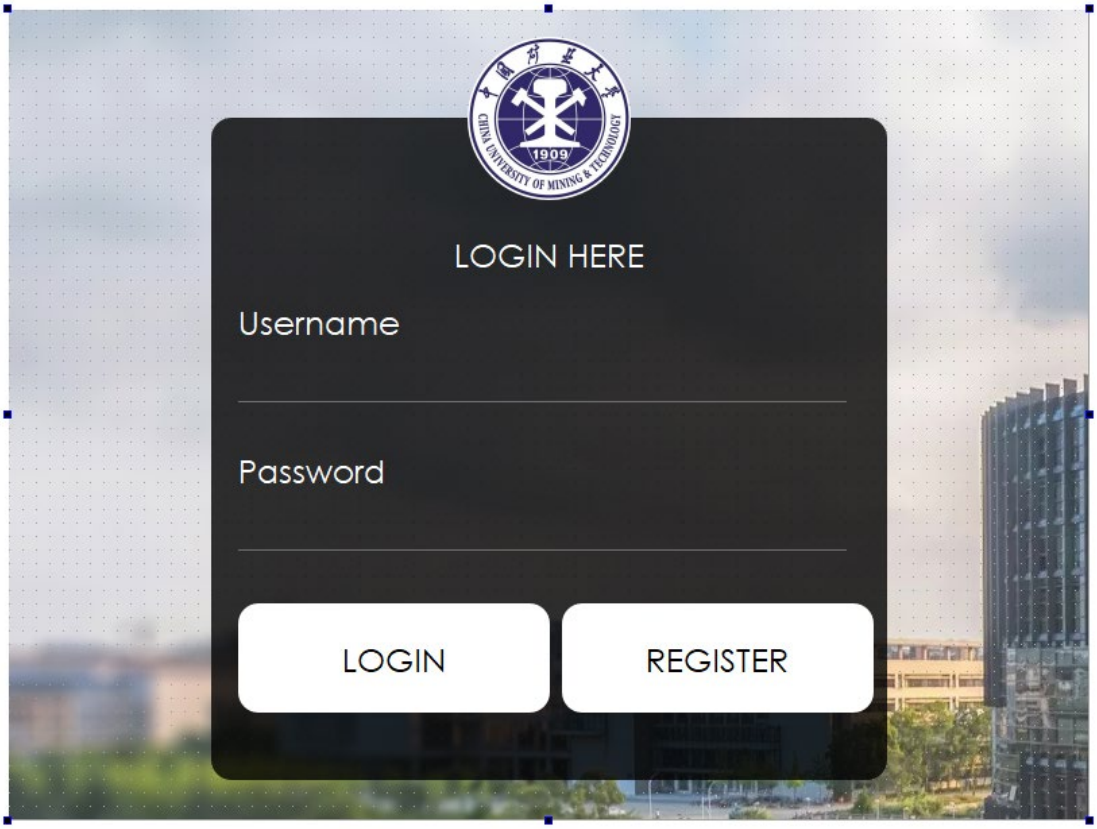


图 4.3 登陆界面设计

图 4.4 主界面设计

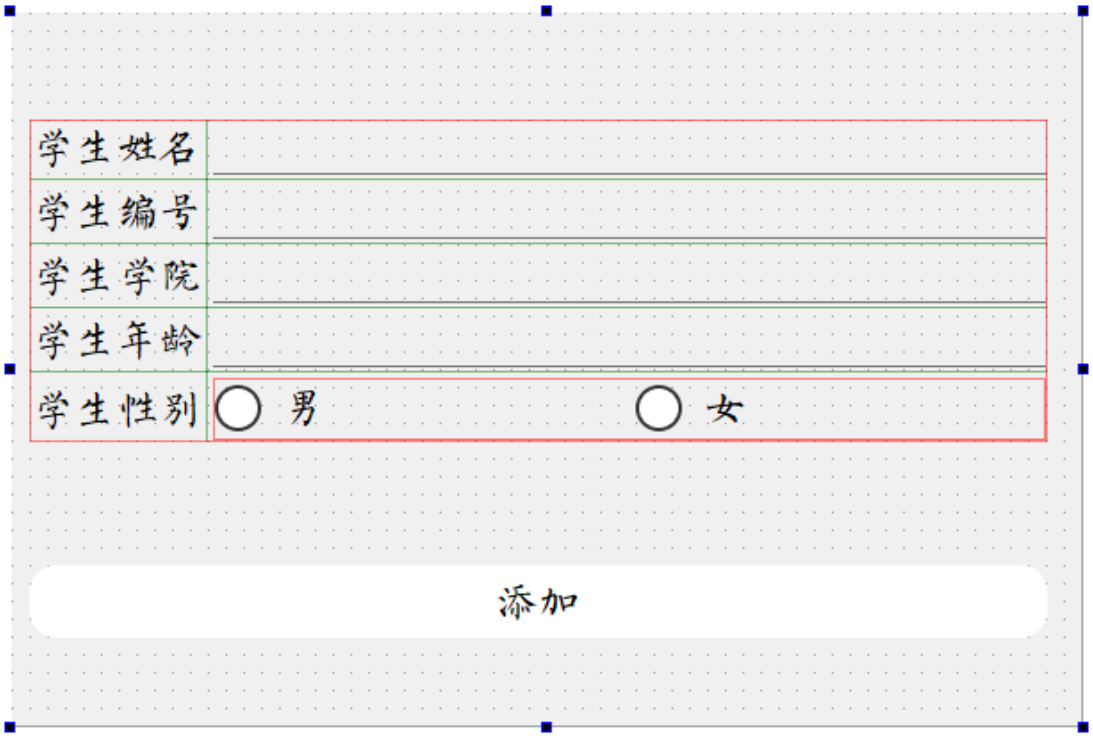


