

项目名称：MiniIDE – C 语言编辑器

文档编号：2.0

MiniIDE – C语言编辑器

【设计方案】

总页数		正文		附录		生效日期	
编制				批准			

目录

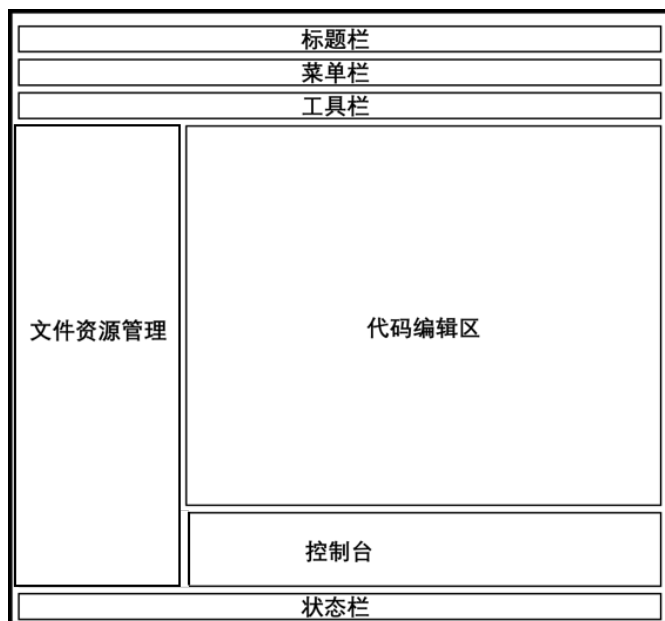
1	界面设计方案	4
1.1	界面整体布局设计方案	4
1.1.1	界面整体布局设计图	4
1.1.2	界面整体布局预览	4
1.2	标题栏设计方案	5
1.3	菜单栏设计方案	5
1.3.1	菜单栏整体设计方案	5
1.3.2	文件区域设计方案	5
1.3.3	编辑区域设计方案	6
1.3.4	编译运行区域设计方案	6
1.3.5	帮助区域设计方案	7
1.3.6	工具栏设计方案	8
1.4	代码编辑区设计方案	9
1.5	文件资源管理区设计方案	9
1.6	状态栏设计方案	10
1.7	控制台设计方案	10
2	功能设计方案	11
2.1	程序功能表	11
2.2	文本编辑	12
2.2.1	复制	12
2.2.2	粘贴	12
2.2.3	查找	12
2.2.4	替换	12
2.2.5	撤销	13
2.2.6	重做	13
2.2.7	显示当前行数和列号	14
2.2.8	调整字体大小	14
2.3	代码编辑	15
2.3.1	关键字识别高亮	15
2.3.2	关键字联想提示、补全	15
2.3.3	括号自动匹配高亮	15
2.3.4	括号自动补全	15
2.3.5	变量重命名	16
2.3.6	自动缩进	17
2.3.7	多行注释	18
2.3.8	注释隐藏 / 显示	19
2.3.9	代码块折叠	19
2.3.10	对已编辑的函数进行高亮显示	20
2.3.11	代码跳转	20
2.3.12	行格式排版	21
2.3.13	整体格式排版	22
2.4	代码运行	23
2.4.1	通过 gcc 对当前文件编译	23
2.4.2	通过 gcc 对当前文件运行	24

2.4.3	通过 gcc 实现多文件编译.....	25
2.4.4	通过 gcc 实现项目运行.....	27
2.4.5	在控制台显示编译结果.....	27
2.4.6	通过 gdb 对当前文件调试.....	28
3	文件管理方面.....	30
3.1.1	新建文件.....	30
3.1.2	打开文件.....	30
3.1.3	打开文件夹.....	31
3.1.4	保存文件.....	32
3.1.5	另存为文件.....	33
3.1.6	多文件编辑.....	34
3.1.7	树形文件资源管理.....	35
4	UML 设计.....	37
4.1	各功能设计.....	37
4.1.1	编辑器.....	37
4.1.2	编译功能.....	37
4.1.3	调试功能.....	37
4.1.4	窗口功能.....	37
4.1.5	按钮监听功能.....	37
4.2	UML 图.....	38

