

# 软件质量与评测技术

## 软件测试计划文档

C语言集成开发环境

By Team2

## 目录（鼠标左击可快速跳转到对应部分）

<b>1 范围</b>	<b>4</b>
1.1 标识	4
1.2 系统概述	5
1.3 文档概述	8
1.4 引用文档	11
<b>2 测试团队简介</b>	<b>11</b>
2.1 团队定位	11
2.2 测评职责	11
2.3 测试组规模	12
<b>3 总体测评要求</b>	<b>13</b>
3.1 测试目标	13
3.2 测试策略	13
3.3 测试管理	14
3.4 QA 部分测评要求	15
3.5 QC 部分测评要求	16
3.6 测试内容和测试方法	17
3.7 测试出入标准	20
3.8 定义测试重要级、优先级	21
3.9 数据采集要求	22
3.10 被测软件评价准则和方法	23
3.11 测试任务结束条件	23
<b>4 测试环境要求</b>	<b>24</b>
4.1 测试环境	24
4.2 测试人员	27
4.3 开发方配合要求	27
<b>5 测试内容</b>	<b>28</b>

5.1 功能性测试 .....	28
5.2 性能效率测试 .....	37
5.3 兼容性测试 .....	37
5.4 易用性测试 .....	38
5.5 可靠性测试 .....	38
5.6 信息安全性测试 .....	39
5.7 维护性测试 .....	39
5.8 可移植性测试 .....	40
5.9 用户文档集测试 .....	40
<b>6 项目管理 .....</b>	<b>41</b>
6.1 测试审核过程 .....	41
6.2 测试人力资源管理 .....	42
6.3 工作分工与进度安排 .....	42
6.4 使用 GITHUB 进行项目文档管理 .....	43
6.5 使用华为云 PROJECTMAN 进行测试用例管理 .....	44
6.6 测试人员激励办法 .....	46
6.7 测试团队的知识共享与培训 .....	48
6.8 跟踪与控制 .....	48
<b>7 测试风险管理计划 .....</b>	<b>52</b>
7.1 项目风险检查表 .....	52
7.2 定义风险等级 .....	54
7.3 风险应对方法 .....	54
<b>8 缺陷管理 .....</b>	<b>55</b>
8.1 定义缺陷优先级 .....	55
8.2 缺陷报告表 .....	55

# 1 范围

## 1.1 标识

- 1) 标题：基于 QT 的 C 语言集成开发软件测试计划；
- 2) 本文档适用的计算机软件：C 语言集成开发软件；
- 3) 术语和缩略语：

我们术语与缩略与针对本测试计划编写，主要涉及一些测试用例中需要用到的一些术语与缩略语。现在具体描述如下：

术语及其对应关系：

功能性测试设计术语：

界面测试	PageTest
文件相关操作的测试	FileTest
字符串查找功能测试	StringLookupTest
字符串替换功能测试	StringReplaceTest
括号自动匹配测试	AutoParenthesesMatchTest
鼠标操作	MouseOperationTest
关键字操作	KeyWordTest
键盘操作	KeyboardOperationTest
换肤	StyleChangeTest
鼠标键盘操作	KeyboardAndMouseTest
自动缩进	AutoIndentationTest
未保存检测	NotSaveTest
函数折叠	FunctionoFoldTest
Debug 功能	DebugTest
编译运行	CompileAndRunTest
支持多文件、文件树	Multi-FileTest

非功能性测试及其术语：

性能测试	PerformanceTest
兼容性测试	CompatibilityTest
易用性测试	UsabilityTest
可靠性测试	ReliabilityTest
信息安全性测试	InformationSecurityTest
用户文档集测试	UserDocumentTest

缩略语:

测试阶段缩略语:

U 代表单元测试,

S 代表系统测试,

I 代表集成测试,

R 代表回归测试,

A 代表验收测试。

测试方法缩略语:

S 代表静态测试,

D 代表动态测试,

B 代表黑盒测试,

W 代表白盒测试。

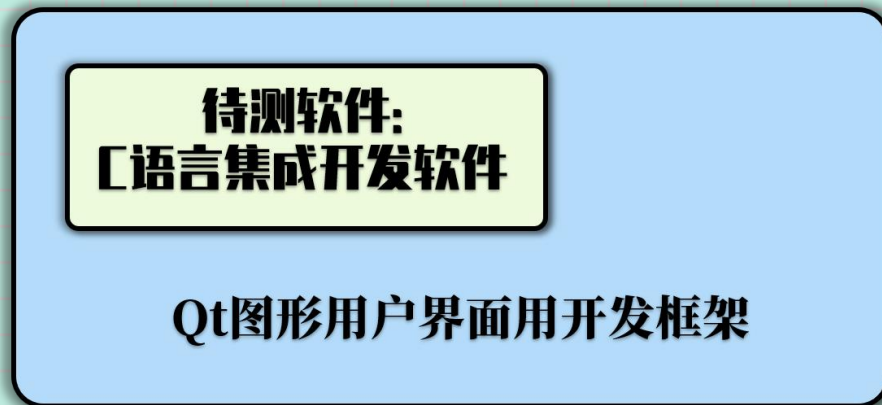
## 1.2 系统概述

### 1.2.1 功能概述

C 语言集成开发软件是一款简单的轻量级的 C 语言集成开发环境 IDE, 主要目的是为 C 语言编程者提供一个良好的编程环境, 帮助 C 语言编程者更好编写代码。除了满足一款 IDE 基本的代码编辑、程序运行、编译与文件读写功能,

