# 个人总结文档

1120201215 杨昕璋

## 主要工作：

### 8.19~8.29：完成括号匹配类第一版

**主要源代码：**

Brackets类

#ifndef BRACKETS\_H

#define BRACKETS\_H

#include<QObject>

#include<QWidget>

class Brackets

{

public:

Brackets();

Brackets(int m,int dir,int r,int p);

void Set(int m,int dir,int r,int p);

int mode;//1=大括号，2=中括号，3=小括号

int direction;//1=左括号，2=右括号

int rank;//是第几个括号

int position;//在全部文本中的位置

int match;//配对括号

};

#endif // BRACKETS\_H

Matching类

#ifndef MATCHING\_H

#define MATCHING\_H

#include<QString>

#include<brackets.h>

#include<QObject>

#include<QList>

#include<QWidget>

#include<QStack>

class Matching :public QWidget

{

Q\_OBJECT

public:

Matching();

int finding(int pos);

void SetString(QString str);

private:

Brackets \* head;

QString s;

int rankLS ;//小括号

int rankRS;

int rankLM ;//中括号

int rankRM;

int rankLL ;//大括号

int rankRL;

QStack <Brackets> matchL;

QStack <Brackets> matchM;

QStack <Brackets> matchS;

QList <Brackets> ListL;

QList <Brackets> ListM;

QList <Brackets> ListS;

public slots:

void matching();

};

#endif // MATCHING\_H

主要功能：检测文本变化，实时生成括号类，在光标移动时，检测是否在括号后，如果在一个括号后面，则将对应的配对括号和该括号变色，以达到提醒括号匹配的作用。

存在的bug：该版本暂时无bug。

### 8.31~8.31：将括号匹配与ide的主面板连接起来

主要功能：检测文本变化，实时生成括号类，在光标移动时，检测是否在括号后，如果在一个括号后面，则将对应的配对括号和该括号变色，以达到提醒括号匹配的作用。

存在的bug：与其他人写的一部分程序冲突，导致选中和括号匹配不能一起运行。该bug触发原因在于两段程序都要设置鼠标光标的位置。

### 9.02~9.08：完善括号匹配的功能，新增括号自动生成功能

**主要源代码：**

void IDEEditor :: autoMatch(){

prelen = length;

length = textedit->toPlainText().length();

if(prelen>=length){

return ;

}

QString str;

textedit->blockSignals(true);

QTextCursor cursor = textedit->textCursor();

int pos = cursor.position();

cursor.movePosition(QTextCursor::PreviousCharacter,QTextCursor::MoveAnchor);

cursor.movePosition(QTextCursor::NextCharacter,QTextCursor::KeepAnchor);

textedit->setTextCursor(cursor);

str = cursor.selectedText();

cursor.movePosition(QTextCursor::NextCharacter,QTextCursor::MoveAnchor);

textedit->setTextCursor(cursor);

if(str == "{"){

for(int i=0;i<m->ListL.size();i++){

if(pos == m->ListL[i].position){

qDebug()<<m->ListL[i].match;

if(m->ListL[i].match!=0){

textedit->blockSignals(false);

return ;

}

}

}

qDebug()<<"auto match";

textedit->insertPlainText("}");

transToMatching();

cursor.movePosition(QTextCursor::PreviousCharacter);

textedit->setTextCursor(cursor);

}

else if(str == "["){

for(int i=0;i<m->ListM.size();i++){

if(pos == m->ListM[i].position){

qDebug()<<m->ListM[i].match;

if(m->ListM[i].match!=0){

textedit->blockSignals(false);

return ;

}

}

}

qDebug()<<"auto match";

textedit->insertPlainText("]");

transToMatching();

cursor.movePosition(QTextCursor::PreviousCharacter);

textedit->setTextCursor(cursor);

}

else if(str == "("){

for(int i=0;i<m->ListS.size();i++){

if(pos == m->ListS[i].position){

qDebug()<<m->ListS[i].match;

if(m->ListS[i].match!=0){

textedit->blockSignals(false);

return ;

}

}

}

qDebug()<<"auto match";

textedit->insertPlainText(")");

transToMatching();

cursor.movePosition(QTextCursor::PreviousCharacter);

textedit->setTextCursor(cursor);

}

else{

textedit->blockSignals(false);

return ;

}

textedit->blockSignals(false);

return ;