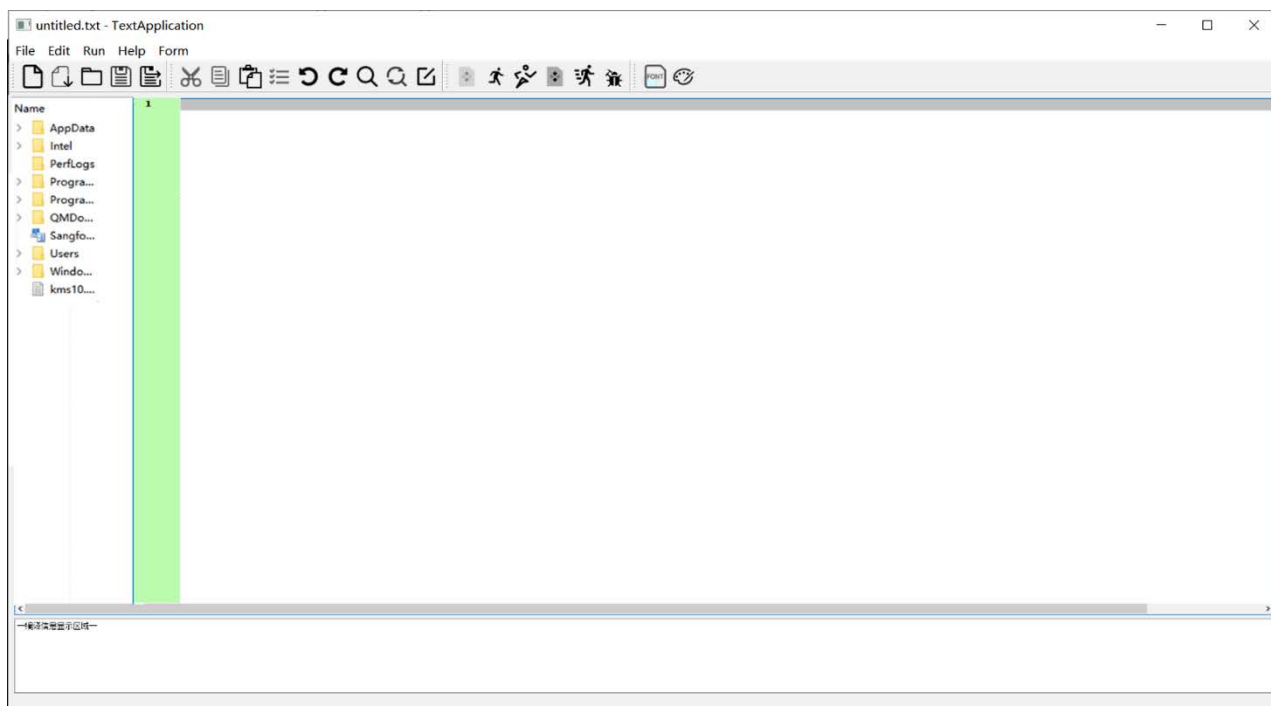
1 界面设计方案

1.1 界面整体布局设计方案

1.1.1 界面整体布局设计图



1.1.2 界面整体布局预览



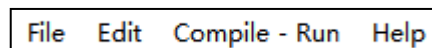
1.2 标题栏设计方案



应用图标 icon + 显示当前文件名（未保存时为 untitled.txt）

1.3 菜单栏设计方案

1.3.1 菜单栏整体设计方案

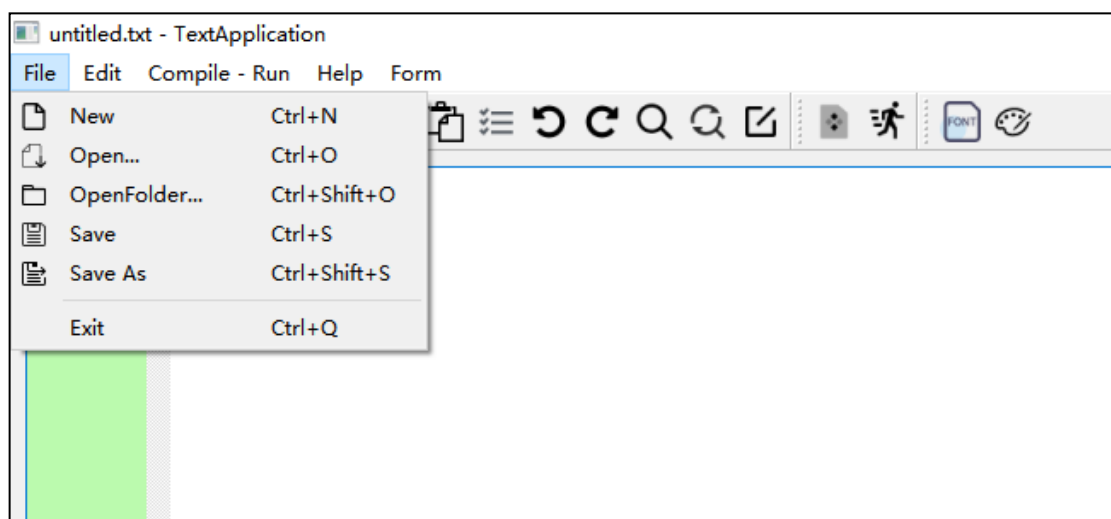


- (1) 文件选项
- (2) 编辑选项
- (3) 代码编译运行选项
- (4) 帮助选项

1.3.2 文件区域设计方案

文件区域实现功能：

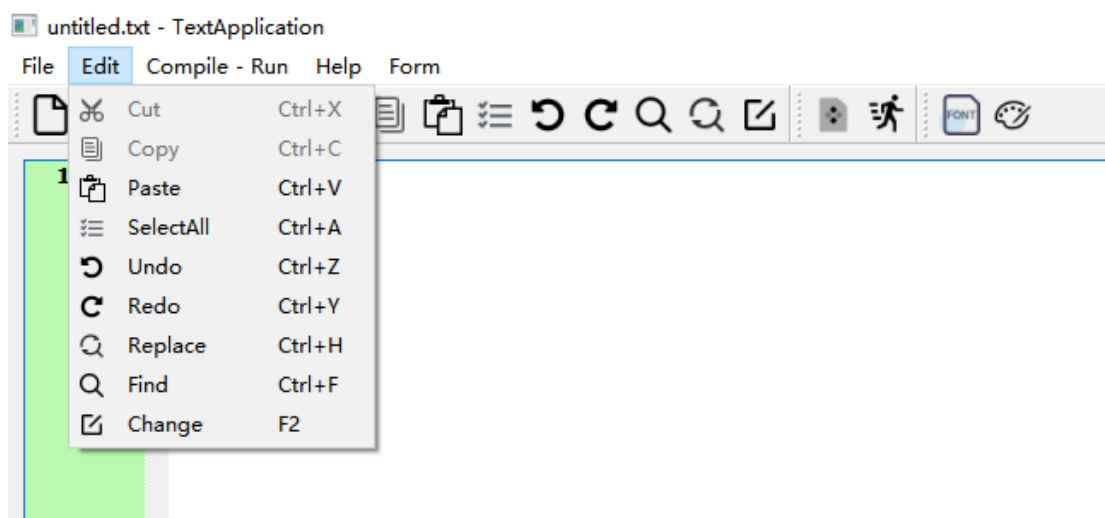
- | | |
|-----------|------------------|
| (1) 新建文件 | Ctrl + N |
| (2) 打开文件 | Ctrl + O |
| (3) 打开文件夹 | Ctrl + Shift + O |
| (4) 保存 | Ctrl + S |
| (5) 另存为 | Ctrl+ Shift + S |
| (6) 退出程序 | Ctrl + Q |



1.3.3 编辑区域设计方案

编辑区域实现功能：

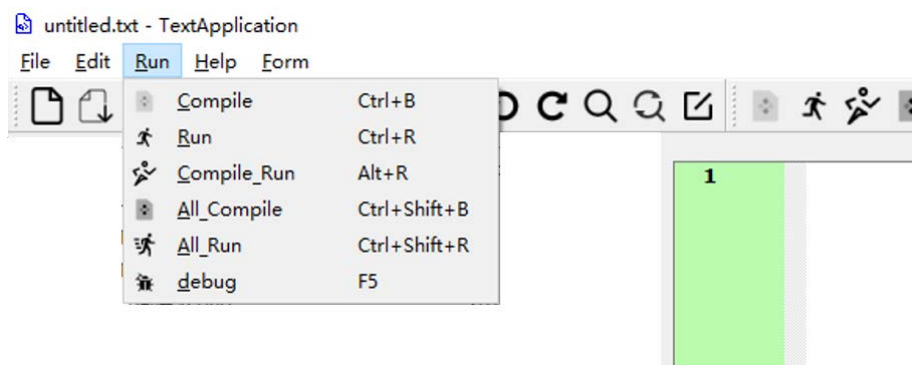
- | | |
|------------|----------|
| (1) 复制 | Ctrl + C |
| (2) 粘贴 | Ctrl + V |
| (3) 剪切 | Ctrl + X |
| (4) 全选 | Ctrl + A |
| (5) 撤销 | Ctrl + Z |
| (6) 重做 | Ctrl + Y |
| (7) 查找 | Ctrl + F |
| (8) 替换 | Ctrl + H |
| (9) 变量名重命名 | F2 |



1.3.4 编译运行区域设计方案

编译运行区域实现功能：

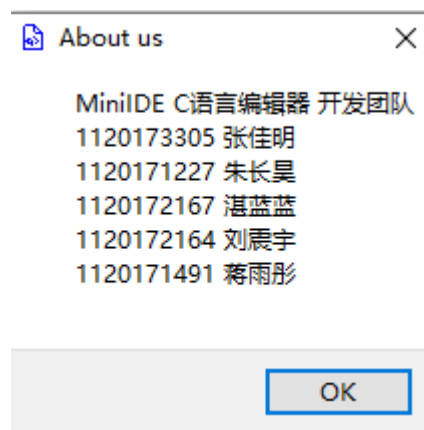
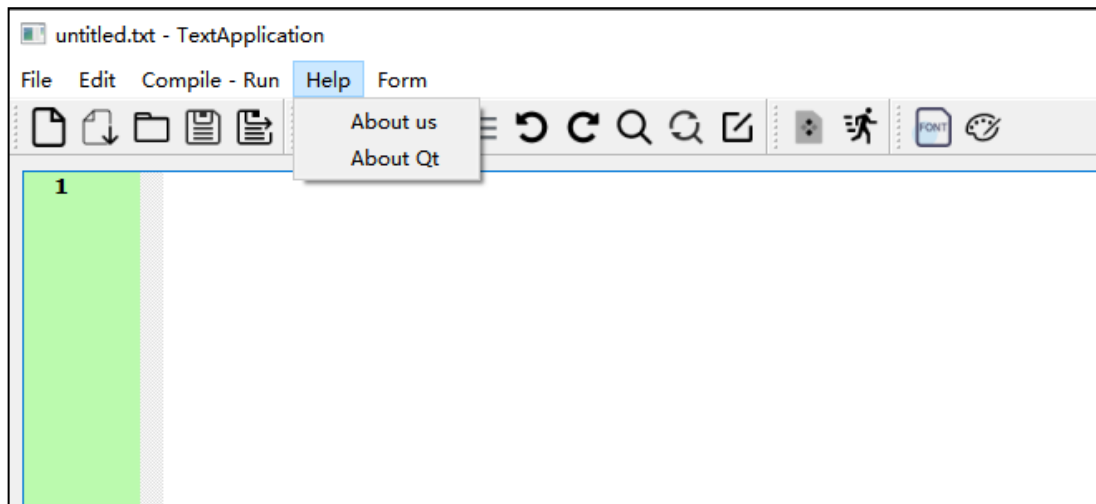
- | | |
|-----------|------------------|
| (1) 编译 | Ctrl + B |
| (2) 运行 | Ctrl + R |
| (3) 编译运行 | Alt + R |
| (4) 多文件编译 | Ctrl + Shift + B |
| (5) 多文件运行 | Ctrl + Shift + R |
| (6) 调试 | F5 |



1.3.5 帮助区域设计方案

帮助区域实现功能：

- (1) 关于我们
- (2) 关于 Qt



1.3.6 工具栏设计方案

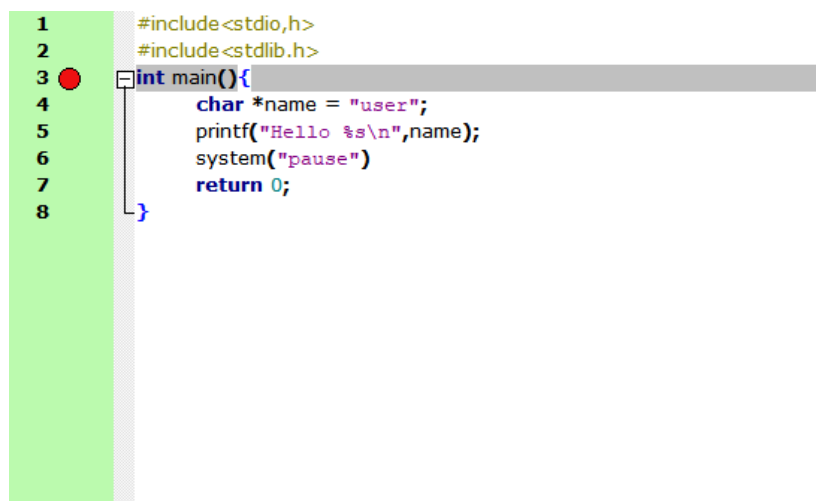


- | | |
|-----------|------------------|
| (1) 新建文件 | Ctrl + N |
| (2) 打开文件 | Ctrl + O |
| (3) 打开文件夹 | Ctrl + Shift + O |
| (4) 保存 | Ctrl + S |
| (5) 另存为 | Ctrl+ Shift + S |
| (6) 退出程序 | Ctrl + Q |

- | | |
|-------------|----------|
| (7) 复制 | Ctrl + C |
| (8) 粘贴 | Ctrl + V |
| (9) 剪切 | Ctrl + X |
| (10) 全选 | Ctrl + A |
| (11) 撤销 | Ctrl + Z |
| (12) 重做 | Ctrl + Y |
| (13) 查找 | Ctrl + F |
| (14) 替换 | Ctrl + H |
| (15) 变量名重命名 | F2 |

- | | |
|------------|------------------|
| (16) 编译 | Ctrl + B |
| (17) 运行 | Ctrl + R |
| (18) 编译运行 | Alt + R |
| (19) 多文件编译 | Ctrl + Shift + B |
| (20) 多文件运行 | Ctrl + Shift + R |
| (21) 调试 | F5 |

1.4 代码编辑区设计方案

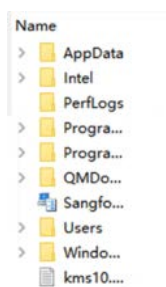


代码编辑区实现功能：

- | | |
|----------------|------------------------|
| (1) 调整字体大小 | Ctrl + mouseWheelEvent |
| (2) 关键字高亮 | |
| (3) 关键字补全 | |
| (4) 括号匹配高亮 | |
| (5) 括号、引号自动补全 | |
| (6) 自动缩进 | |
| (7) 代码自动整理 | |
| (8) 代码整体整理 | Ctrl + K + D |
| (9) 代码块折叠 | |
| (10) 多行代码注释 | Ctrl + Shift + / |
| (11) 行号显示 | |
| (12) 当前行阴影 | |
| (13) 添加断点 | |
| (14) 注释隐藏 / 显示 | |