还提供代码自动补全、自动缩进、全局查找与替换、关键字高亮和函数段折叠等进阶功能，可以极大地提升用户编写代码的效率和生产力。

软件外部交联图如下。



### 1.2.2 性能效率概述

C 语言集成开发软件基于 Qt 平台开发，通过便捷调用轻量简洁的函数库可以进行只能语法分析，响应灵敏快速，在规定条件下，运行软件各项功能时，具有较短的处理时间，并且在运行过程中不会占用大量的 CPU 资源和内存，即使是在较低配置的机器上也能流畅地运行。除了能完成基本的对于普通单个程序文件的运行，还支持代码量较大的单个文件以及多个程序同时运行的情况。

### 1.2.3 兼容性概述

C 语言集成开发软件在不同的软件系统中都有较好的相互集成的效果，在移植到不同系统平台的过程中不需要进行大规模地重新编写，软件开发以及部署的成本较低，经过一定的修改后可以稳定高效地运行在不同的系统平台，并且在多任务操作过程中，多个同时运行的软件不会出现冲突错误，可以方便地将数据共享给其他软件，不需要额外进行复杂地格式转换。

### 1.2.4 易用性概述

C 语言集成开发软件有直观简洁的可视化操作界面，无需事先进行复杂地安

软件质量与评测技术



装与配置过程，运行后即可直接编辑 C 语言程序。通过点击界面上的编译、运行等按钮图标即可一键实现相应功能。通过开发的进阶功能：代码自动补全、自动缩进、全局查找与替换、关键字高亮和函数段折叠等，大幅度地提升了用户在使用软件时的便捷度和代码编写的效率。

#### 1.2.5 可靠性概述

C 语言集成开发软件提供准确可靠的编译与调试功能，软件有较强的运行成熟性，可以避免掉运行过程中绝大多数的错误而导致的软件崩溃。并且在违背指定规则或者软件出现故障的情况下，仍然可以在运行时维持一定的基本响应功能。倘若已经受影响出现了软件失效的情况，也能在重新恢复后复原到崩溃前自动保存或记录的状态。

#### 1.2.6 信息安全性概述

C 语言集成开发软件具有基本的保护信息和数据的能力，以使未授权的人员或系统不能阅读或修改这些信息和数据，而不拒绝授权人员或系统对它们的访问。对用户、其他产品或者系统具有不同程度的授权类型与授权级别。软件会自行对不成功的操作记录进行预先定义，并明确规定达到该值时是否采取了具有规范性和安全性的措施来实现鉴别失败的处理。

#### 1.2.7 维护性概述

C 语言集成开发软件面对可能存在的修正、改进或者功能方向与规格变更都有较好的可维护性。软件的可理解性较高，开发过程中对特定功能的关键实现代码都做了清楚明了的注释说明，方便在维护过程中读懂开发时的思路。并且对不同的功能提供了模块化的编程，在维护过程中可以快速定位到对应的类或代码段。

### 1.2.8 可移植性概述

C 语言集成开发软件应用软件兼容性与系统兼容性较强,在移植到不同系统平台的过程中不需要进行大规模地重新编写,软件开发以及部署的成本较低。在目标环境下直接运行软件可执行代码的规范度也比较匹配,不需要为了适配不同的使用环境进行复杂的配置,同时也严格遵照了同类型软件相关的软件标准与约定,在同样的使用情况下可以方便地与同类软件相互替换。

### 1.2.9 软件其他信息

本次软件测试程序如下表。

表 1 被测软件程序清单

规模	开发环境/语言	运行平台	版本	重要度
3000行	Qt/C++	Windows	V3.0	重要

## 1.3 文档概述

本测试项目的主要文档包括:项目设计审查文档、需求分析审查文档、手工测试用例文档、手工测试需求文档、自动化测试用例文档、自动化测试需求文档、测试用例审查文档、缺陷报告文档、缺陷修复文档、质量报告文档、会议纪要、个人总结、团队总结。

