|  |
| --- |
| IDE设计文档 |
| 详细设计说明书 |

|  |
| --- |
| 邱瑞亨、陈雨、柯杭  2019-9-8 |

详细设计说明书

2019年9月

目录

[1引言 2](#_Toc19003137)

[1.1编写目的： 2](#_Toc19003138)

[1.2项目背景： 2](#_Toc19003139)

[1.3术语/缩略词： 2](#_Toc19003140)

[1.4参考资料： 2](#_Toc19003141)

[2总体设计 3](#_Toc19003142)

[2.1需求概述： 3](#_Toc19003143)

[2.2软件结构： 3](#_Toc19003144)

[3程序描述 4](#_Toc19003145)

[3.1功能简介： 4](#_Toc19003146)

[3.1.1功能： 4](#_Toc19003147)

[3.1.2性能： 4](#_Toc19003148)

[3.1.3输入项目： 4](#_Toc19003149)

[3.1.4输出项目： 4](#_Toc19003150)

[3.2算法： 4](#_Toc19003151)

[3.2.1文件操作模块: 4](#_Toc19003152)

[3.2.2编辑模块: 5](#_Toc19003153)

[3.2.3帮助模块: 5](#_Toc19003154)

[3.2.4编译模块: 5](#_Toc19003155)

[3.3程序逻辑： 5](#_Toc19003156)

[3.4 接口定义： 6](#_Toc19003157)

[3.4.1文本主题使用Qsciscintilla类指针，变量名为editor 6](#_Toc19003158)

[3.4.2关于界面主体及按钮： 7](#_Toc19003159)

[3.4.3杂项： 10](#_Toc19003160)

[3.5测试要点：给出测试模块的主要测试要求 10](#_Toc19003161)

# 1引言

## 1.1编写目的：

详细设计是相对概要设计而言的，是瀑布开发流程的一个重要环节，在概要设计的高层设计的基础上，从逻辑上实现了每一模块的功能，是编码阶段的主要参考资料，是从高层到低层、逐步精化思想的具体实现。

本详细设计文档的内容包括各个模块的算法设计，接口设计，数据结构设计，交互设计等，可以作为需求人员、总体设计人员与开发人员的沟通工具，把静态页面无法体现的设计体现出来，包含整体设计对模块设计的规范，体现对设计上的一些决策，例如选用的算法，对一些关键问题的设计考虑等等，使开发人员能快速进入开发，提高沟通效率，减少沟通问题。

## 1.2项目背景：

代码推动着信息时代的发展，目前推出的各种代码编译器使得代码的书写和调试变得十分方便。不同的代码编译器有不同的风格，但最终目的都是提高人们编写的效率。本项目旨在开发出一款功能齐全，却又简约便捷的代码书写编译器。

## 1.3术语/缩略词：

|  |  |  |
| --- | --- | --- |
| 序号 | 术语/缩略语 | 说明 |
| 1 | QT | QT是一种跨平台c++图形用户界面应用程序开发框架。 |
| 2 | QsciScintilla | Scintilla是一个免费、跨平台、支持语法高亮的编辑控件。它完整支持源代码的编辑和调试，包括语法高亮、错误指示、代码完成（code completion）和调用提示(call tips)。能包含标记（marker）的页边（margin）可用于标记断点、折叠和高亮当前行。  而QScintilla是Scintilla在QT上的移植。 |
| 3 | TextStream | TextStream是一个流对象，主要用于访问文本文件中的内容。 |
| 4 | QMessageBox | 自定义消息提示对话框 |

## 1.4参考资料：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 文档名 | 作者 | 时间 | 版本 |
| 1 | 实用软件工程 | 孙玮 | 2011 | 第三版 |
| 2 | 详细设计说明书模板 | 百度文库 | 2018 | 整合版 |