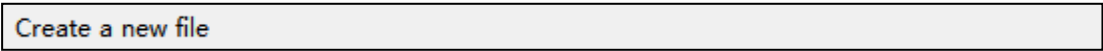
1.5 文件资源管理区设计方案

- (1) 文件资源管理区实现“打开文件夹”功能的可视化
- (2) 提供 可折叠展开的树形管理机制 和 “打开文件”功能的引用



(3)

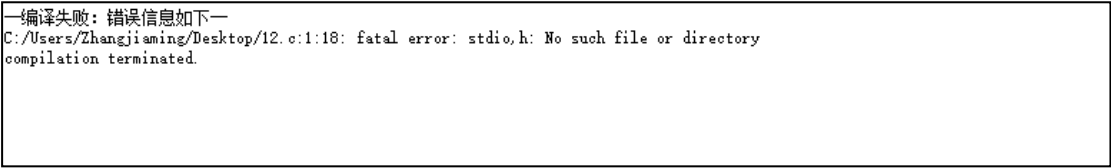
1.6 状态栏设计方案



状态栏提供功能：

- (1) 提供鼠标悬停所在功能的提示
- (2) 提供功能实现后的反馈提示
- (3) 提供编写时当前代码的行列显示

1.7 控制台设计方案



状态栏提供功能：

- (1) 提供当前文件编译结果的显示
- (2) 提供该程序运行的输入和输出

2 功能设计方案

2.1 程序功能表

程序功能表	文本编辑	复制	Ctrl + C
		粘贴	Ctrl + V
		查找	Ctrl + F
		替换	Ctrl + H
		撤销	Ctrl + Z
		重做	Ctrl + Y
		在状态栏显示当前行数	
		调整字体大小	Ctrl + wheelEvent
	代码编辑	关键字识别高亮	
		关键字联想提示、补全	Tab
		括号自动匹配高亮	
		括号自动补全	
		变量重命名	F2
		自动缩进	
		多行注释	Ctrl + Shift + /
		代码块折叠	
		对已编辑的函数进行高亮显示	
		代码跳转	Ctrl + mouseClick
		行格式排版	
		整体格式排版	F3
		注释显示 / 隐藏	F8
	代码运行	通过 gcc 对当前文件编译	Ctrl + B
		通过 gcc 对当前文件运行	Ctrl + R
		通过 gcc 实现多文件编译	Ctrl + Shift + b
		通过 gcc 实现项目运行	Alt + R
		在控制台显示编译结果	
		通过 gdb 对当前文件调试	F5
	文件管理	新建文件	Ctrl + N
		打开文件	Ctrl + O
		打开文件夹	Ctrl + Shift + O
		保存文件	Ctrl + S
		另存为文件	Ctrl + Shift + S
		多文件编辑	
		树形文件资源管理	

2.2 文本编辑

2.2.1 复制

(1) 功能描述

复制用户在编辑区内选中的文本；

(2) 快捷键：Ctrl + C；

2.2.2 粘贴

(1) 功能描述

粘贴用户剪切板中复制的内容；

(2) 快捷键：Ctrl + V；

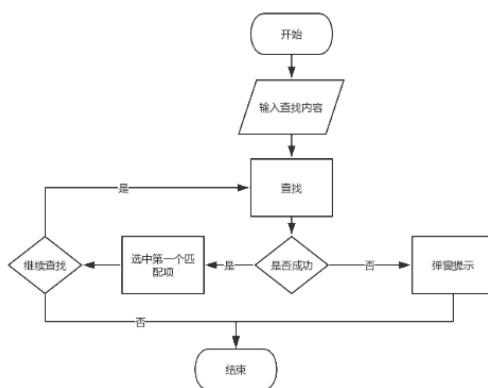
2.2.3 查找

(1) 功能描述

在文本编辑区内查找，可以选择全词/部分匹配、向前/向后查找。弹出对话框供用户输入待查找内容并设置查找选项。查找成功则在编辑区选中第一个符合条件的，失败则弹窗提示。

(2) 快捷键：Ctrl + F

(3) 流程图



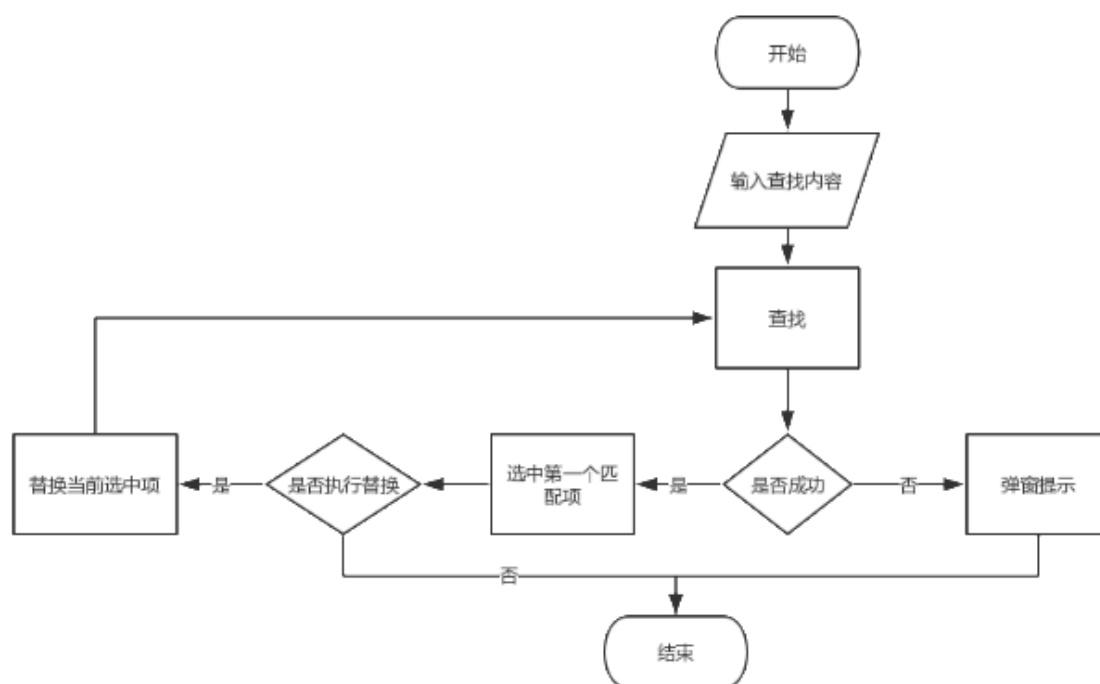
2.2.4 替换

(1) 功能描述

在文本编辑区内查找关键词并替换为新的关键词，替换选项包括替换选中/全部关键词，查找选项包括全词/部分匹配、向前/向后查找。弹出对话框供用户输入查找关键词、替换关键词并设置查找选项。查找成功则在编辑区选中第一个符合条件的，根据用户选择进行下一步操作，失败则弹窗提示。

(2) 快捷键：Ctrl + H

(3) 流程图



2.2.5 撤销

(1) 功能描述

撤销用户当前执行的操作；

(2) 快捷键：Ctrl + Z;

2.2.6 重做

(1) 功能描述

重新执行用户撤销的操作；

(2) 快捷键：Ctrl + Y;