### 1.3.1 手工测试用例文档编号规则:

示例: 044-F-AutoParenthesesMatchTest-U-D-B 测试用例文档

044 是手工测试用例的序列号

F 代表功能性测试,可选项为 F 与 NF, NF 代表非功能性测试

AutoParenthesesMatchTest 代表括号自动匹配,可选项在术语中可以找到。



U 代表单元测试。可选项为（单元测试 U，系统测试 S，集成测试 I，回归测试 R，验收测试 A）

D 代表动态测试。可选项为（静态测试 S，动态测试 D）

B 代表黑盒测试。可选项为（白盒测试 W，黑盒测试 B）

### 1.3.2 手工测试需求文档编号规则：

每个测试需求文档的命名规则：

示例：PageTest 测试需求文档

PageTest 代表界面测试，可选项在术语中可以找到。

测试需求文档每个小点的命名规则：

示例：001-F-Page

001 是手工测试需求的序列号，与手工测试用例的序列号一一对应。

F 代表功能性测试，可选项为（F 功能性测试与 NF 非功能性测试）

Page 是 PageTest 去掉 Test，代表界面需求。

### 1.3.3 自动化测试用例文档编号规则：

示例：001-NF-CodeTest-S-S-W 测试用例文档

001 是手工测试用例的序列号，自动化测试用例的序列号不同于手工测试用例的序列号，是重新开始记号的

NF 代表非功能性测试，可选项为 F 与 NF，NF 代表非功能性测试

CodeTest 代表代码检测。

S 代表系统测试。可选项为（单元测试 U，系统测试 S，集成测试 I，回归测试 R，验收测试 A）

S 代表静态测试。可选项为（静态测试 S，动态测试 D）

W 代表白盒测试。可选项为（白盒测试 W，黑盒测试 B）

#### 1.3.4 自动化测试需求文档编号规则：

每个测试需求文档的命名规则：

示例：AutoCodeTest 测试需求

AutoCodeTest 代表代码测试。

测试需求文档每个小点的命名规则：

示例：001-NF-AutoCode

001 是自动化测试需求的序列号，与自动化测试用例的序列号一一对应。

NF 代表非功能性测试，可选项为（F 功能性测试与 NF 非功能性测试）

AutoCode 是 AutoCodeTest 去掉 Test，代表代码测试需求。

#### 1.3.5 会议纪要编写规则：

示例：第一次会议纪要

之后命名为第二次会议纪要、第三次会议纪要……

#### 1.3.6 缺陷报告文档编号规则：

示例：Defect001-TestCase003 缺陷报告文档

Defect001 代表缺陷 1,001 位序列号，需要排列下去

TestCase003 代表缺陷 1 对应的手工测试用例编号为 003

#### 1.3.7 缺陷修复文档编号规则：

示例：DefectFix001 缺陷修复文档

DefectFix001 代表修复的第一个缺陷，001 与 Defect001 对应。

#### 1.3.8 测试用例审查文档编号规则：

示例：测试用例审查文档 1

之后命名为测试用例审查文档 2、测试用例审查文档 3……

## 1.4 引用文档

- 1) GB/T 25000.51-2016 《系统与软件工程 系统与软件质量要求和评价 (SQuaRE) 第 51 部分：就绪可用软件产品 (RUSP) 的质量要求和测试细则》
- 2) BITSTC-JG27-03-V02 软件测试计划编写规程 (民品)

## 2 测试团队简介

### 2.1 团队定位

本测试实践团队的团队定位是进行开发方测试，其中小组成员有 4 名是直接开发 C 语言 IDE 项目的开发人员，2 名没有直接参与 C 语言 IDE 项目的开发，所以对于我们而言，按人数的比例，我们整个团队应该属于开发团队，所以我们进行的是开发方的测试。开发方测试的主要测试目的是发现我们项目的不足，并对于可以修改的错误进行修改。需要澄清的是，我们的测试团队的主要职责是测试，修改程序仅仅占非常非常小的一部分，我们的主要任务仍然是尽可能的发现错误。

### 2.2 测评职责

刘睿泽：测试组长、资深测试工程师

陈晓璐：内审员、资深测试工程师

梁坤：资深测试工程师

吕泽超：资深测试工程师

张桢：资深测试工程师、实验室管理员

杨宜松：资深测试工程师

## 2.3 测试组规模

### 2.3.1 测试团队成员参与开发的比例

我们的测试是基于之前小学期我们自己开发的项目，因为本课程需要有 6 人组队，所以并不是所有小组成员都参与本项目的开发，统计参与开发的人员的数量如下：

小组总人数	参与开发的人数	未参与开发的人数
6 人	4 人	2 人

### 2.3.2 人员预期投入测试的工时与投入修改的工时的比例

因为我们测试小组的主要目的是测试，所以在测试组规模模块不具体划分软件测试人员与软件开发人员，而是令所有小组成员都作为软件测试人员对软件进行测试。但是我们的团队定位是开发方测试，所以要进行一定的错误代码修改的过程，所以我们规定人员预期投入测试的工时与投入修改的工时的比例，以此来规划成员投入到测试与修改错误上的时间。所占的比例以 10 份为最大单位。

小组成员	投入测试的时间占比	投入修改错误的时间占比
刘睿泽	9	1
陈晓璐	9	1
吕泽超	9	1
梁坤	9	1
张桢	10	0
杨宜松	10	0

## 3 总体测评要求

### 3.1 测试目标

测试目标在我们的测试项目中会在两处被提到，这两处测试目标的含义是不一样的，我在这里要澄清一下。第一处提到的测试目标是项目测试目标，它定义了整个项目的测试目的，它是制定测试计划与分析测试需求的前提。第二处提到的测试目标是测试用例中的测试目标，它是在每个测试用例中都会出现的一个测试用例元素。它定义了一个具体的测试用例的测试目的，也就是指明了测试用例具体是测试什么功能的。

#### 3.1.1 项目的测试目标

本项目的测试目标是：以尽可能多的发现软件的错误为目的，通过人工测试与测试工具相结合的方法，使用课上所学的黑盒测试用例设计技术与白盒测试用例设计技术，并结合课外知识，对自己编写的 C 语言 IDE 项目进行测试。

在测试过程中的目标是：测试过程中的目标是让每个小组成员都参与到测试管理过程中来，让每个人都能感受到测试管理的过程与方法。

测试完成后的目标是：对已经发现的错误进行修改，但是我们不会对所有错误进行修改，我仅对很少一部分错误进行修改。

#### 3.1.2 测试用例中的测试目标

测试用例中的测试目标会在每个测试用例文档中出现，可以去测试用例文档寻找。

### 3.2 测试策略

我们小组采用了 H 模型的测试过程模型作为我们测试的整体策略，H 模型

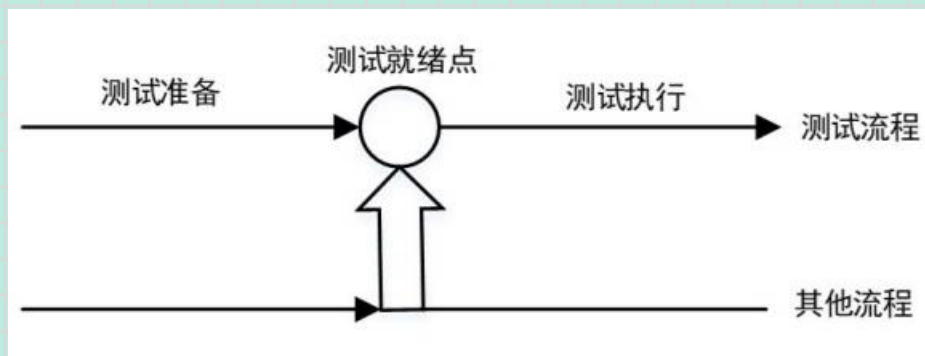


将测试活动完全独立出来，形成一个独立的流程，将测试准备活动和测试执行活动清晰地体现出来。

H 模型的测试流程分为：

测试准备活动：需求分析、测试计划、测试分析、测试编码、测试验证

测试执行活动：测试运行、测试报告、测试分析



应用 H 模型的意义在于：

1. 测试准备和测试执行分离，有利于资源调配，降低成本，提高效率
2. 充分体现测试过程（不是技术）的复杂性
3. 有组织、独立化的测试流程，有助于跟踪测试投入的流向