图 4.5 添加信息窗口设计

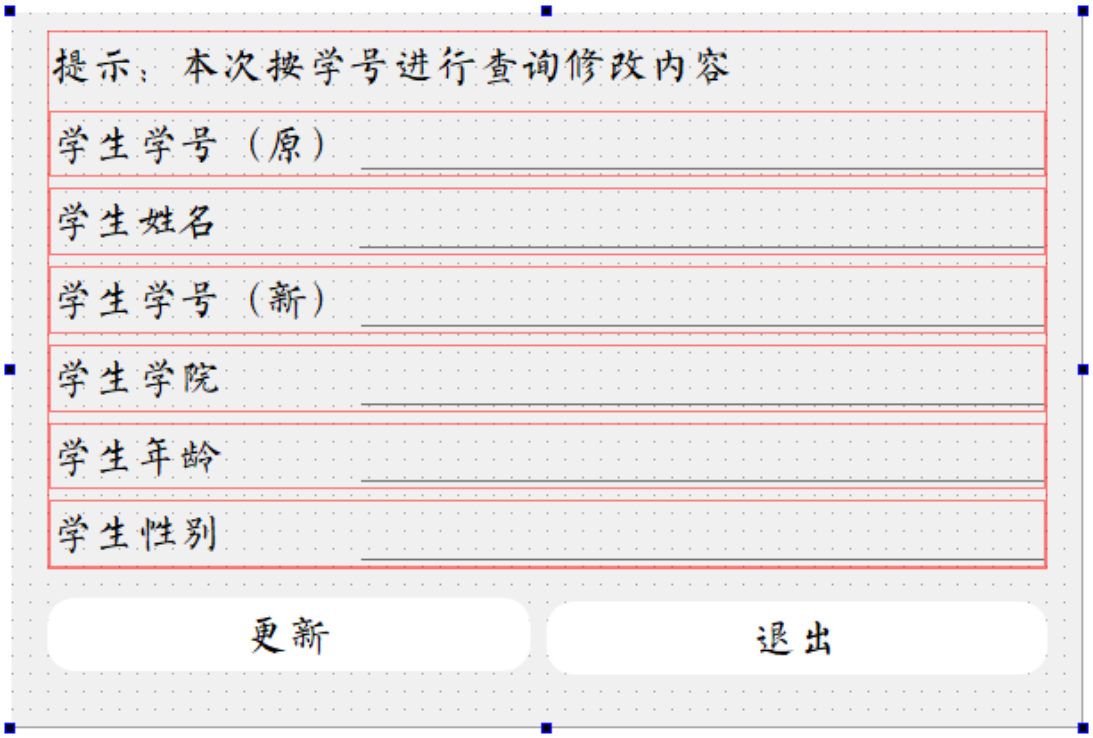


图 4.6 修改信息窗口设计

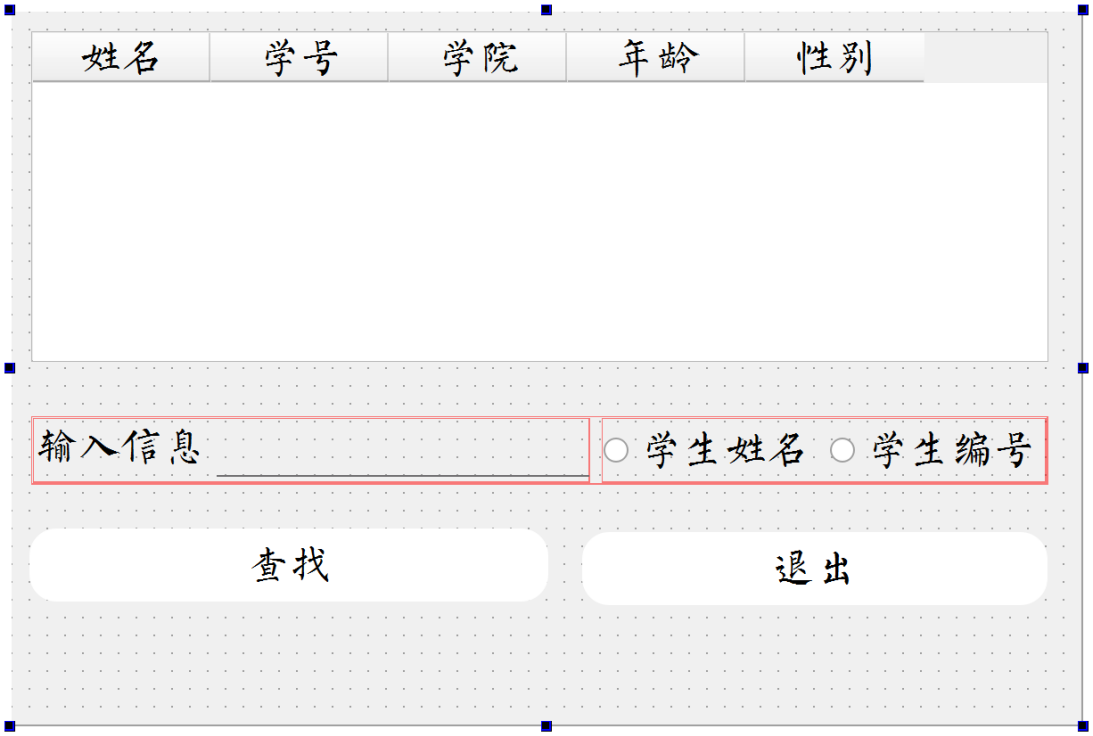