2.2.7 显示当前行数和列号

(1) 功能描述

获取当前光标的行号和列号，并在状态栏显示；

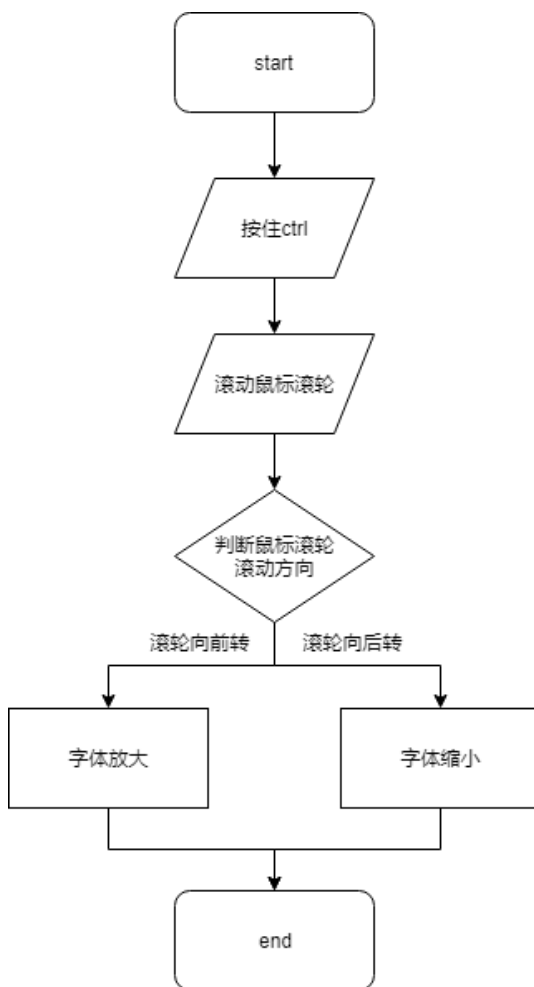
2.2.8 调整字体大小

(1) 功能描述

通过“Ctrl + wheelEvent”实现编辑区的字体大小调整；

(2) 快捷键：Ctrl + wheelEvent

(3) 流程图



2.3 代码编辑

2.3.1 关键字识别高亮

(1) 功能描述

检测编辑区文本是否为关键字，如果是则高亮显示

2.3.2 关键字联想提示、补全

(1) 功能描述

根据用户当前输入提供关键字提示列表，按 Tab 进行补全；

2.3.3 括号自动匹配高亮

(1) 功能描述

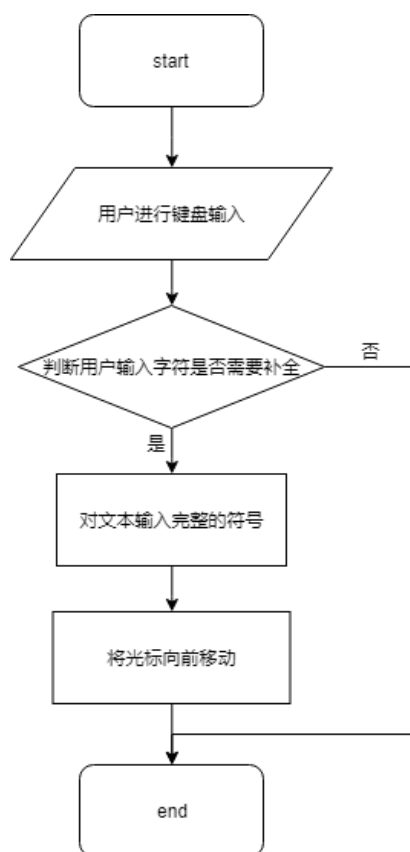
查找当前光标所在括号的所匹配的括号，并通过高亮显示；

2.3.4 括号自动补全

(1) 功能描述

根据用户输入的符号进行符号补全，并将光标移动至符号中间；

(2) 流程图



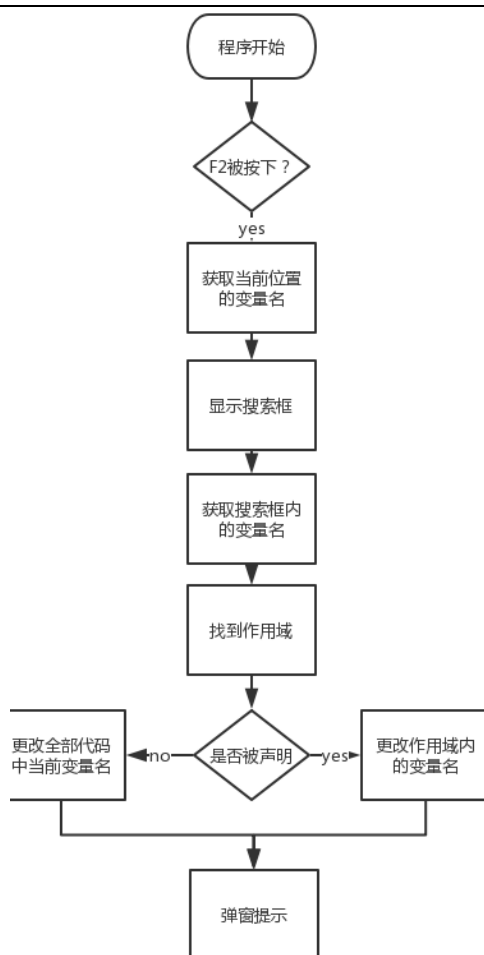
2.3.5 变量重命名

(1) 功能描述

将光标位于需要更改的变量上，更改变量所在作用域的名字。

(2) 快捷键：F2

(3) 流程图



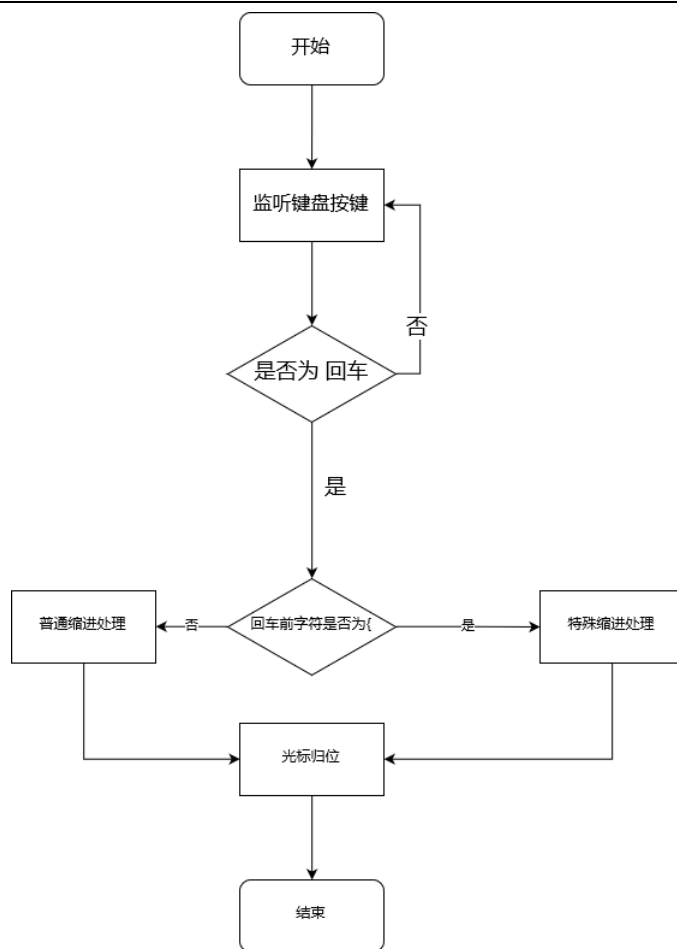
2.3.6 自动缩进

(1) 功能描述

当监听到键盘输入回车时，对此行进行缩进整理。具体实现缩进的情况有以下几个方面：

- 正常情况下，下一行的缩进与上一行保持一致
- 如果回车前一个符号为{，则下一行的缩进要在正常基础上多一次
- }所在的行缩进与其对应的{保持一致。

(2) 流程图



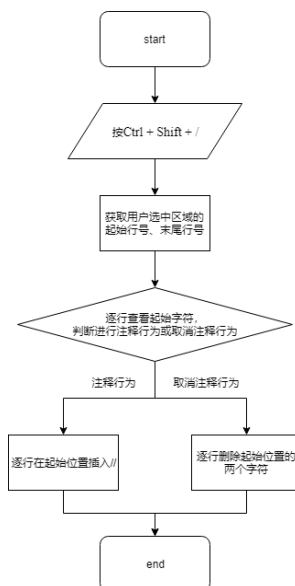
2.3.7 多行注释

(1) 功能描述

将用户选择文本区域逐行通过添加 // 进行注释；

(2) 快捷键：Ctrl + Shift + /

(3) 流程图



2.3.8 注释隐藏 / 显示

(1) 功能描述

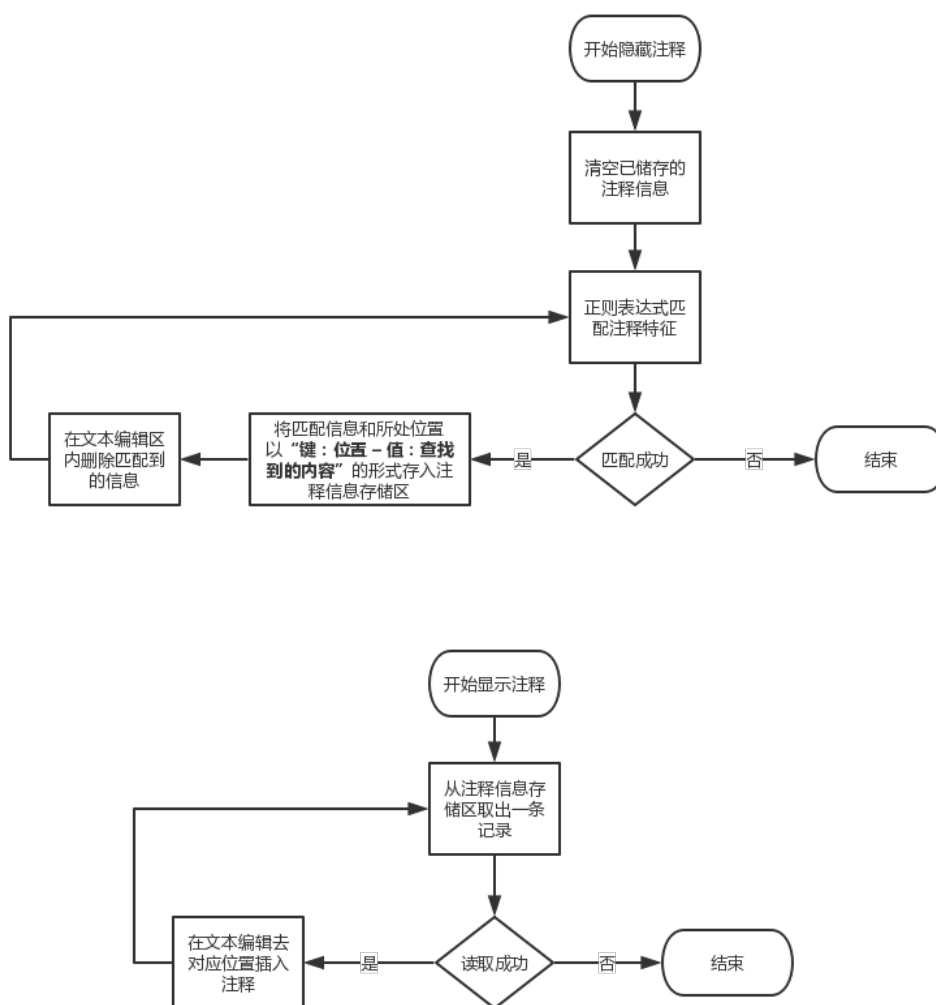
展示/隐藏当前编辑区内所有注释。

(2) 快捷键：F8

(3) 数据结构

map，键值对为“键：行号 - 值：注释”