# 2总体设计

## 2.1需求概述：

功能需求：

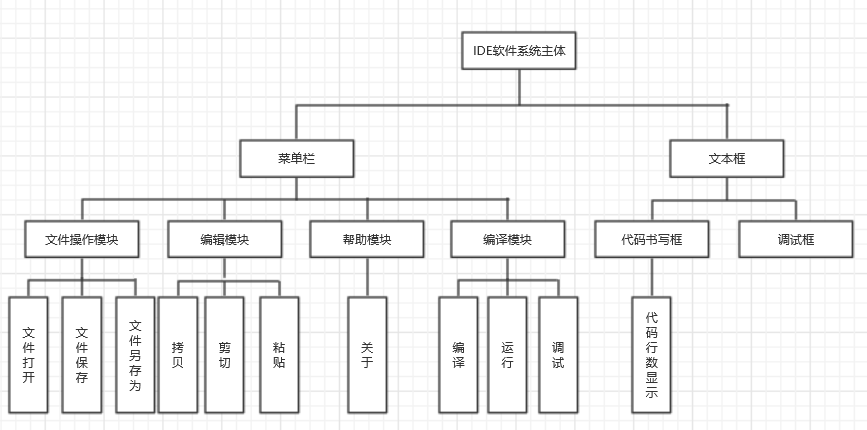
1. 提供对C程序的基本编辑功能：复制、粘贴、剪切，使得用户可以在IDE上进行代码的书写以及后续的修改工作。
2. 支持多文件编辑。
3. 能够对C语言源程序进行编译，并反馈编译结果，给出正确程序的执行结果。程序的编译和执行结果是程序员比较关心的问题，也是判断一个IDE是否有效的关键。
4. 支持关键字识别和自动补全，括号自动匹配，可以实现代码的高亮。关键字的识别和括号匹配减少了代码书写的工作量，代码高亮使得程序可读性更强，提高了工作的效率。
5. 支持Debug功能，针对代码运行中存在的相关问题可以快速进行调试。

性能需求：

1. 操作响应无明显延迟，将源代码编译以及结果显示时间降到最低，减少时间和资源的浪费。
2. 能够对C语言源程序的代码行数进行显示，便于对代码进行相应的操作和后续的管理。

## 2.2软件结构：

软件系统结构图 ：



# 3程序描述

## 3.1功能简介：

### 3.1.1功能：

利用Qt框架，实现简约的c语言编程。可以接收代码并进行编译、给出编译结果。对于代码文本，支持utf-8编码、关键字高亮以及自动补齐、行号计数显示、括号匹配等，在编辑过程中可以隐藏/打开程序中的全部注释语句。

### 3.1.2性能：

文本编辑功能，可接收代码行数为1000行以下，支持多文件编译，点击响应无人眼可分辨延迟，可给出正确编译结果。

### 3.1.3输入项目：

通过本程序提供的文件操作按钮打开或新建txt文本文件，在主窗口中的文本框输入想要编译运行的c语言代码。

### 3.1.4输出项目：

在较短时间内输出正确的c语言编译运行结果。

## 3.2算法：

### 3.2.1文件操作模块:

使用Qt库中得QFile类保存文件名和属性，QTextStream 流类型获取文本内容进行操作，注意字符串类型用QString封装类以保证兼容性。

利用QAction：：setShortcut设置不同操作的快捷键。

QString QFileDialog::getOpenFileName()打开文件的默认路径，其中第四个参数filter，是对话框的后缀名过滤器，比如我们使用"Image Files(\*.jpg \*.png)"就让它只能显示后缀名是jpg或者png的文件。如果需要使用多个过滤器，使用";;"分割，比如"JPEG Files(\*.jpg);;PNG Files(\*.png)"；

打开文件考虑用户取消的情况，如果用户取消则直接返回，其他情况下考虑文件的属性，最终将文件内容获取并利用editor->show()显示在editor中。

文件的保存分两种情况，新文件的保存和原有文件修改下的保存。利用flag\_isOpen判断文件是否打开，flag\_isNew判断文件是否新建。

另存为模块与文件保存模块使用函数基本一致。

### 3.2.2编辑模块:

Qsciscintilla 类自带编辑操作函数，cut paste selectall copy，利用editor->copy()，editor->paste()等即可实现拷贝、粘贴等功能。