图 4.7 查询信息窗口设计

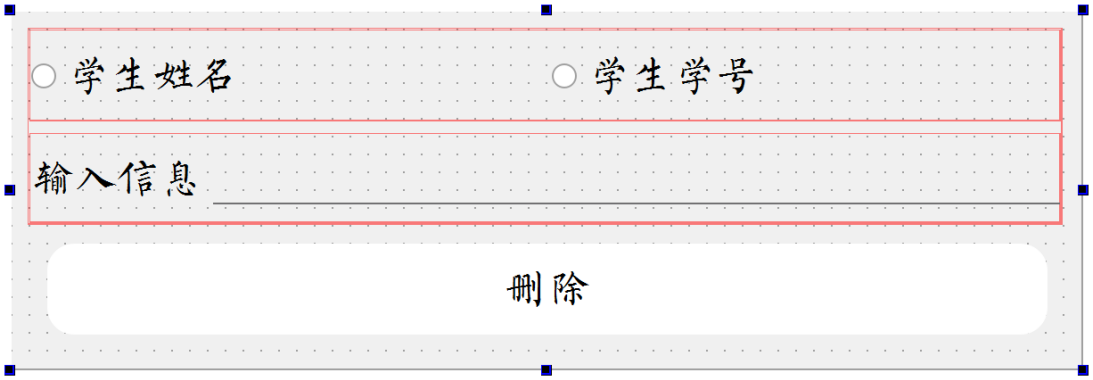


图 4.8 删除信息窗口设计

本程序按钮较多且分散，在这里不给出按键绑定图。

2.4 关键类图

见图 4.9