(4) 流程图



2.3.9 代码块折叠

(1) 功能描述

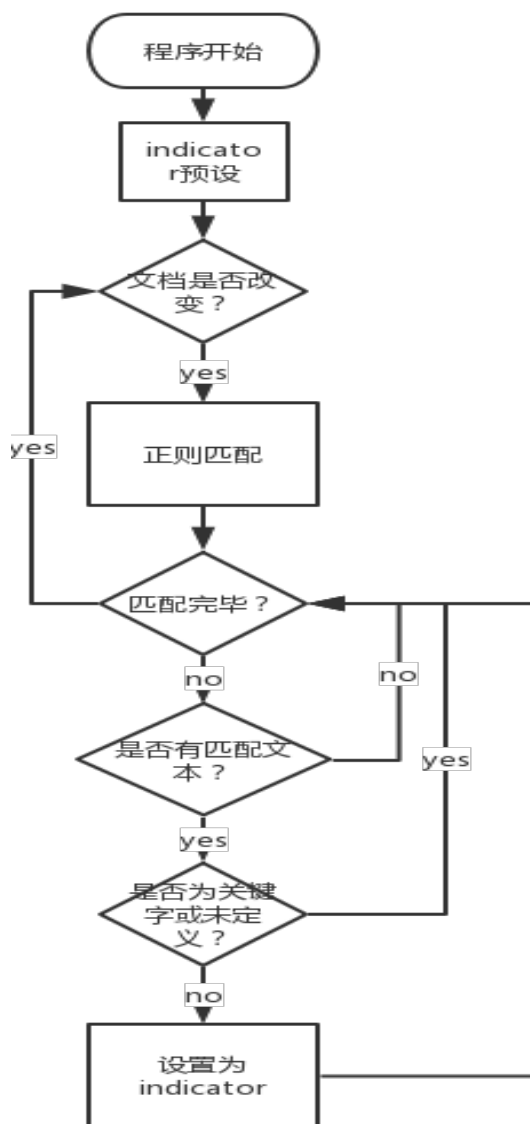
在编辑区左侧设置按钮，能够将代码进行折叠；

2.3.10 对已编辑的函数进行高亮显示

(1) 功能描述

对于已经有了声明的函数会进行高亮显示，未声明的函数不会高亮。

(2) 流程图



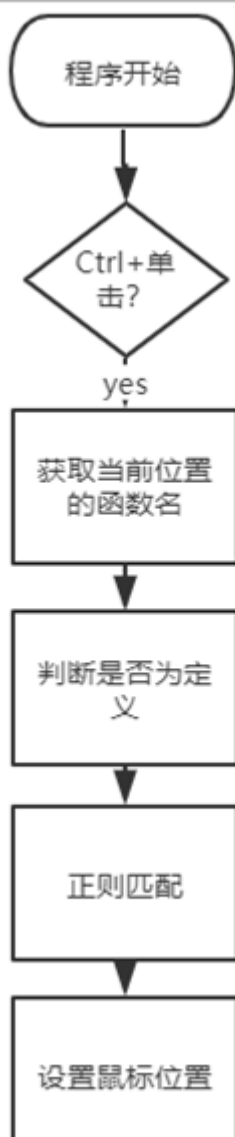
2.3.11 代码跳转

(1) 功能描述

Ctrl + 双击一个函数的声明或者具体引用时，会自动跳到他的定义；CTRL+单击一个函数的定义时，会跳转到他的声明。

(2) 快捷键：Ctrl + 双击

(3) 流程图



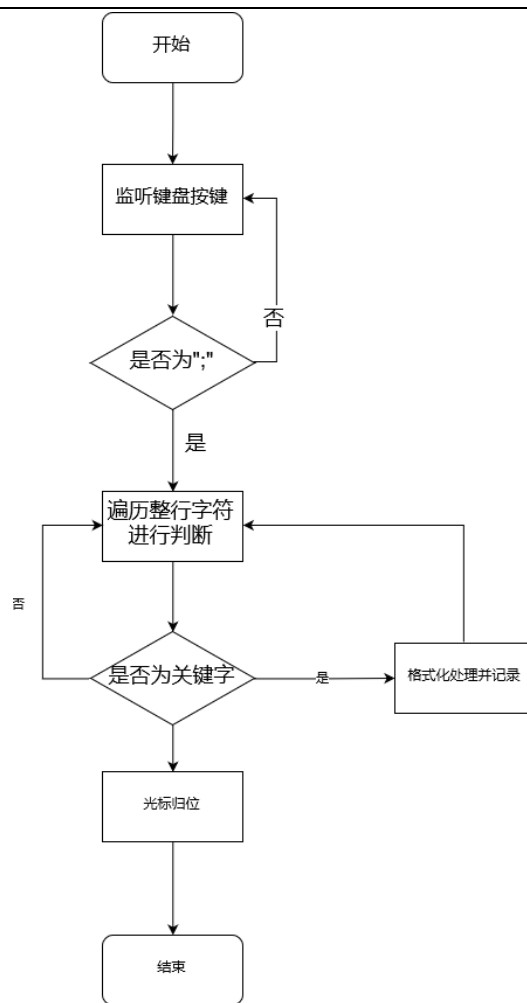
2.3.12 行格式排版

(1) 功能描述

当监听到键盘输入“;”时，对此行进行格式化整理。具体实现的格式化内容有以下几个方面：

- 对于“+”、“-”、“*”、“/”、“(”、“)”、“=”、“<”、“>”、“:”、“;” 这些符号，在符号的两侧使用空格进行分隔
- 一些特殊的符号组保留原有的格式，如：“++”、“--”、“->”、“+=”、“-=”等等

(2) 流程图



2.3.13 整体格式排版

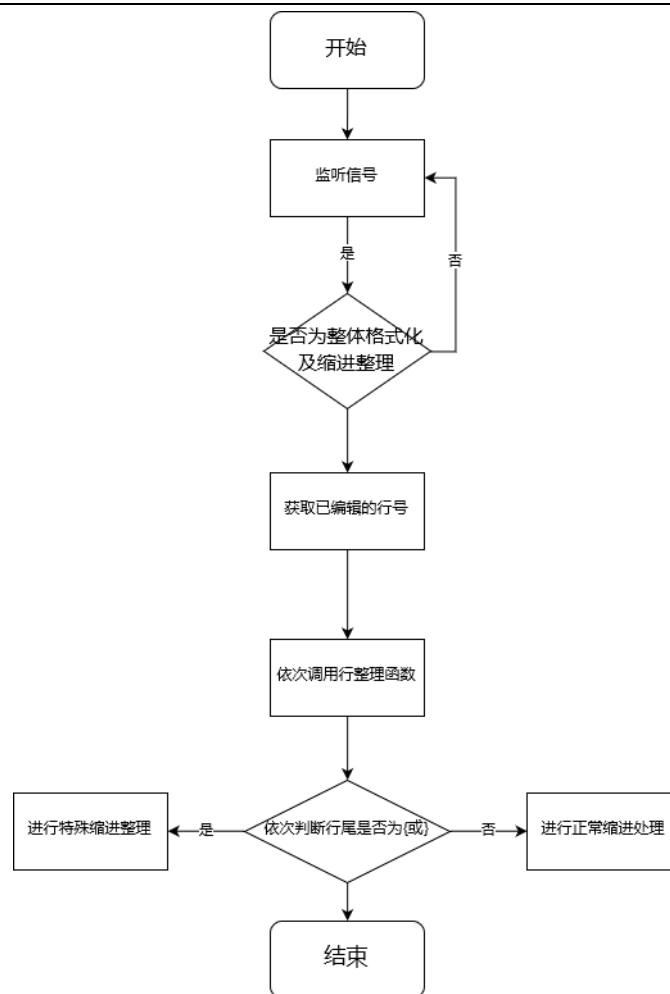
(1) 功能描述

当收到 `formatting all` 的信号时，对已编辑的全部代码进行格式化和缩进整理。具体实现的格式化内容有以下几个方面：

- 对于“+”、“-”、“*”、“/”、“(”、“)””、“=”、“<”、“>”、“;”、“,” 这些符号，在符号的两侧使用空格进行分隔
- 一些特殊的符号组保留原有的格式，如：“++”、“--”、“->”、“+=”、“-=”等等
- 行首保持相同缩进，如果有{}的嵌套则进行额外缩进