### 3.2.3帮助模块:

connect函数链接按钮触发打开txt相关帮助文本文件即可。

### 3.2.4编译模块:

编译模块上主要运用了C语言向cmd命令行输入和信息提取。

#### 3.2.4.1编译：

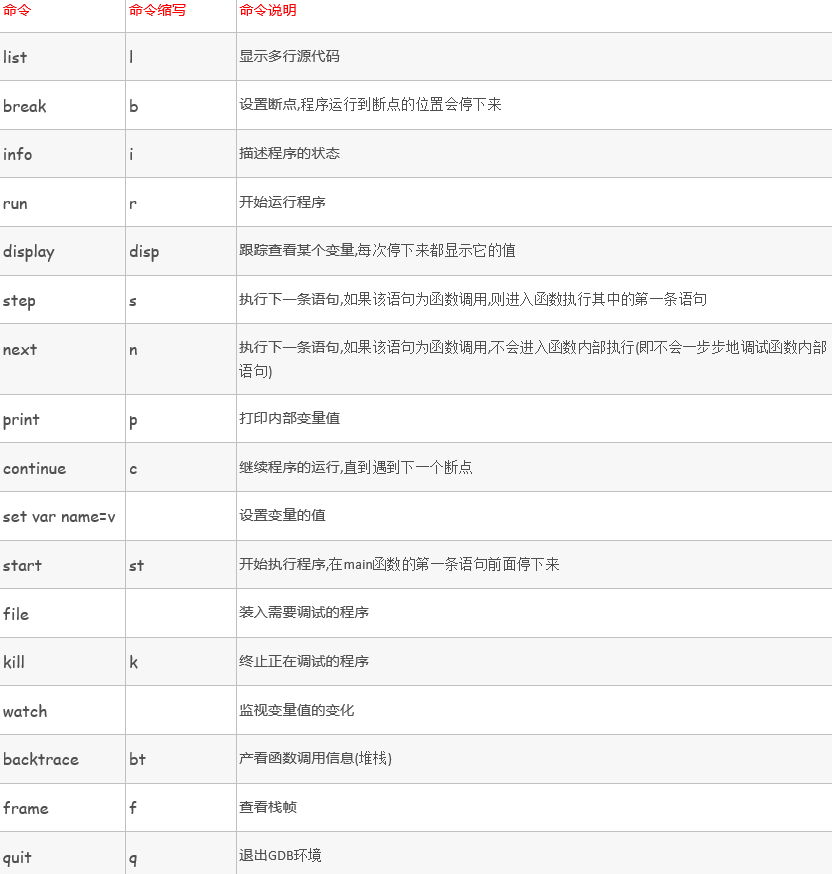
此IDE直接运用GCC命令进行编译，首先将.txt文件保存未.c文件，再通过gcc filename.c -o filename > err.txt 2>&1 这样的指令进行编译，同时，将cmd传输回来的错误正确信息保存在根目录的err.txt文件内，当文件不为空时，证明编译错误，将错误信息打印在提示框内。期间，文件的名字可以通过其所在路径提取。

#### 3.2.4.2运行：

由于之前编译已经将.exe文件建立，运行就可以直接 在cmd上输入filname.exe，同时需要用户在自己每个代码后加上system(“pause”)可以使窗口停留下来。

#### 3.2.4.3 调试：

调试同样有对应的指令，首先需要先生成用于调试的文件，先cmd输入gcc filename.c -o filename -g。生成后，先cmd输入指令 gdb filname 进入调试状态，输入不同的指令有不同的效果，如下：