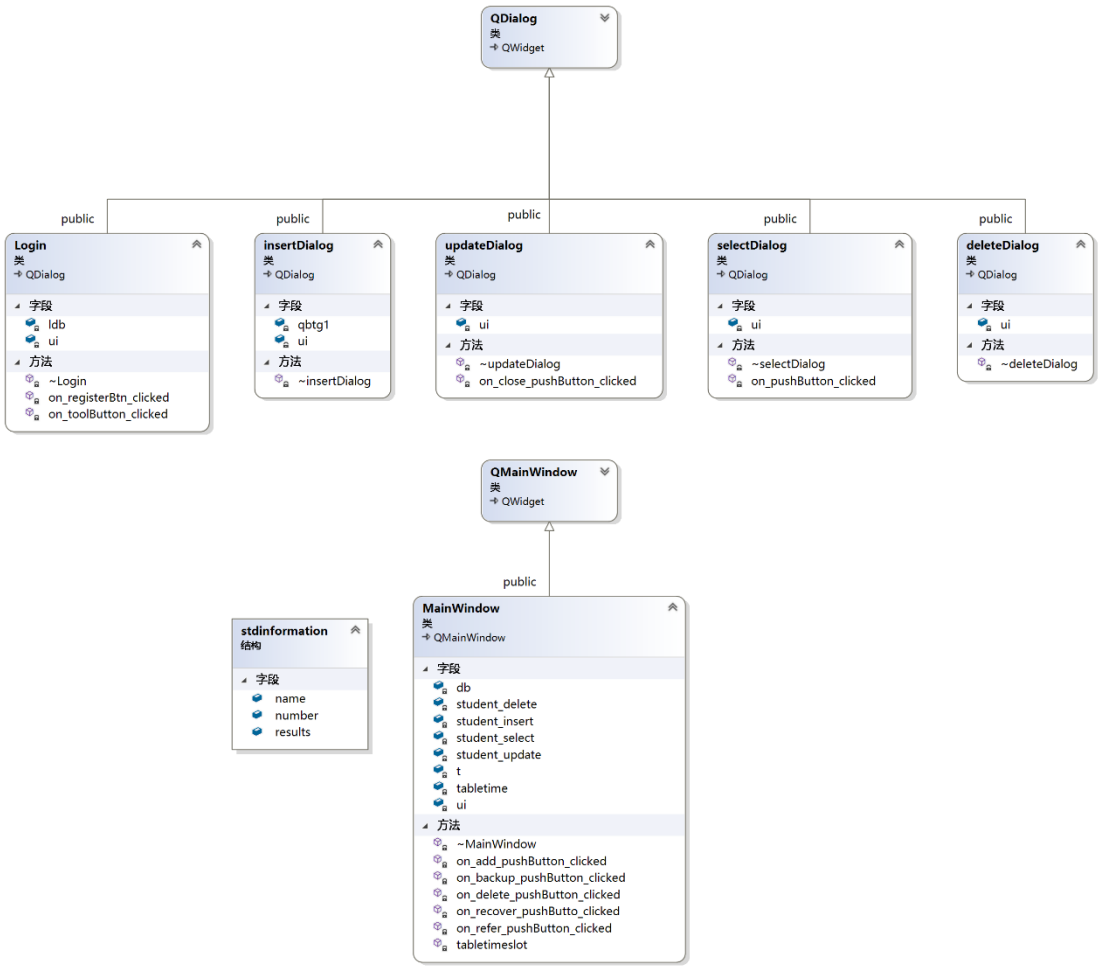


图 4.9 学生管理系统关键类图

3 系统实现（运行调试）

一些示例操作见图 4.10-图 4.13