(2) 快捷键：F3

(3) 流程图



2.4 代码运行

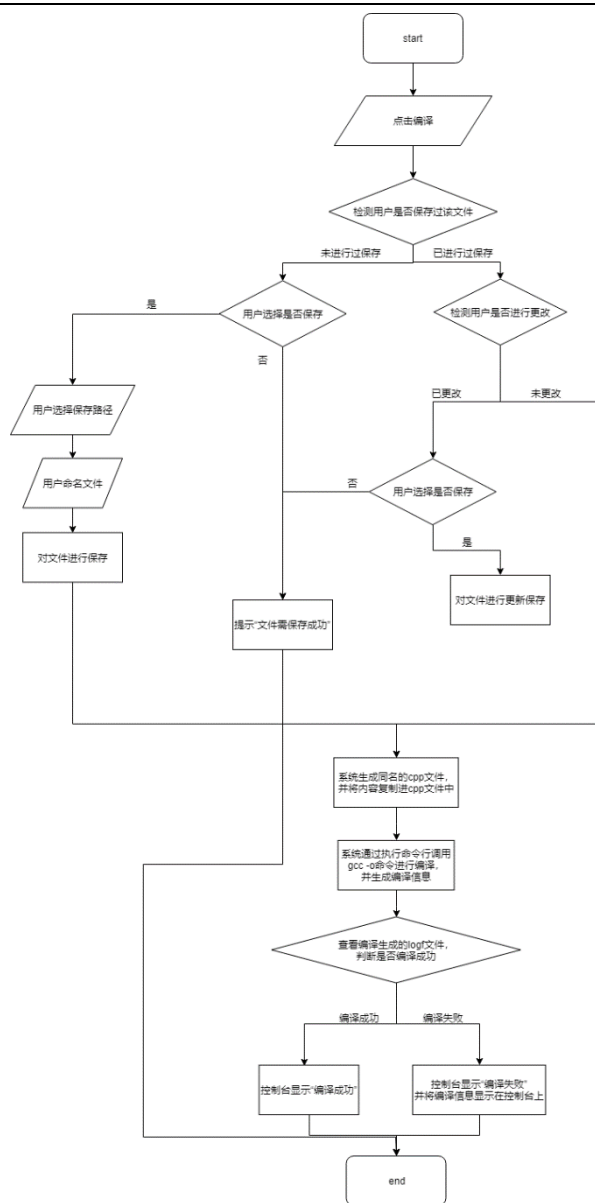
2.4.1 通过 gcc 对当前文件编译

(1) 功能描述

调用 gcc 编译器在当前文件路径下生成可执行.exe 文件；

(2) 快捷键：Ctrl + B

(3) 流程图



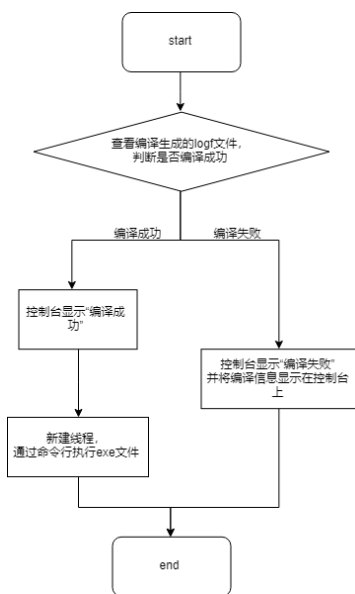
2.4.2 通过 gcc 对当前文件运行

(1) 功能描述

运行文件编译成功后生成的 exe 文件；

(2) 快捷键：Ctrl + R

(3) 流程图



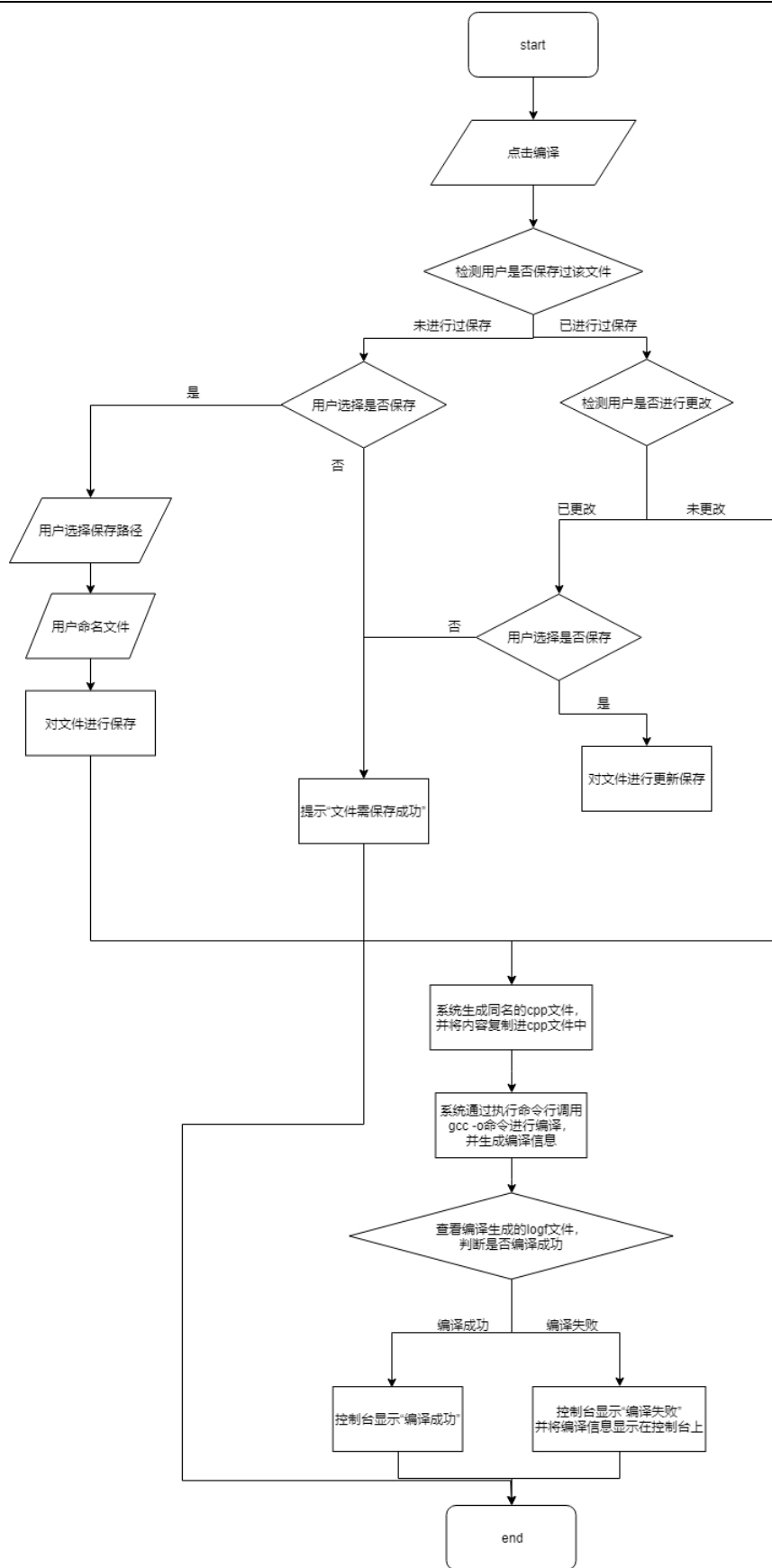
2.4.3 通过 gcc 实现多文件编译

(1) 功能描述

查看主文件引用的外部头文件，并调用 gcc 编译器将其同主文件在当前文件路径生成可执行.exe 文件；

(2) 快捷键：Ctrl + Shift + B

(3) 流程图



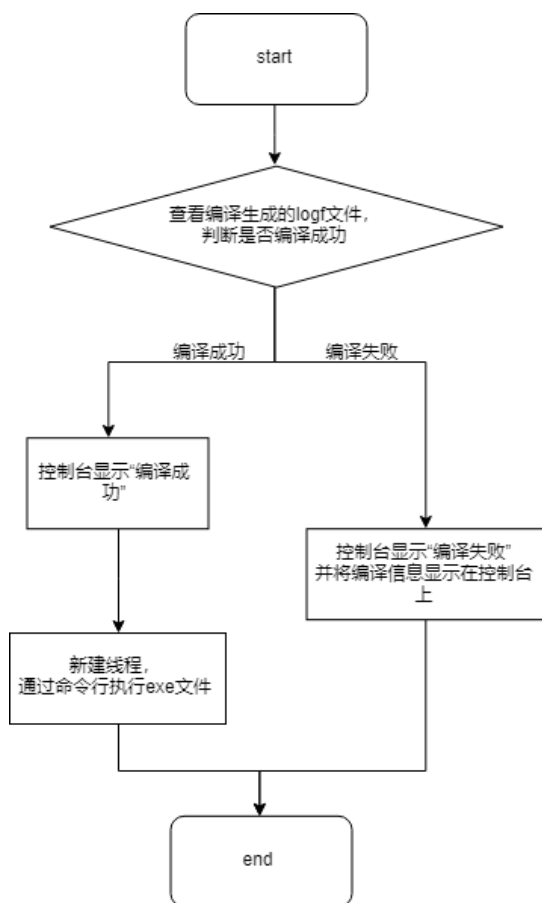
2.4.4 通过 gcc 实现项目运行

(1) 功能描述

运行文件编译成功后生成的 exe 文件；

(2) 快捷键：Ctrl + Shift + R

(3) 流程图