由于dubug使状态，所以我们需要使用到多线程对其进行操作。同时，将cmd返回的信息存入文件内，再在程序上输出。

### 3.2.5 代码文本模块:

### 3.2.5.1关键字自动补齐及高亮

QsciLexer \*textLexer=new QsciLexerCPP;//设置cpp解析器

QsciAPIs \*apis = new QsciAPIs(textLexer);

apis->prepare();

editor->*setLexer*(textLexer);//设置词法分析器

editor->*setAutoCompletionSource*(QsciScintilla::AcsAll);//设置自动完成所有项

editor->*setAutoCompletionCaseSensitivity*(true);//大小写敏感

editor->*setAutoCompletionThreshold*(1);//每输入1个字符就出现自动完成的提示

editor->*setAutoIndent*(true);//自动缩进

### 3.2.5.2行号计数显示

editor->setMarginType(0,QsciScintilla::NumberMargin);

editor->*setMarginLineNumbers*(0,true);

editor->*setMarginWidth*(0,35);//设置行号计数

### 3.2.5.3选中行颜色设置

editor->*setIndentationGuides*(QsciScintilla::SC\_IV\_LOOKBOTH);

editor->*setCaretLineVisible*(true);//选中行颜色可见

editor->*setCaretLineBackgroundColor*(QColor(187,255,255));//选中所在行颜色

editor->*setMarginsBackgroundColor*(Qt::lightGray);//行号颜色

**3.2.5.4注释一键隐藏和显示**

默认检测最大行数，即最初设置的最大行数1000行，分两种情况检测：1、开头是//或句中带有//；2、开头是/\*

检测到后用QsciScintillaBase::SCI\_BEGINUNDOACTION设置undo断点起点然后使用QsciScintillaBase::SCI\_DELETERANGE信号对于范围内的注释进行删除，继续for循环检测所有行。最后设置QsciScintillaBase::SCI\_ENDUNDOACTION undo断点终点，用于重新显示注释。

利用QsciScintillaBase::SCI\_GETLINEENDPOSITION以及QsciScintillaBase::SCI\_GETLINEENDPOSITION信号以及for循环的标记i获取操作和结束行