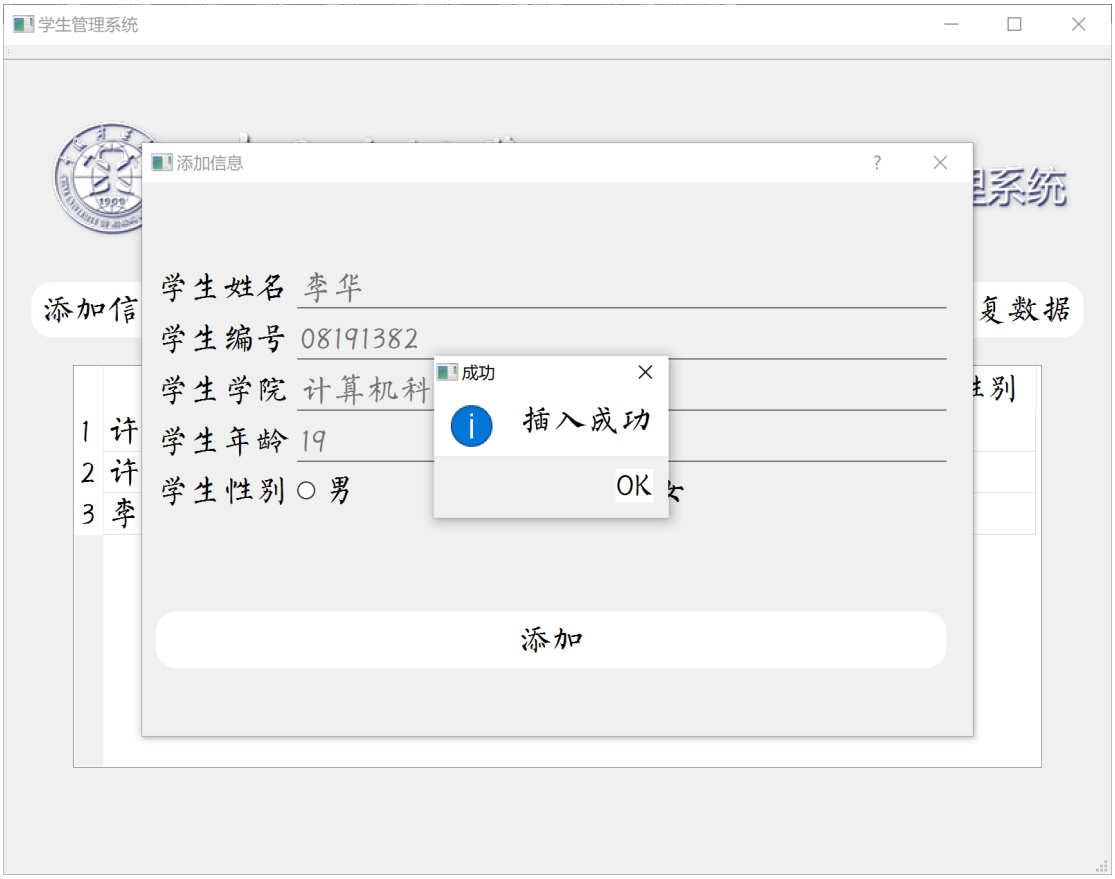

学生管理系统
添加信息
学生姓名 李华
学生编号 08191382
学生学院 计算机科
学生年龄 19
学生性别 男
添加
成功
插入成功
OK
添加信
复数据
1 许
2 许
3 李