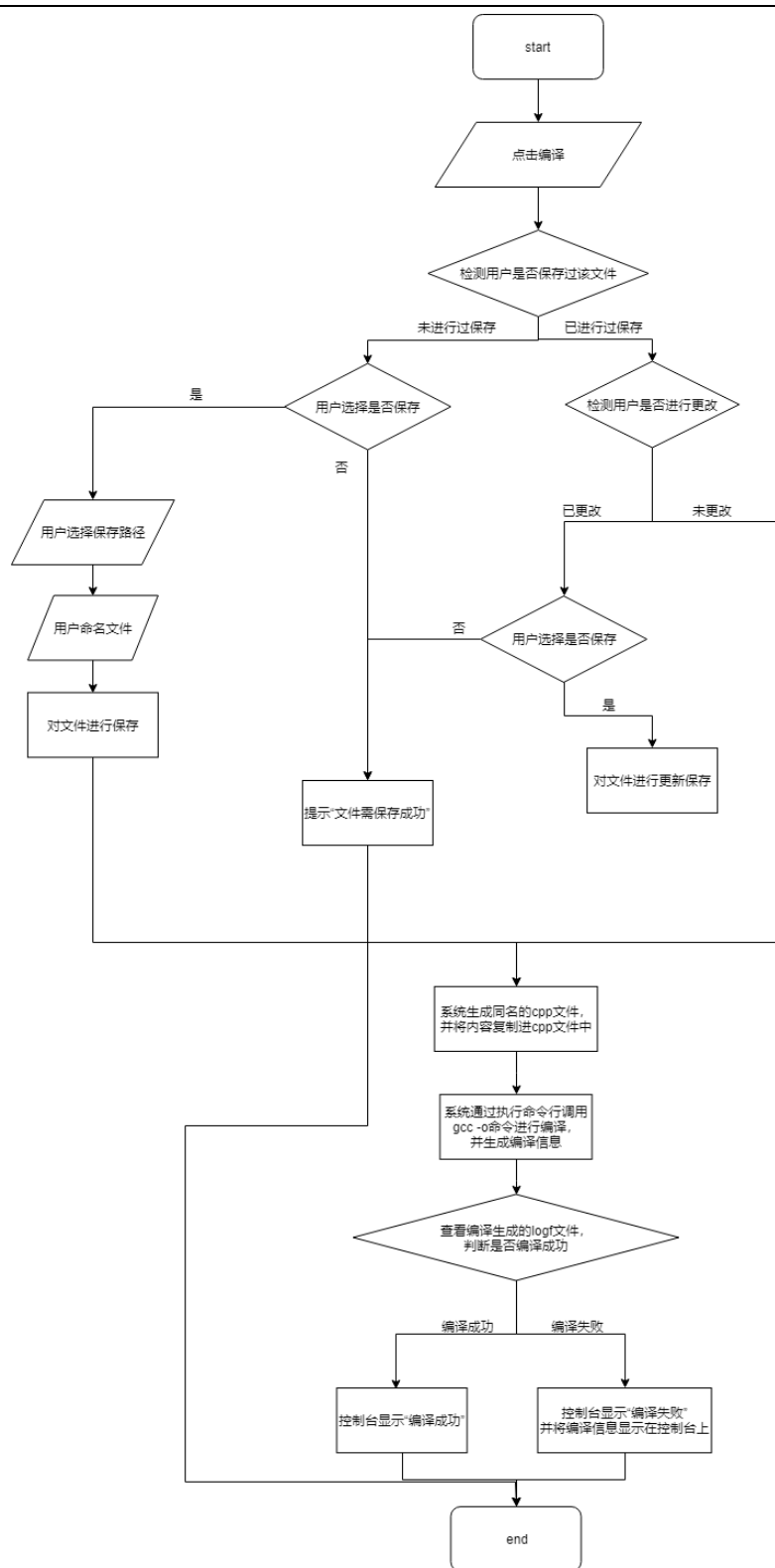


2.4.5 在控制台显示编译结果

(1) 功能描述

将编译结果显示在控制台；

(2) 流程图



2.4.6 通过 gdb 对当前文件调试

(1) 功能描述

编译当前代码，并启动 `gdb` 追踪代码的运行流程。将程序运行位置显示在编辑区侧边栏，将程序运行状态显示在调试对话框中。在程序运行过程中启用 `Debug` 模式，通过设置断点、添加查看变量可以分析定

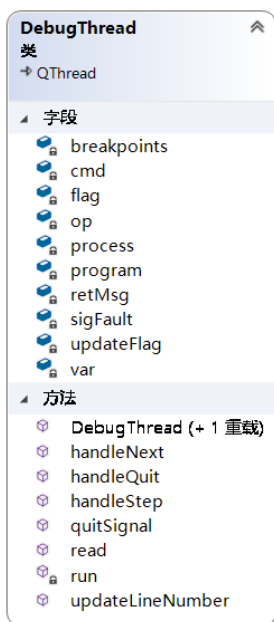
位异常发生的位置，以及追踪在运行过程中参数的变化。

提供的功能如下：

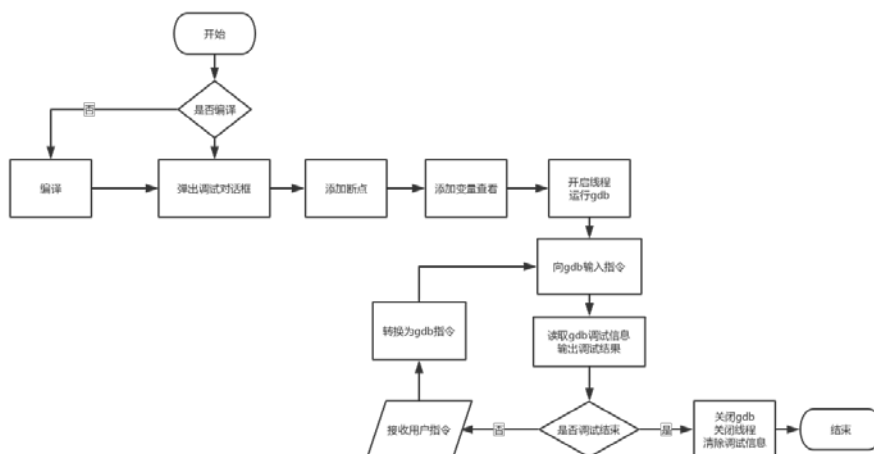
- 添加查看：接收用户输入的变量。在将来的调试过程中随语句执行显示设置的变量在当前语境的值；
- 开始调试：根据设置的断点呵查看变量信息开启 gdb 环境，进行调试。调试过程中不允许更改查看变量设置；
- 单步进入：执行下一条语句。如果该语句为函数调用，则进入该函数执行其中的第一条语句；
- 单步跳过：执行下一条语句。如果该语句为函数调用。则不会进入函数内部执行（即不会一步步地调试函数内部语句）；
- 继续执行：继续运行程序，直到遇到下一个断点；
- 退出调试：强制终止当前调试过程，退出 gdb 环境。