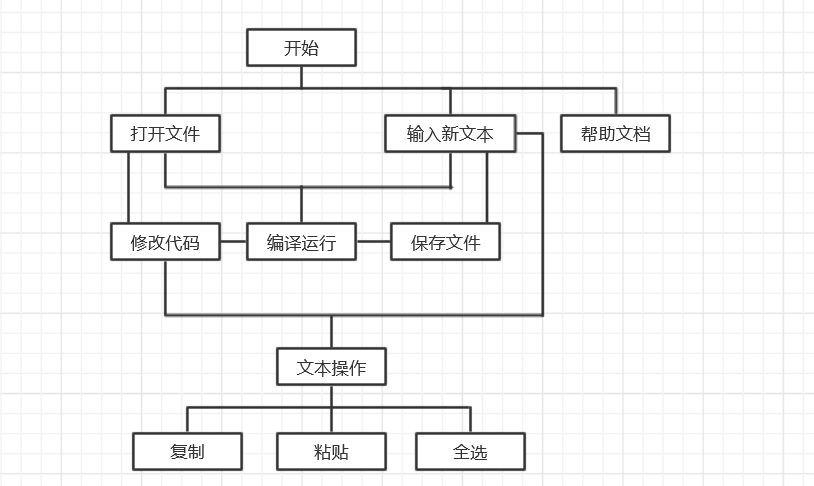
实际逻辑判断：

1. 对于开头是//的语句直接隐藏。

2-1检查句中的//隐藏//后半部分

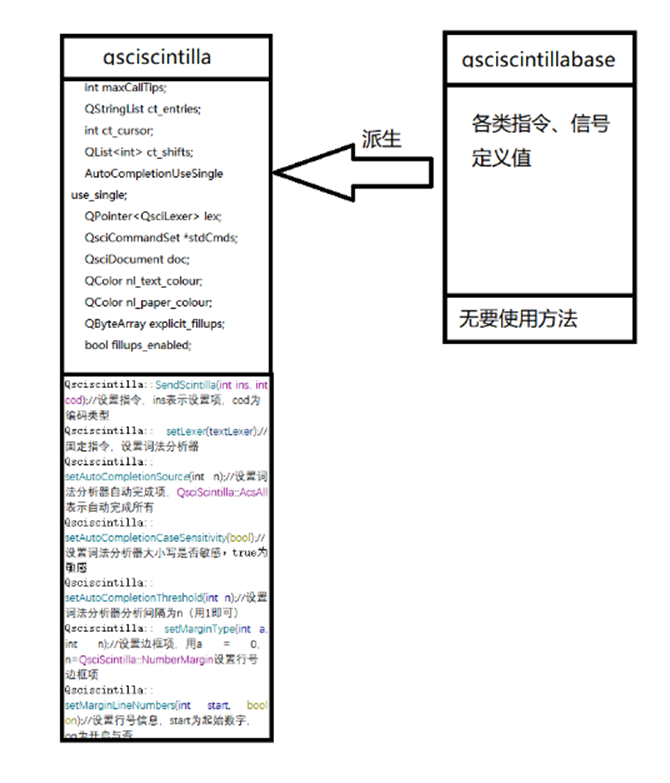
2-2检查句中的/\*另开for循环向下循环直到检测到\*/全部隐藏

## 3.3程序逻辑：



## 3.4 接口定义：

### 3.4.1文本主题使用Qsciscintilla类指针，变量名为editor



Qsciscintilla \*editor；

Qsciscintilla::SendScintilla(int ins, int cod);//设置指令，ins表示设置项，cod为编码类型

Qsciscintilla::setLexer(textLexer);//固定指令，设置词法分析器

Qsciscintilla::setAutoCompletionSourc*e*(int n);//设置词法分析器自动完成项，QsciScintilla::AcsAll表示自动完成所有

Qsciscintilla::setAutoCompletionCaseSensitivity(bool);//设置词法分析器大小写是否敏感，true为敏感

Qsciscintilla:: setAutoCompletionThreshold(int n);//设置词法分析器分析间隔为n（用1即可）

Qsciscintilla:: setMarginType(int a, int n);//设置边框项，用a = 0，n=QsciScintilla::NumberMargin设置行号边框项

Qsciscintilla::setMarginLineNumbers(int start, bool on);//设置行号信息，start为起始数字，on为开启与否