### 3.3 测试管理

测试管理包括 QA(质量保证)与 QC(质量控制)，首先我们对 QA 与 QC 在项目中的作用进行介绍：

QA 的目标是预防缺陷和错误的发生，QA 是属于防御性的方法。我们的测试项目中对 QA 所进行的活动有：需求分析审查、设计审查、测试用例审查等等。

QA 所进行的是静态测试和保证部分流程质量。

QC 的目标是找出缺陷和错误，QC 采取主动出击的方法，主动寻找肯能存在的错误。我们的测试项目中对 QC 所进行的活动有：测试、跟踪、监督等等。

其中对测试的跟踪与监督体现在回归测试中。QC 进行的是动态测试。

还有一种理解是，软件质量由组织、流程、和技术三个方面来决定，其中 QA 是从流程方面来保证软件质量，如协助 SEPG 制订流程、对流程的执行进行监督和审核、提出流程改进意见等等，而测试是从技术方面来保证质量，包括静态测试（如评审）和动态测试。

QA 是流程上的质量保证，用来对风险进行评估

QC 是最后一道质量屏障，用来确保发现尽可能多的问题，测试是 QC 中的具体措施。

下面我们对 QA 与 QC 分别进行测评要求的规定。

### 3.4 QA 部分测评要求

QA 部分测评前期主要是对需求分析以及设计进行审查，这是在测试未开始之前进行的审查，主要作用是保证需求分析文档的正确性以及设计文档的正确性，方便后续开展测试工作，在测试中期我们需要对测试用例进行审查，审查的内容就是该测试用例设计是否合格，以保证测试用例的质量。对于以上三方面的测试具体内容描述如下：

测试内容名称	测试内容标识	测试内容描述
需求分析审查	XQ	1. 需求文档描述不清晰，文档内容需完善细节调整 2. 文档内容混乱逻辑梳理不清很难看懂
设计审查	SJ	1. 缺少流程图或原型图描述，功能不清晰 2. 设计不清晰 3. 文档内容混乱逻辑梳理不清很难看懂

测试用例审查	YL	1. 测试设计的测试用例是否符合要求
--------	----	--------------------

3.5 QC 部分测评要求

因为测试是 QC 中的具体措施。所以此部分主要规定测试的要求。本次测试为系统测试,经过分析,测试内容包括 9 个方面,即功能性、性能效率、兼容性、易用性、可靠性、信息安全性、维护性、可移植性及用户文档集。具体见表。

针对本次测试的 9 个测试内容,采用的主要测试用例设计方法有:功能分解法、等价类划分法、边界值分析法、正交实验设计法、猜错法等。不同用例中所使用的方法将在每个用例中具体说明。

对测试过程中发现的软件缺陷进行一次回归测试,测试的最终结果以回归后的测试结果为准。

表 2 测试内容说明

测试内容名称	测试内容标识	测试内容描述
功能性测试	GN	产品的功能正确性、功能实现的完整性、功能的计算准确性、安全性等。
性能效率测试	XN	测试任务响应时间和资源利用率等。
兼容性测试	JR	测试软件间的共存性和互操作性。
易用性测试	YY	测试软件产品的界面交互能力、帮助文档的易理解性、易学性和易操作性等。
可靠性测试	KK	包括容错性、数据保护、系统在测试期间运行稳定性

		等。
信息安全性测试	XA	测试软件运行期间数据的保密性、完整性和真实性等。
维护性测试	WH	软件模块化、易分析性、易修改性、易测试等。
可移植性测试	YZ	测试软件的易安装性、易替换性等。
用户文档集测试	WD	测试用户文档的完备性、正确性、一致性、可操作性、易学易理解性等。
回归测试	HG	测试修改过的错误是否都正确，是否会产生新的错误
验收测试	YS	测试程序的主要功能是否全部正确

### 3.6 测试内容和测试方法

表 3 测试方法说明

测试内容名称	测试方法
功能性测试	根据需求规格说明，采用等价类划分、边界值分析、错误推测、场景法等黑盒测试技术，通过设计覆盖全部软件产品功能实现的测试用例的方法，从功能实现的功能正确性、功能实现的完整性、功能的计算准确性、安全性等方面对用户需求的软件产品全部功能性进行质量测试，并将功能性检测结果与用户需求中的功能要求比较，评价该软件产品功能是否符合用户需求和文档要求。

性能效率测试	根据需求规格说明，采用压力测试、负载测试、容量测试、强度测试等黑盒测试技术，通过使用通用或专用测试工具及设备 and 设计测试用例的方法，从响应时间、事物成功率、资源利用率等方面对软件产品的效率进行质量测试，并将效率测试结果与用户需求中的效率要求比较，评价软件产品效率是否符合有那刚好需求和文档要求。
兼容性测试	根据被测软件产品的特点，采用跨平台测试、数据设计测试等黑盒测试技术，通过设计测试用例，从硬件兼容性、软件兼容性、数据兼容性、网络协议兼容性等方面对软件产品的兼容性质量特性进行测试，并将兼容性的测试结果与用户需求中的兼容性要求比较，评价软件产品兼容性是否符合用户需求和文档要求。
易用性测试	根据需求规格说明，采用验证软件执行的各种操作/输入/问题/消息/结果的易理解性、检查文档和帮助信息、模拟演示界面的互操作性等黑盒测试技术，通过验证、检查和实际测试用例的方法，从易理解性、易学性、易操作性等方面对软件产品的易用性进行质量测试，并将易用性测试结果与用户需求中的易用性要求比较，评价软件产品的易用性是否符合用需求和文档要求。
可靠性测试	根据需求规格说明采用错误推测方法、边界分析、错误恢复、恢复性测试和稳定性测试等黑盒测试技术，通过设计软件产品异常处理及风险处理测试用例的方法，从容错性、