}

主要功能：打出一个前括号自动生成一个后括号。修改了之前括号匹配时存在的一些bug，如：匹配错误，不能实时匹配。

**存在的bug：在工作过程中出现了很多bug。**

**1.括号匹配和括号自动生成不能同时起效。Bug原因在于信号函数和槽函数的问题。**

**2.删除时也会触发括号自动生成的功能。Bug原因在于没有区分新增和删除。**

**3.打出反括号后不会自动匹配。**

**以上bug在工作过程中已经解决。造成bug的原因让我很困惑：关于Qt的信号和槽函数的触发机制，同一个信号连接多个槽函数时，它们的触发顺序是由什么决定的？在测试中我发现他们并不由connect的先后顺序决定，这导致了我之前写的多个connect函数不能很好的一起运行。于是我把所有槽函数放在了同一个槽函数里，让他们有严格的先后执行顺序，解决了bug。**

### 9.09~9.10：函数折叠功能

**主要源代码：**

void IDEEditor::fold(){

int rowNum = lineNumberWidget->textCursor().blockNumber()+1;

qDebug()<<rowNum;

//起始行数，终止行数

int start =0;

int end = 0;

//遍历查找

for(int i=0;i<m->ListL.size();i++){

Brackets b = m->ListL.at(i);

qDebug()<<b.row<<"bracket";

if(b.row == rowNum&&b.direction == 1){

qDebug()<<"true";

for(int j=0;j<m->ListL.size();j++){

if(m->ListL.at(j).direction == 2&&b.match == m->ListL.at(j).match){

start = rowNum-1;

// qDebug()<<rowNum<<"start";

end = m->ListL.at(j).row-1;

break;

}

}

}

}

//如果可以折叠

if(start>=0&&end>=0&&end>start){

// qDebug()<<"FOLD";

QTextDocument \* doc = textedit->document();

QTextDocument \* rowDoc = lineNumberWidget->document();

for(int x=start+1;x<end;x++){

// qDebug()<<x;

QTextBlock block = doc->findBlockByNumber(x);

// qDebug()<<block.document()->toPlainText();

QTextBlock rowBlock = rowDoc->findBlockByNumber(x);

//如果折叠了就展开，如果展开了就折叠

if(rowBlock.isVisible() == true){

// qDebug()<<"visible";

block.setVisible(false);

qDebug()<<block.isVisible();

rowBlock.setVisible(false);

textedit->viewport()->update();

lineNumberWidget->viewport()->update();

}

else{

// qDebug()<<"invisible";

block.setVisible(true);

rowBlock.setVisible(true);

textedit->viewport()->update();

lineNumberWidget->viewport()->update();

}

doc->adjustSize();

rowDoc->adjustSize();

}

}

lineNumberWidget->blockSignals(true);

QTextCursor cursor = lineNumberWidget->textCursor();

cursor.movePosition(QTextCursor::NextBlock);

lineNumberWidget->setTextCursor(cursor);

lineNumberWidget->blockSignals(false);

return ;

}

主要功能：点击前大括号所在行数可以折叠该大括号。再次点击可以展开。

**出现bug：函数折叠以后左边行数条和右侧代码条不再同步。**

**Bug原因：最初的时候设置的是**cursor.movePosition(QTextCursor::Start);**这样就导致左侧的行数条会直接回到顶端。后来把其改成了cursor.movePosition(QTextCursor::NextBlock);就不会出现bug。**

## 个人感想

由于上学期学习的面向对象语言是JAVA，所以这个小学期实际上是我第一次用C++写面向对象的程序。在此期间实际上出现了很多问题，最普遍的就是在写代码时候没有真正面向对象，缺少封装的思想，写出的程序实际上还是面向过程的，耦合性和复用性都非常差。然后在多人合作，各自完成各自的模块时，代码拼接就会出现很大的问题。

举一个例子：我们在进行这个IDE设计时，本应该把每个模块都当作一个单独的类来写，但实际上最初的版本中，textedit就是一个单独存在的东西，而不是封装在某个类里面。这就导致后续的多文件工作非常难以进行。比较好的想法应该是把右侧的代码编辑栏和必要组件打包成一个类，每次新建窗口只需要new一个这样的类就可以。然而最初并没有这样进行，导致后续工作又花了很大的精力去修改。在后续意识到这个问题后，小组写的程序已经优化了不少，代码的复用性和耦合性都好了不少。

除了缺乏面向对象的思想以外，我对于C++里面的宏，Qt中的信号和槽函数了解甚少，导致在使用中出现不少问题。我给同一个信号函数连接了多个槽函数，最后运行时发生冲突，我不了解他们触发先后的机制。用debug进行调试后也未能看出其中的原理。最后只能改变办法，重写一个总体的槽函数，将之前的函数放进去，使其有严格的先后执行顺序，才解决了众多bug。

还有一个主要问题就是最初对于程序整体的设计并不完善，有很多功能都是后续缝补，类也并没有按照最初设想的类图来完成。以及我对异常处理并不了解，在完成程序中颇感吃力。

总体来说这次小学期的程序设计实践给我带来了很多体验和收获。让我对于小组完成程序的过程有了更深刻的认识，也让我对面向对象的编程有了实践意义上的体会。