Qsciscintilla::setMarginWidth(0,35);//设置行号起始结束横向宽度坐标

Qsciscintilla::setIndentationGuides(QsciScintilla::SC\_IV\_LOOKBOTH);//默认缩进显示

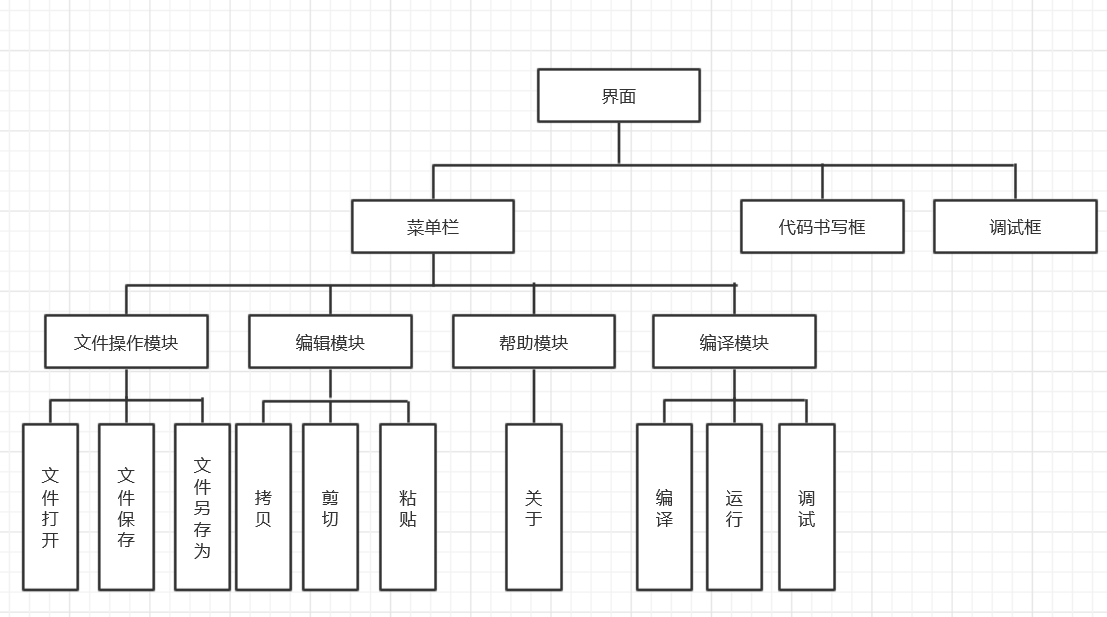
Qsciscintilla::setCaretLineVisible(bool on);//设置显示选中行行色true为显示

Qsciscintilla::setCaretLineBackgroundColor(QColor(x, x, x));//设置选中行行色，具体颜色查表即可

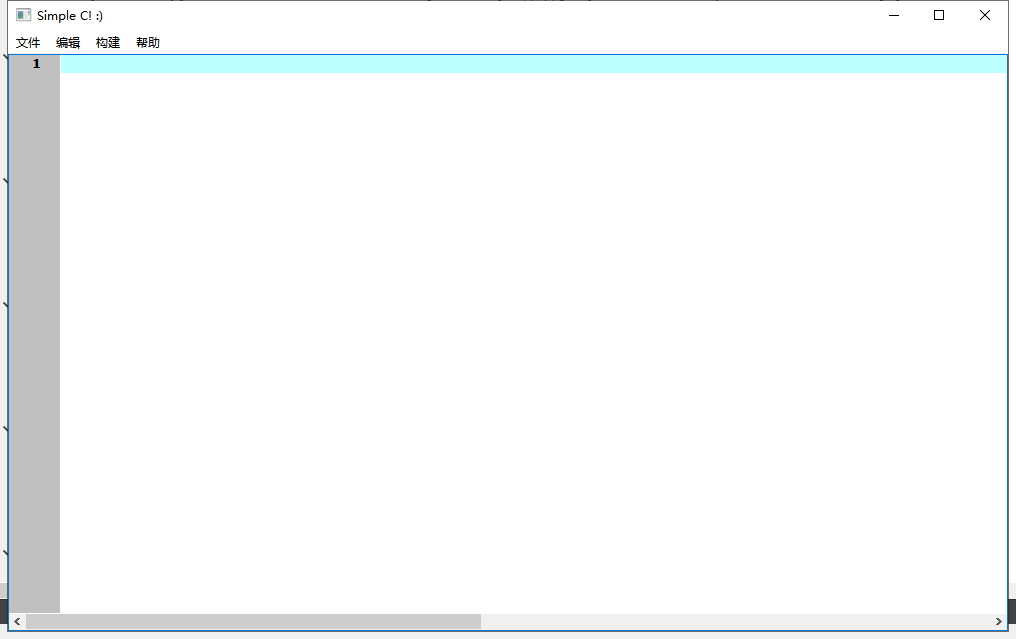
Qsciscintilla::setMarginsBackgroundColor(QColor(x, x, x));//行号色