	数据保护、运行稳定性等方面对软件产品的可靠性进行质量测试，并将可靠性测试结果与用户需求中的可靠性要求比较，评价软件产品的可靠性是否符合用户需求。
信息安全性测试	根据被测软件产品的特点，选取正常或非正常测试用例，测试软件产品授权访问、权限控制等实际情况，并将新鲜安全性测试结果与软件产品新鲜安全性要求进行比较，评价信息安全的符合性。
维护性测试	根据被测软件的特点，采用接口测试技术、流程控制测试、数据流测试和模块黑盒测试技术，同验证、检查和测试用例的方法，从易分析性、易改变性、稳定性、易测试性和维护的依从性等方面对软件产品的维护性进行质量测试，滨江维护性测试结果与软件维护性测试要求进行比较，评价软件维护性的符合性。
可移植性测试	根据被测软件的特点，选取产品说明中列出的所有支持平台和系统，对软件进行安装，运行、卸载，并将可移植性测试结果与软件的可移植性要求进行比较，评价软件可移植性的符合性。
用户文档集测试	对被测软件的用户文档内容进行测试，重点验证检查其完备性、正确性、一致性、易理解性、易学性、可操作性。
回归测试	回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。自动回归测试将大幅降低系统测试、维护升级等阶段的成本。

验收测试	验收测试是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能如同用户所合理期待的那样。
------	--

### 3.7 测试出入标准

测试阶段	测试输入标准	要求	测试输出标准
原项目需求文档测试	用户与产品的需求定义、需求分析文档与相关技术文档	需求定义要准确、完整和一致，真正理解客户的需求	需求文档测试对应的测试用例文档与需求定义中出现的问题列表
原项目设计文档测试	产品设计说明书、系统架构文档与技术设计文档	系统结构要具有合理性、处理过程要具有正确性	设计文档测试对应的测试用例文档与设计定义中出现的问题列表
单元测试	源程序、编程规范与想要实现的功能的明确定义	要求满足功能实现的一致性与正确性	单元测试对应的测试用例文档
集成测试	通过单元测试的模块、编程规范与集成的规则	接口定义清楚且正确、模块集成在一起工作正常	集成测试对应的测试用例文档
系统测试	可以运行的软件	系统可以正常有	系统测试对应的

	包、测试环境与系统测试规则	效地工作，包括性能、可靠性、安全性、兼容性等	测试用例文档
验收测试	产品规格设计说明书、预发布的软件包	向用户表明系统能够按照预定要求那样工作	集成测试对应的测试用例文档、验收测试报告
回归测试	已经修复的好的代码、该代码段相关的测试用例文档	重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误	集成测试对应的测试用例文档

## 3.8 定义测试重要级、优先级

### 3.8.1 重要级

我们测试的重要级分为 4 个程度，分别为：最高、高、中、低。这四个重要级的定义如下：

最高：从功能上定义，他们是我们项目最重要的一些功能，是项目的核心功能，必须要实现的功能。

高：从功能上定义，他们是我们项目的核心功能的一些外延的功能，主要是帮助核心功能运行的一些功能

中：从功能上定义，他们是我们项目的次重要的功能，主要包括进阶功能，有用这些概念可以更加方便用户的使用，但是没有这些功能也无伤大雅。

低：从功能上定义，他们是我们项目中的不重要的功能，是否实现对整个项目几乎无影响。

### 3.8.2 优先级

我们测试的优先级分为 4 个程度，分别为：最高、高、中、低。这四个优先级的定义如下：

最高：确定此版本是否可测的测试用例，此部分测试用例如果 fail 会阻碍大部分其他测试用例的验证。如果此类测试用例不通过，那么整个程序的可能无法正确执行。

高：测试的功能是最常执行的功能，以保证功能性是稳定的；包括基本功能测试，和重要的错误、边界测试

中：更全面地验证功能的各个方面，包括异常测试，边界、中断、断网、容错、UI 等测试用例

低：不常常被执行，包括性能、压力、兼容性、稳定性、安全、可用性等等。

## 3.9 数据采集要求

测试过程中要采集如下方面的数据，以评价被测软件和测试质量：

- 1) 被测软件数据：代码规模、测试项个数、可测项个数；
- 2) 用例数据：生成的测试用例总数，每个测试项中不同测试类型的用例数，执行了的用例数，未执行的用例数；
- 3) 缺陷数据：测试发现的缺陷数，并按类型和重要度进行划分；典型缺陷及影响；
- 4) 管理数据：工作量、进度；