图 4.10 示例操作 1

图 4.11 示例操作 2

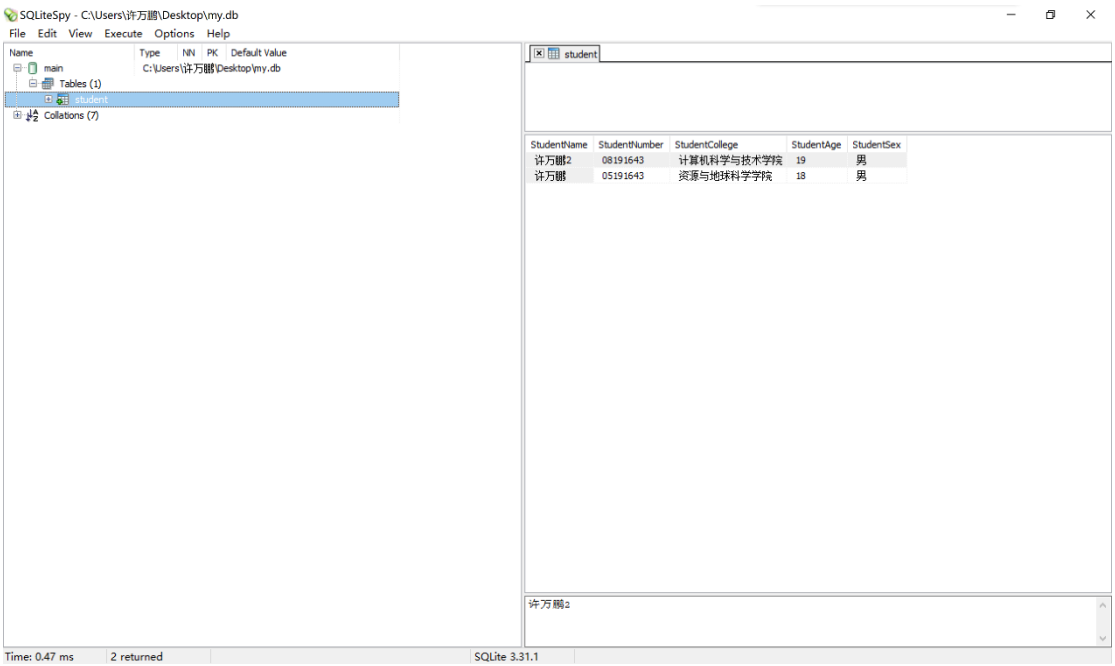


图 4.12 示例操作 3

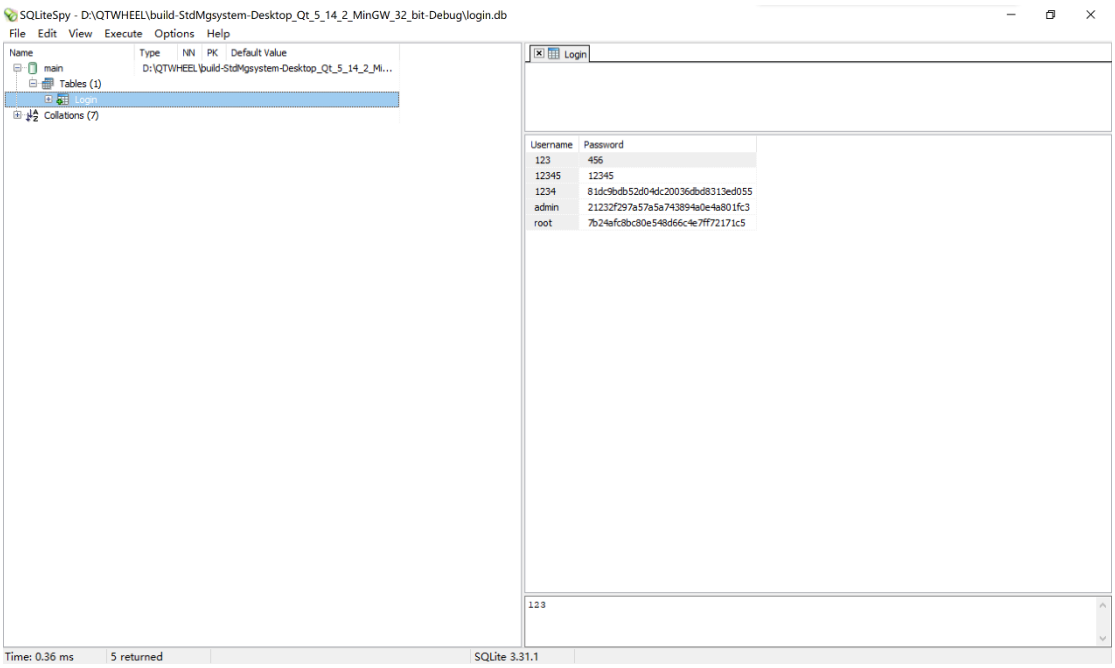


图 4.13 示例操作 4

4 系统扩展

4.1 登陆界面设计

登陆界面采用 UI Designer 与 CSS 样式表混合编程而来，UI Designer 见图 4.3，下面给出 CSS 代码：

```
*{
font-size:24px;
font-family:Century Gothic;
}
QFrame{
background:rgba(0,0,0,0.8);
border-radius:15px;
}
#Login{
background:url(D:/QTWHEEL/myContracts/pic/loginbg.jpg);
}

QToolButton{
background:white;
border-radius:60px;
}
QLabel{
color:white;
background:transparent;
}
QPushButton{
background:white;
border-radius:15px;
}
QPushButton:hover{
background:#333;
border-radius:15px;
background:#49ebff;
```

```
29   }
30   QLineEdit{
31   background:transparent;
32   border:none;
33   color:#717072;
34   border-bottom:1px solid #717072;
35   }
```

“登陆”功能与查询学生信息类似；

“注册”功能与添加学生信息类似，这里不再赘述。

为了使密码不被明文泄露，增加 MD5 保护，关键代码如下：

```
1  QString MD5;
2        QByteArray str;
3        str =
   QCryptographicHash::hash(pwd.toLatin1(),QCryptographicHash::Md5);
4
5        MD5.append(str.toHex());
```

