# TinyIDE 集成开发环境软件设计文档

组长：林宇志

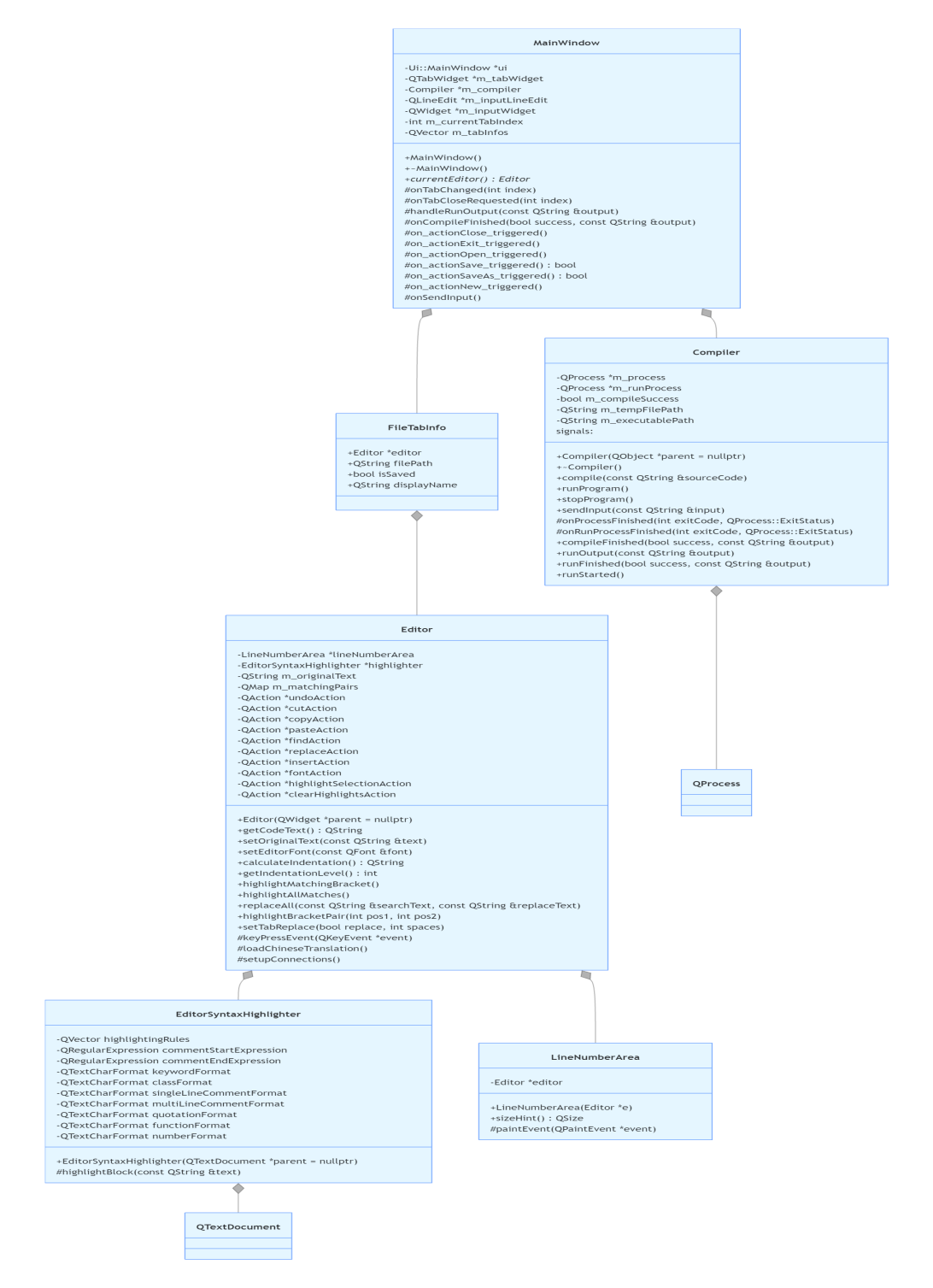
组员：李天灏、吕择琛、宋建宇、张骄阳

## 引言

## 1.1 文档目的

本文档详细描述 TinyIDE 集成开发环境的设计实现，包括系统架构、模块设计、类结构及核心功能，为开发小组提供技术参考，确保开发一致性。

### 1.2 范围

TinyIDE 是一款 C 语言集成开发环境，支持代码编辑、编译、运行及输入输出交互，面向初学者和小型开发任务。

## 2. 总体设计

### 2.1 系统架构

### 2.2 技术栈

* 框架：Qt 5.12.10（C++）
* 编译器：GCC
* 核心库：Qt Core、Qt Widgets
* 构建工具：qmake

## 3. 详细设计

### 3.1 编译器模块 (Compiler)

#### exported_image (2)3.1.1 类图

#### 3.1.2 功能说明

Compiler 类是 IDE 的核心模块，负责协调源代码的编译和执行过程：

* 管理两个独立进程：一个用于编译，一个用于程序运行
* 自动预处理源代码，添加必要头文件和行缓冲设置
* 使用临时文件存储源代码，避免对用户文件的直接修改
* 通过信号槽机制与 UI 层通信，传递编译结果和程序输出

#### 3.1.3 核心流程

1. **编译流程**：
   * 预处理源代码（添加头文件、设置行缓冲）
   * 创建临时文件并写入处理后的代码
   * 调用 GCC 编译生成可执行文件
   * 处理编译结果并发送信号
2. **运行流程**：
   * 检查编译状态和可执行文件
   * 启动程序并捕获标准输出 / 错误
   * 提供输入接口与运行中的程序交互
   * 程序结束后清理临时文件

### 3.2 其他关键类

#### exported_image (1)3.2.1 主窗口类 (MainWindow)

#### exported_image3.2.2 代码编辑器类 (Editor)

## 4. 核心功能实现

### 4.1 代码预处理机制

Compiler 类在编译前会对用户代码进行自动处理：