### 3.10 被测软件评价准则和方法

在软件测试报告中从如下方面对软件进行定性评价：

- 1) 功能性方面：系统中存在未修改的“关键缺陷”大于等于 1 个或存在未修改的“重要缺陷”大于等于 3 个时视为“不通过”，其它情况视为“通过”；
- 2) 性能效率方面：性能效率符合指标要求则“通过”，否则“不通过”；
- 3) 兼容性方面：同功能性方面；
- 4) 易用性方面：同功能性方面；
- 5) 可靠性方面：同功能性方面；
- 6) 信息安全性方面：同功能性方面；
- 7) 维护性方面：同功能性方面；
- 8) 可移植性方面：同功能性方面；
- 9) 用户文档集方面：同功能性方面；
- 10) 系统总体评价：系统各项质量要求均为“通过”时则系统视为“通过”，除功能性要求外只有 1 项质量要求为“不通过”，其它为“通过”时则系统也视为“通过”；功能性要求为“不通过”是则系统视为“不通过”，除功能性要求外多于 1 项质量要求为“不通过”则系统视为“不通过”。

### 3.11 测试任务结束条件

完成了如下工作则测试任务结束：

- 1) 所有测试项都达到了测试终止要求，且发现的每个软件缺陷都作了归



零处理（进行了修改并通过回归测试，或没修改但给出了降低风险的措施，或进行风险分析后可以接受不修改带来的风险故不作修改。）且提交了相应的测试文档；

- 2) 完成了测试协议中所规定的所有工作；
- 3) 通过了测试相关评审，且按照评审意见中的建议对工作进行了完善。

## 4 测试环境要求

### 4.1 测试环境

#### 4.1.1 软件项

表 4 软件项

序号	软件项名称	版本	用途
1	C语言集成开发软件	代码审查后版本	被测软件
2	Qt图形用户界面应用开发框架	V6.1.2	应用软件编译器、代码连接器
3	操作系统	Win10/Win11	操作系统

#### 4.1.2 硬件和固件项

测试硬件的标准配置：

操作系统：Windows11

系统类型：64 位操作系统，基于 x64 的处理器

处理器：Intel(R) Core(TM) i5-10200H CPU

内存：16.0GB

CPU: NVIDIA

测试硬件的最低配置：

操作系统：Windows10

系统类型：64 位操作系统，基于 x64 的处理器

处理器：Intel(R) Core(TM) i5-10200H CPU

内存：8.0GB

CPU: NVIDIA

测试硬件的最高配置：

无最高配置，配置越高越好

测试人员的硬件与固件项

表 5 测试人员硬件和固件项

序号	硬件和固件项名称	用途	最低配置要求
1	PC机（客户端）	作为载体承担待测软件的运行与测试	cpu: i5-5200U@2.20GHz 硬盘：1TB 内存：8G

			分辨率：1366x768 显卡：930M
--	--	--	-------------------------

#### 4.1.3 网络环境

测试人员	网络环境
刘睿泽	以太网
陈晓璐	以太网
吕泽超	以太网
梁坤	以太网
张桢	以太网
杨宜松	以太网

#### 4.1.4 数据准备

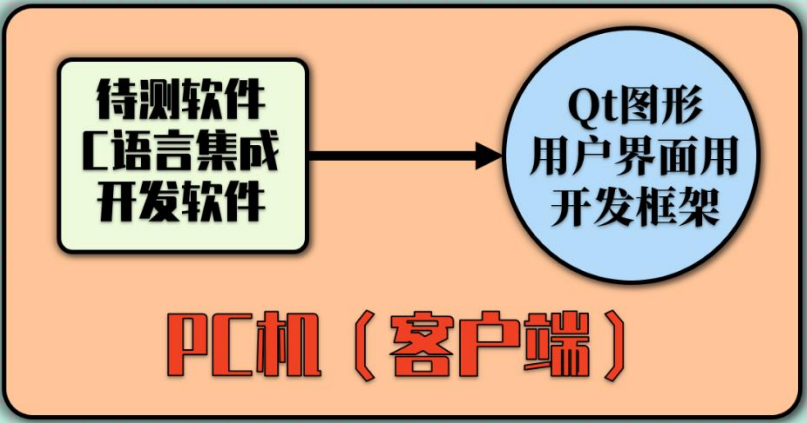
本 C 语言集成开发软件主要对文件进行操作，所以需要的数据为 C 语言文件或者 txt 文件。文件可以放置到任意位置。

#### 4.1.5 测试工具

主要介绍我们软件运用的第三方的测试工具。

#### 4.1.6 测评运行环境

根据测试要求，建立 C 语言集成开发软件配置项测试环境，测试环境示意图如下：



4.2 测试人员

杨宜松 张 桢

刘睿泽 吕泽超

陈晓璐 梁 坤

4.3 开发方配合要求

经协商，开发方在如下阶段进行相关配合：

表 6 开发方配合内容

阶段	内 容	开发方人员
测试需求分析	介绍软件功能	陈晓璐
测试策划	各时间节点和配合要求讨论	刘睿泽
测试设计和实现	提供检测仪器，用例讨论	吕泽超、刘睿泽
测试执行	结果确认，确认问题签字，修改软件，填	梁坤

	写更改单，回归测试配合	
测试总结	确认评审时间	刘睿泽

## 5 测试内容

### 5.1 功能性测试

#### 5.1.1 菜单检查

测试项名称	基本菜单界面	测试项标识	
追踪关系			
测试项描述	<p>菜单栏，是由下拉菜单组成的，点击每一个菜单都会出现一个下拉菜单，里面包含具体的功能。菜单栏最重要的是文件、编辑两个菜单。</p> <p>工具栏，是由图标组成的，每一个图标代表一个常用的功能，点击图标就可触发响应功能。工具栏最重要的是新建文件，打开文件，保存文件与编译运行。</p>		
软件设计约束	无。		
测试内容	<p>1.直接运行程序，点击菜单栏的具体菜单，查看是否有下拉菜单；</p> <p>2.点击功能，看能否实现该功能。</p> <p>3.将鼠标悬停于图标，查看文字；鼠标点击图标查看是否能进行功能跳转。</p> <p>4.首先查看左部的文件视图栏和底部的编译输出框是否可以正常显示；其次点击按钮查看是否能隐藏和再现</p>		