被 MD5 保护的密码见图 4.13。

4.2 备份功能

设计思路：

① 使用 QFileDialog::getExistingDirectory 获取文件夹路径；

② 检查是否存在同名数据库，若存在，则删除；

③ 使用 QFile::copy 将数据库文件从当前工作目录拷贝到选择路径。

流程图如图 4.14

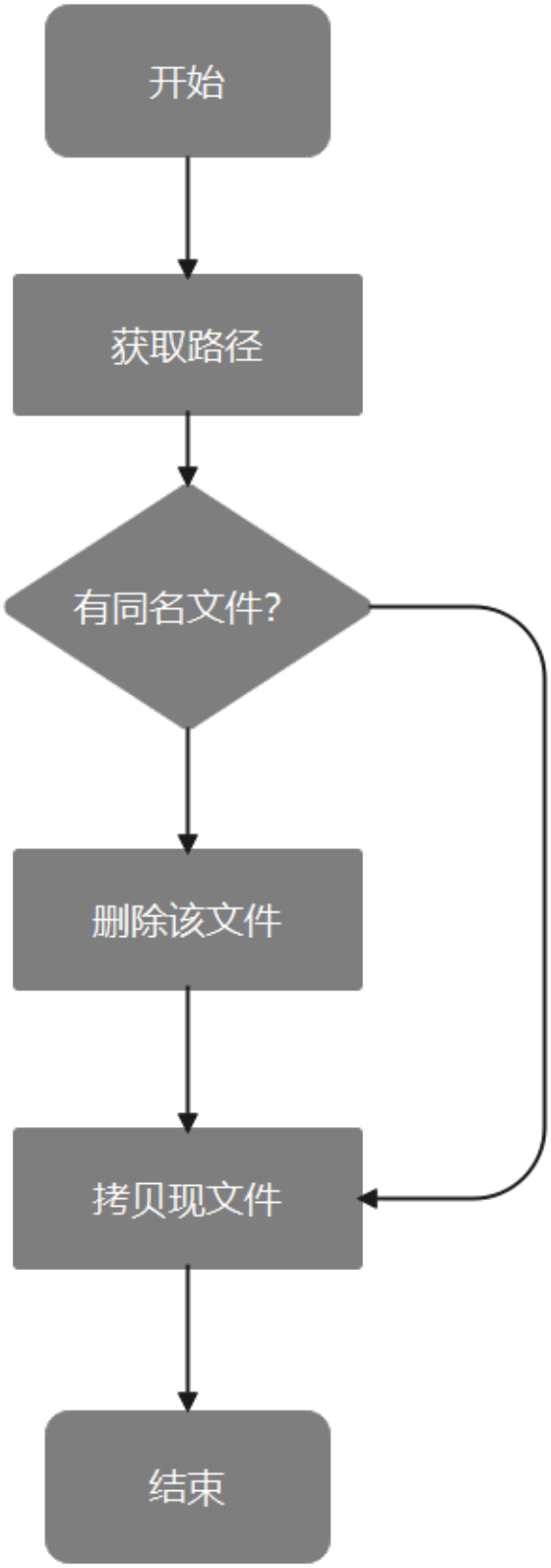


图 4.14 备份数据流程图

代码如下：

```
1   void MainWindow::on_backup_pushButton_clicked()
2   {
3       QString dirpath = QFileDialog::getExistingDirectory(this, "选择
4   目录", "./", QFileDialog::ShowDirsOnly);
5       if (QFile::exists(dirpath+"/my.db"))
6       {
7           QFile::remove(dirpath+"/my.db");
8       }
9
10      QFile::copy("my.db",dirpath+"/my.db");
11  }
```

4.3 恢复功能

设计思路：

基本同上，但因数据库正在被本程序占用，故需使用 db.close()关闭；拷贝完成后再打开，流程图见图 4.15。