### 3.4.2关于界面主体及按钮：

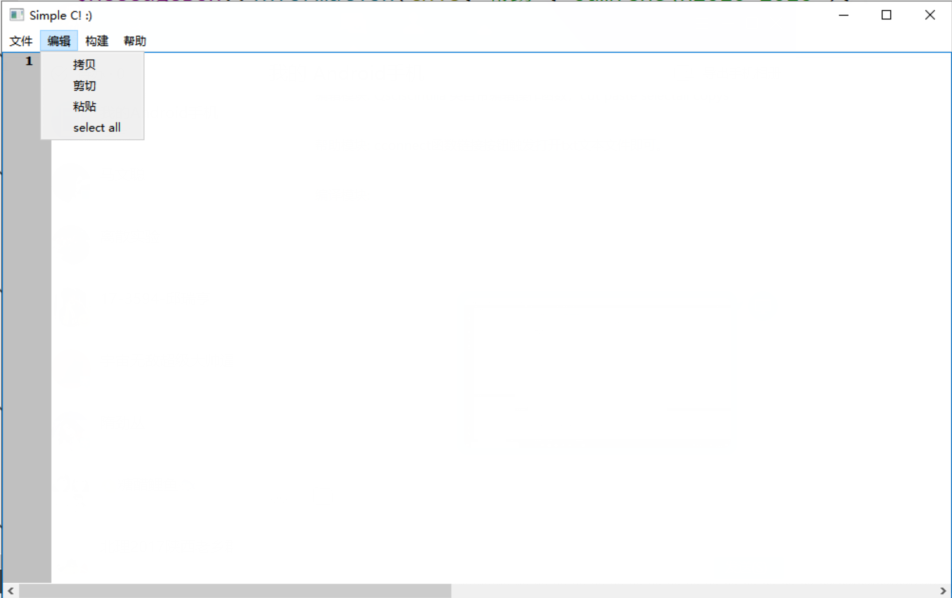
整体设计：



界面主体：



菜单栏详情：



class MainWindow: public QMainWindow {  
private:

explicit MainWindow(QWidget \*parent = 0);

QsciScintilla \*editor=new QsciScintilla(this);//文本主体  
  
//文件menu  
QMenu \*file;  
QAction \*file\_open;

QAction \*file\_exit;

QAction \*file\_save;//普通保存

QAction \*file\_save1;//另存为  
  
//编辑menu  
QMenu \*edit;  
QAction \*edit\_copy;

QAction \*edit\_cut;

QAction \*edit\_paste;

QAction \*select\_all;

//帮助menu;  
QMenu \*help;

QAction \*help\_about;  
  
//编译menu  
QMenu \*build; QAction \*Compile;  
QAction \*build\_compile;

QAction \*build\_run;

private slots:

void **on\_open**();

void **on\_about**();

void **on\_exit**();

void **on\_copy**();

void **on\_cut**();

void **on\_paste**();

void **on\_selectall**();

void **on\_save**();

void **on\_compile**();

void **on\_run**();

void **Save\_File**();

}  
  
class QMenu {  
private:  
char\* name;//菜单名称  
int num;//功能数   
public:  
QMenu(char\* aname) {  
strcpy(this->name, aname);  
this->num = 0;  
}  
  
void addAction(QAction buf);//关联功能  
void addbar(void);//加分隔符分隔不同功能  
}  
  
class QAction {  
private:  
char\* name;//动作命名  
char\* shortcut;//快捷键  
MainWindow self;//指向主体  
public:  
QAction(char\* aname, MainWindow aself) {  
strcpy(this->name, aname);  
this->self = aself;  
}  
  
void setShortcut(tr());//设置快捷键  
}；

void **get\_name**(char \*pf,int l);//由文件路径获取文件名字

void **get\_cmd**(char \*ch,char \*cmd,int mode);//获取向cmd上输入的命令

void **fd\_copy**();//由txt文件生成C文件

### 3.4.3杂项：

int flag\_isOpen = 0;判断是否已打开或新建文件，允许接收输入

int flag\_isNew = 0;//判断文件是否新建

QString Last\_FileName;//保存上一个文件名

QString Last\_FileContent;//保存上一个文件内容

QString filename;//正在操作的文件名