	5.打开任意包含中英文和符号的文件，尝试修改，查看显示是否正确；查看行号对应是否正确
--	--

### 5.1.2 文件基本操作检查

测试项名称	文件操作	测试项标识	
追踪关系			
测试项描述	<p>包含在菜单操作中，有 保存/打开/另存/关闭 功能。保存需要指定保存的路径，文件名，文件类型；打开需要选择打开哪个文件；另存需要指定保存的路径，文件名，文件类型；关闭需要提示用户是否需要保存。</p> <p>在用户创建一个类项目时，可以自动生成类的头文件和类的函数的.cpp文件，可以进行命名，之后再在源文件中创建一个包含主函数的文件即可。</p> <p>支持用户在头文件中创建.h文件，在源文件中创建.cpp文件。</p>		
软件设计约束	无。		
测试内容	<p>1.打开或新建多个文件；</p> <p>2.打开文件关闭；打开文件修改后关闭</p> <p>3.点击文件菜单下的新建按钮</p> <p>4.目录中右键点击文件夹新建文件，点击新建文件，输入文件名，选择文件类型，点击确定，在该路径下新建文件。</p> <p>5.键点击目录中的任意文件夹弹出自定义的右键菜单，点</p>		

	<p>击新建文件夹，弹出新建文件夹弹窗，输入文件夹名字，点击确定，文件夹被新建在当前目录下，并添加到本地。</p> <p>6.点击建立一个文件夹。</p> <p>7.点击menubar中的文件，弹出下拉菜单，点击打开，弹出文件选择页面，选择文件，点击打开，文件所在路径下的文件展示在侧边目录栏中，并根据文件类型添加图标，文件内容显示在右侧编辑框中。</p> <p>8.点击menubar中的关闭文件。</p> <p>9.右键点击工作路径文件夹关闭项目，点击关闭项目，工作路径消失。</p> <p>10.点击menubar中的文件，点击另存，弹出提示框提示你输入文件名和文件类型，输入文件名和保存类型点击保存，弹出提示框显示保存成功。</p> <p>11.编辑文件内容后点击保存，检查本地磁盘中是否已保存文件。</p> <p>12.右键点击任意一个文件，点击删除文件，删除该文件。</p> <p>13.鼠标光标指向一个文件夹，点击右键，弹出定制化弹窗，点击删除文件夹，文件夹在目录中消失，并在本地删除。</p> <p>14.右键点击文件，点击打开所在文件夹，弹出文件资源管理器，显示该路径下的文件。</p> <p>15.双击目录中的文件，显示在右侧编辑框中。</p> <p>16.点击右上角关闭按钮。</p>
--	---

### 5.1.3 文本基本操作检查

测试项名称	文本操作	测试项标识	
追踪关系			
测试项描述	<p>选中文本段，按下“Ctrl+C”进行复制，按下“Ctrl+V”进行粘贴，按下“Ctrl+X”进行剪切，并在“编辑”菜单中有实体的“复制”、“粘贴”、“剪切”按钮。</p> <p>对任意字符串可以进行插入、修改和删除操作，将光标移到需要修改的地方点击，即可进行增、删、改、查等操作，按下“Ctrl+Z”可以撤销上一步操作。</p> <p>提供全局查找功能，对于选中的字符或字符串，点击“全局查找”可以找出全文中含有多少个该字符串，并高亮所有字符串。</p> <p>提供顺序查找功能，对于选中的字符或字符串，可首先高亮显示全文第一个该字符串，点击“next”按钮可找到并高亮下一个字符串。</p> <p>提供全局替换功能，选中某字符或字符串后，右键并选择“全部替换”，在任意一处字符串处修改；或者点击“编辑”菜单中的“替换”，在输入框修改，即可将全文的原字符串替换为新字符串，并且都标亮。</p>		
软件设计约束	无。		

测试内容	<p>1.点击图标或者menubar中的粘贴。</p> <p>2.选中部分文字，点击图标剪切或者menubar中展开的剪切</p> <p>3.编写一段文本，点击全局查找。</p> <p>4.编写一段文本，点击查找上一个，观察具体查找的过程。主要观察查找的高亮。</p> <p>5.编写一段文本，点击查找下一个，观察具体查找的过程。主要观察查找的高亮。</p> <p>6.编写一段文本，选定所要替换的字符与替换之后的字符点击全局替换，观察替换是否成功。</p> <p>7.编写一段文本，点击查找，如果查找到了，字符会显示黄色的高亮，之后写好你想要的字符，点击替换当前字符。</p>
------	---

#### 5.1.4 编译运行检查

测试项名称	编译操作	测试项标识	
追踪关系			
测试项描述	<p>在选择好特定的编译器后点击“编译”按钮（也可按下快捷键F5），会调用对应的编译器将用户组织编写的文本形式的源代码转化为可执行文件（.exe）。</p> <p>在这个过程中，会同时在编译日志项目栏中输出编译日志，日志中包括调用的编译器、警告数、错误数、编译时</p>		

	<p>间。</p> <p>鼠标点击行号左侧可以添加断点。</p> <p>鼠标点击Debug按钮启动Debug程序。</p> <p>点击程序中的变量可以在控制台中显示Debug过程中变量的值。</p>
软件设计约束	无。
测试内容	<ol style="list-style-type: none"><li>1.编写一段正确的代码，点击运行。</li><li>2.编写一段错误的代码，点击运行。</li><li>3.编写一段正确的代码，或使用已经通过编译的文件，点击debug。</li><li>4.编写一段正确的代码，或使用已经通过编译的文件，点击增加，删除，查看断点按钮。</li><li>5.编写一段正确的代码，或使用已经通过编译的文件，增加断点后运行程序。</li><li>6.编写一段正确的代码，或使用已经通过编译的文件，进行调试并查看变量。</li></ol>