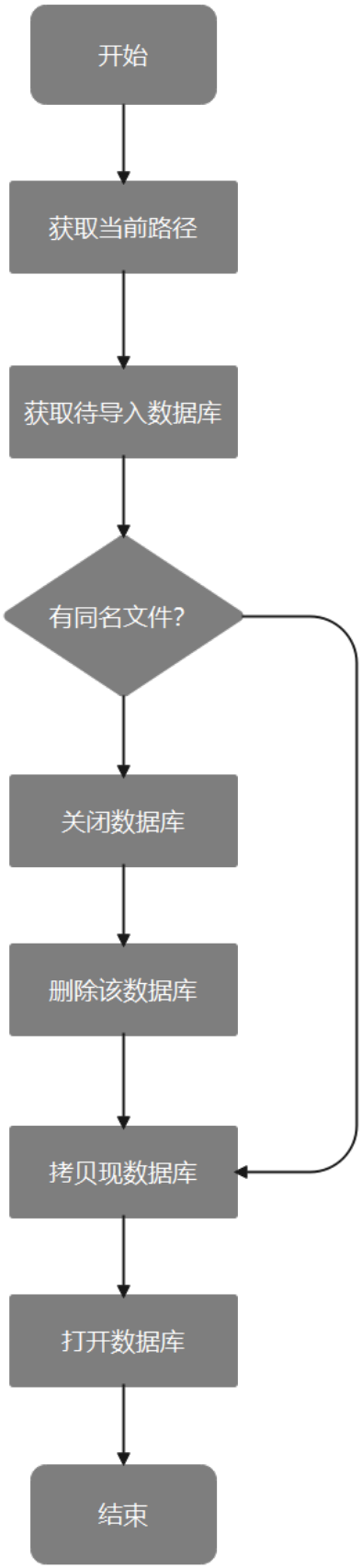


图 4.15 恢复数据流程图

代码如下：

```
1   void MainWindow::on_recover_pushButto_clicked()
2   {
3       QString curPath = QDir::currentPath();
4       QString aFileName = QFileDialog::getOpenFileName(this,"选择文件
5   ","./","数据库文件(*.db);;全部文件(*.*);");
6
7       if (QFile::exists(curPath+"/my.db"))
8       {
9           db.close();
10          QFile::remove(curPath+"/my.db");
11      }
12      QFile::copy(aFileName,curPath+"/my.db");
13      db.open();
14  }
```

5 总结

该项目是本课程最后一个项目，其运用知识在前面三个项目均已掌握，通过本项目主要学习了 QT 程序与 SQL 数据库的通信，了解了 CSS 语言在扁平化图形设计时的强大能力，其与 QML 各有所长，在设计界面时需要根据需求选择适当的语言。