* 检查并自动添加必要的标准头文件 (stdio.h, stdlib.h)
* 在 main 函数开头插入行缓冲设置代码：setvbuf(stdout, NULL, \_IONBF, 0);
* 支持标准int main()和非标准void main()两种形式

### 4.2 进程管理

* 使用 Qt 的 QProcess 类管理外部进程，实现与 GCC 编译器的交互
* 分离编译进程和运行进程，允许独立控制
* 进程结束时自动清理临时文件，包括源文件和可执行文件

### 4.3 输入输出交互

* 实时捕获程序输出并通过信号发送到 UI 层
* 提供 sendInput 方法向运行中的程序发送用户输入
* 错误信息特殊标记显示，便于用户识别

## 5. 接口设计

### 5.1 信号接口

Compiler 类提供以下关键信号供 UI 层响应：

* compileFinished(bool success, const QString& output)：编译完成时触发
* runOutput(const QString& output)：程序有输出时触发
* runFinished(bool success, const QString& output)：程序运行结束时触发
* runStarted()：程序开始运行时触发

### 5.2 公共方法接口

* compile(const QString& sourceCode)：接收源代码并开始编译
* runProgram()：运行编译成功的程序
* sendInput(const QString& input)：向运行中的程序发送输入
* stopProgram()：强制终止当前运行的程序

## 6. 错误处理

* 编译失败时捕获 GCC 错误信息并显示
* 处理临时文件创建失败的情况
* 处理编译器启动失败（如未安装 GCC）的情况
* 处理可执行文件不存在的异常情况
* 进程超时和异常终止处理

## 7. 测试策略

* 测试单文件的处理逻辑
* 测试软件的UI交互逻辑
* 测试编辑器的功能逻辑
* 测试正常代码的编译和运行流程
* 测试错误代码的编译错误处理
* 测试程序输入输出交互功能
* 测试多文件的管理逻辑