(2) 快捷键：F5

(3) 类图



(4) 流程图



3 文件管理方面

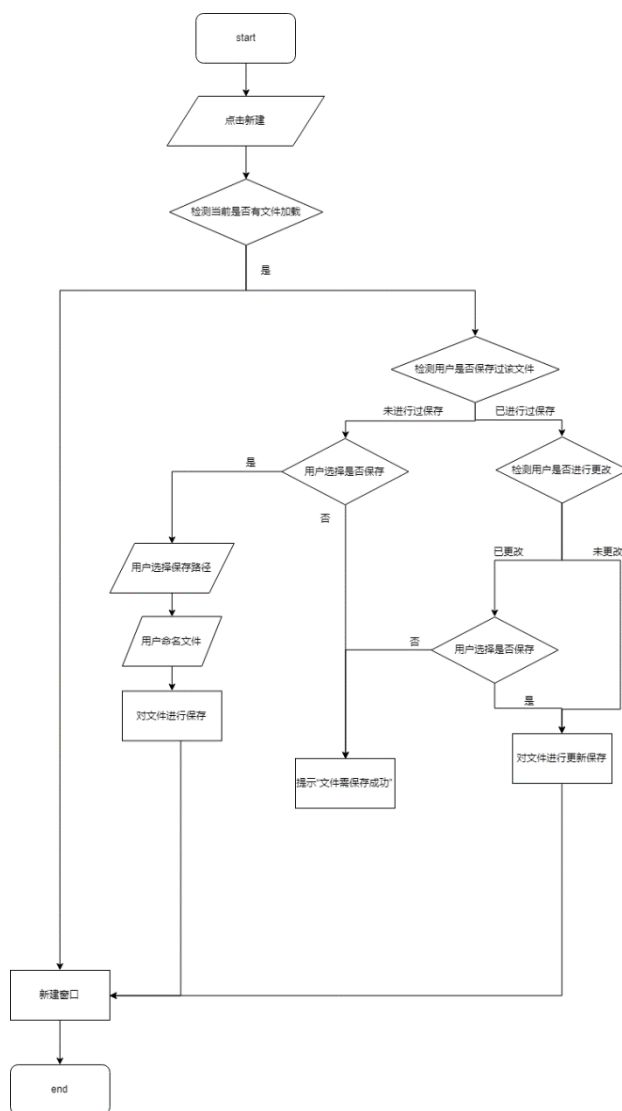
3.1.1 新建文件

(1) 功能描述

在编辑区新建空白文件编辑区（检测当前文件是否保存）；

(2) 快捷键：Ctrl + N

(3) 流程图



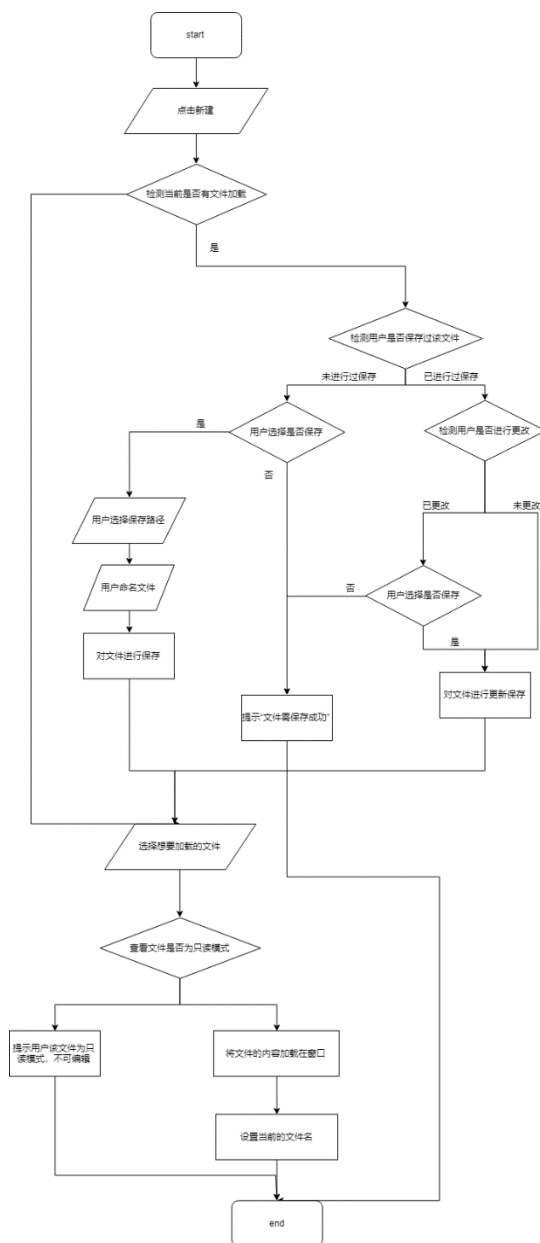
3.1.2 打开文件

(1) 功能描述

用户通过弹窗选择打开的文件，并将文件加载在文件编辑区

(2) 快捷键：Ctrl + O

(3) 流程图



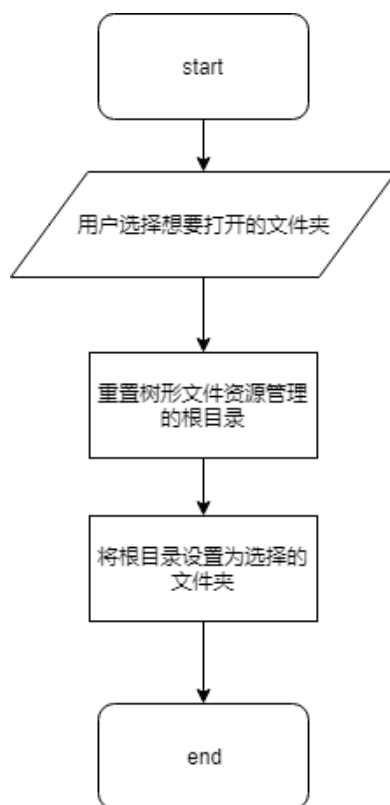
3.1.3 打开文件夹

(1) 功能描述

用户通过弹窗选择想要打开的文件夹，并将文件夹展现在树形文件资源管理区

(2) 快捷键：Ctrl + Shift + O

(3) 流程图



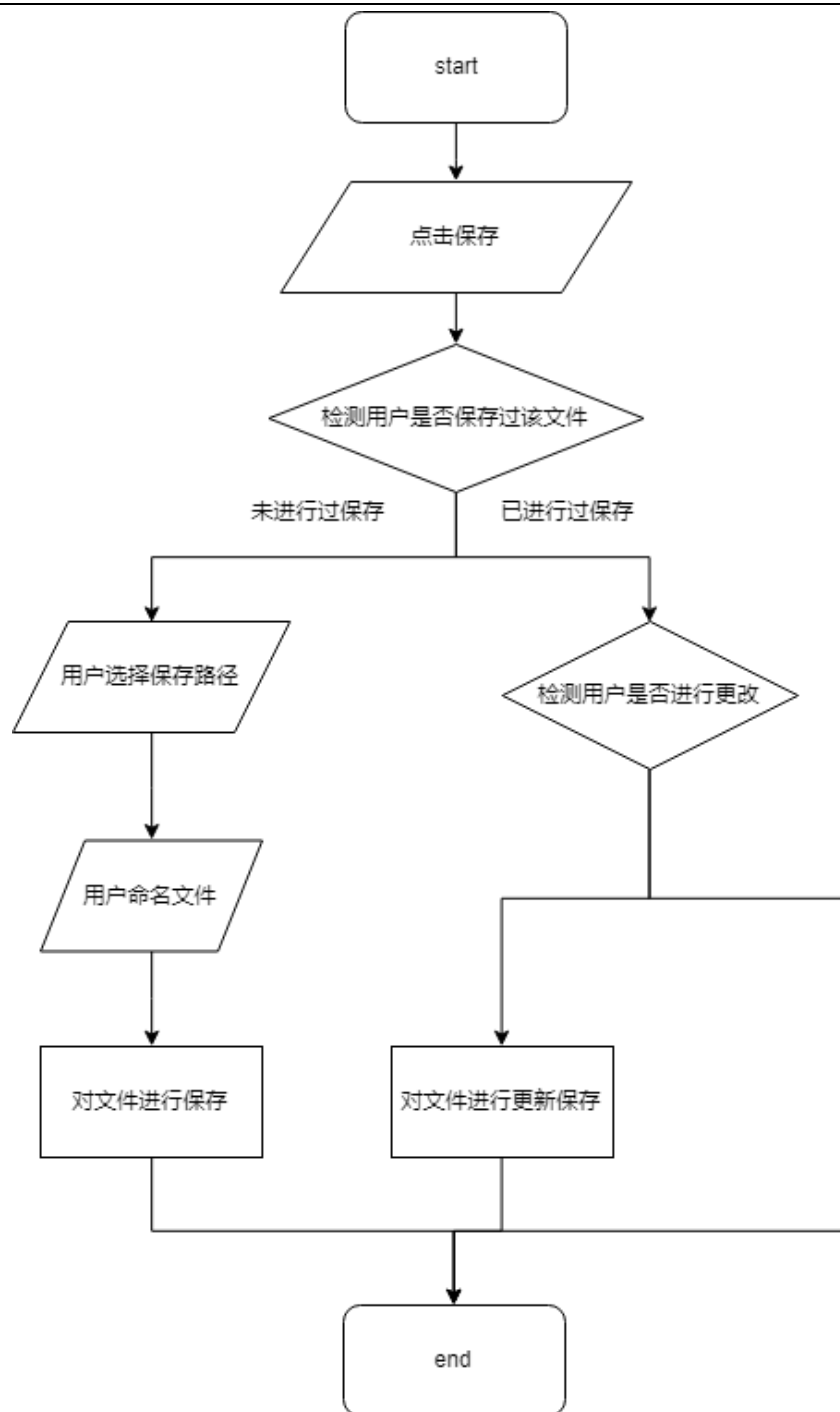
3.1.4 保存文件

(1) 功能描述

用户通过弹窗选择想要保存的文件夹，并将当前文件保存

(2) 快捷键：Ctrl + S

(3) 流程图



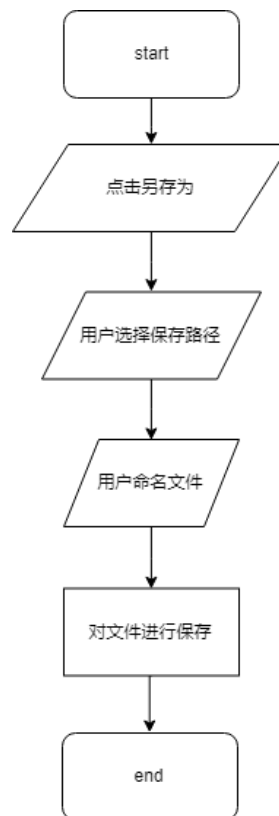
3.1.5 另存为文件

(1) 功能描述

用户通过弹窗选择想要保存的文件夹，并将当前文件保存；

(2) 快捷键：Ctrl + Shift + S

(3) 流程图

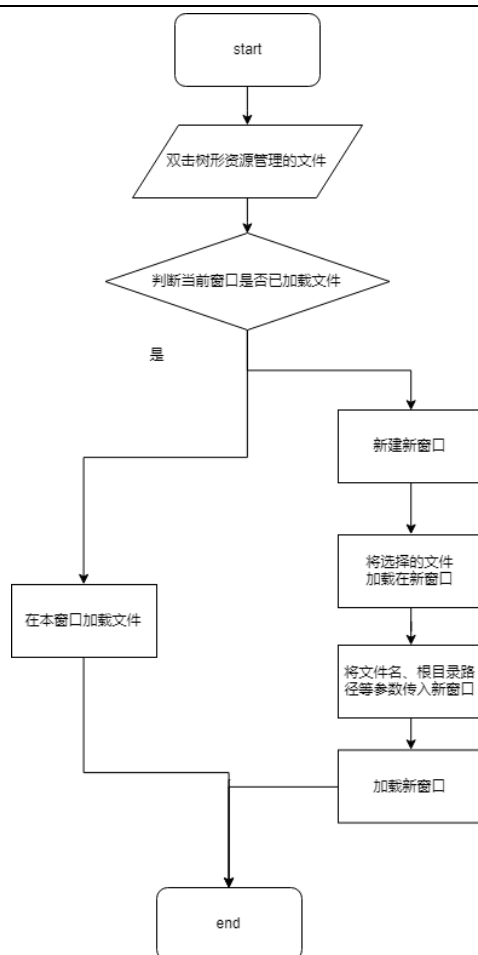


3.1.6 多文件编辑

(1) 功能描述

通过多个窗口实现多文件编辑

(2) 流程图

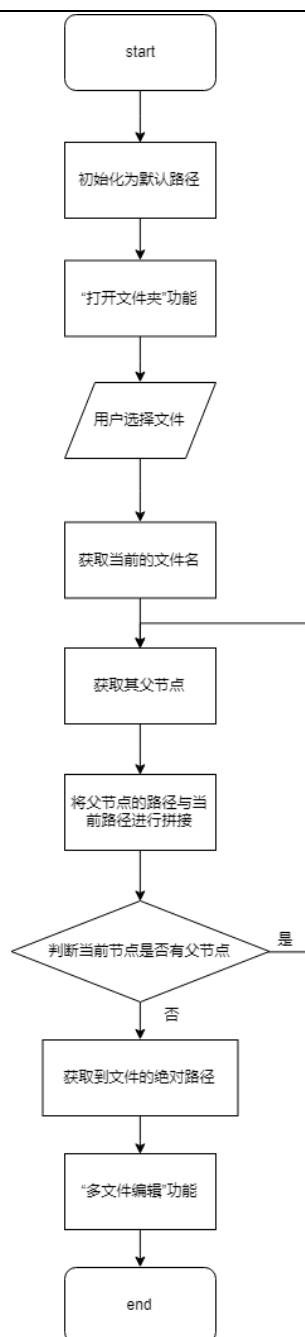


3.1.7 树形文件资源管理

(1) 功能描述

将用户选择打开的文件夹进行展示；

(2) 流程图



4 UML 设计

4.1 各功能设计

4.1.1 编辑器

编辑器采用 QsciScintilla 类，词法分析器采用 QsciLexerCPP 类，可以实现关键字识别高亮、括号自动匹配高亮、自动排版、代码块折叠等功能。

4.1.2 编译功能

编译功能使用 gcc。当用户点击编译按钮，系统会自动调用 cmd 里的 gcc 编译命令，通过 gcc 编译实现程序的编译，运行即直接运行编译好的.exe 文件。

4.1.3 调试功能

调试窗口继承 QDialog 类，窗口的按钮对应写入的指令。

调试功能使用 gdb。当用户点击调试按钮，系统首先调用编译功能，使用 gcc 对本文件进行编译。编译成功后，写入一个新线程进行调试。当用户点击不同的命令后，线程写入不同的指令进行调试，并解析 gdb 返回的信息显示到主窗口中。

4.1.4 窗口功能

窗口方面，mainWindow 继承自 QMainWindow 类，findDialog 继承自 QDialog 类，replaceDialog 类继承自 findDialog 类。

4.1.5 按钮监听功能

按键监听为自定义类，继承自 QObject。

4.2 UML 图