### 5.1.5 代码编写检查

测试项名称	代码编写	测试项标识	
追踪关系			
测试项描述	<p>识别相应关键字，如：int。</p> <p>能自动填充关键字，如输入“in”，光标下方会有提示有</p>		



	<p>哪些，按下“Tab”会自动补足。</p> <p>对不符合格式的关键字，如“include”少写了一个n，下方用红色波浪线标注，光标移至错误处时会有错误提示，如“iclude”:unknown word</p> <p>在函数体、循环体、判断语句内部写代码的时候，所在的外层“{”会有光标停留显示，提示编程人员其所处位置。</p> <p>用户打出一个“{”，程序自动匹配“}”。</p> <p>检查括号的逻辑正确性：如果用户打出“{”但是没有“}”与之匹配，需要标红以提醒用户。</p> <p>在“{”中再次按下“}”键支持光标向右移动一位，退出括号。</p> <p>当光标放在括号位置时，该括号以及与该括号匹配的括号会有光标提示。</p> <p>显示行号。</p> <p>同一个函数，“while”，“for”循环体，以及判断语句的开头部分代码左侧会有折叠按钮</p> <p>点击后，函数以及循环体部分会折叠到函数或循环体第一行后，以“...”形式呈现</p>
软件设计约束	无。
测试内容	<p>1.点击未被折叠的左大括号所在行行号，再点击被折叠的左大括号所在行行号，测试多组不同位置不同嵌套等</p>

	<p>级的大括号。</p> <p>2.输入三个字符，观察是否有提示出现，选择一个提示按下回车，观察是否可以自动补全。</p> <p>3.输入“{”，观察其是否自动补全；在输入回车，观察其是否自动缩进。</p> <p>4.分别测试输入“[”、“{”、“(”，观察在输入框内是否补全了后括号，且光标是否放置于括号中间。</p> <p>5.将光标位置至于括号处，观察该括号以及与其匹配的括号是否出现光标提示。另外，将光标移到其他位置或者字符串发生变化时，该提示是否消失。</p> <p>6.编写代码后，增加几行，减少几行代码，查看行号的变化是否正确。</p> <p>7.点击代码折叠函数，或while/for循环体，检查折叠效果。</p>
--	--

#### 5.1.6 关键字高亮检查

测试项名称	关键字高亮	测试项标识	
追踪关系			
测试项描述	识别相应关键字，如：int，根据识别到的关键字，并根据配色方案进行相应的上色。		
软件设计约束	无。		
测试内容	1.输入对应的字符，观察是否有高亮显示，是否正确。		

	<p>2.输入特殊符号字符，观察是否有高亮显，是否正确。</p> <p>3.输入头文件字符，观察是否有高亮显示，是否正确。</p> <p>4.输入数字字符，观察是否有高亮显示，是否正确。</p> <p>5.输入单行和多行注释，观察是否有高亮显示，是否正确。</p> <p>6.输入引号及字符串，观察是否有高亮显示，是否正确。</p> <p>7.输入函数，观察是否有高亮显示，是否正确。</p> <p>8.输入非关键字，观察是否有高亮显示。</p>
--	---

#### 5.1.7 鼠标键盘操作检查

测试项名称	鼠标键盘操作	测试项标识	
追踪关系			
测试项描述	<p>可以右击鼠标打开菜单</p> <p>滚轮与ctrl配合可以缩小放大屏幕</p>		
软件设计约束	无。		
测试内容	<p>1.进入程序，鼠标右键单击菜单。</p> <p>2.先是不按住Ctrl单独滑动鼠标滚轮上下，然后按住Ctrl后单独滑动鼠标滚轮上下，测试多组不同位置不同幅度的鼠标滚轮滑动。检查放大和缩小。</p>		

#### 5.1.8 切换主题操作检查

测试项名称	主题切换	测试项标识	
-------	------	-------	--

追踪关系	
测试项描述	根据需求进行主题切换，包括颜色和图标的切换。
软件设计约束	无。
测试内容	1.在菜单栏中点击“更换主题”，选择不同风格，观察主题颜色与图标的改变。

## 5.2 性能效率测试

测试项名称	性能效率测试	测试项标识	XN
追踪关系			
测试项描述	支持单文件编辑，文本长度不超过2000行。 装入/保存2000行源程序时间小于3秒。 基本操作（插入/删除/翻页/查找等）响应无明显延迟。 全文自动替换响应时间小于3秒。		
软件设计约束	无。		
测试内容	针对每一种性能需求分别进行测试，对时间进行检查。		

## 5.3 兼容性测试

测试项名称	兼容性测试	测试项标识	JR
追踪关系			
测试项描述	可以在windows, mac, linux系统上运行		

	支持运行用户的.c文件
软件设计约束	无。
测试内容	1. 在windows, mac, linux系统上分别运行软件, 检查功能是否可以正常执行 2. 运行用户的.c文件, 查看是否有运行结果, 结果是否正确

## 5.4 易用性测试

测试项名称	易用性测试	测试项标识	YY
追踪关系			
测试项描述	直观, 人机交互效果好的界面。 操作便捷, 功能完善。		
软件设计约束	无。		
测试内容	1. 用户界面的友好性: 界面的简洁性如何, 是否为系统管理员提供简洁、直观、友好的图形化管理界面, 方便操作。 2. 易操作性: 操作的难易程度, 对主要或常用功能应该提供快捷方式。		

## 5.5 可靠性测试

测试项名称	可靠性测试	测试项标识	KK
追踪关系			
测试项描述	用户的文件数据要求必须完整读入编译 软件在运行期间不会崩溃 对于软件异常进行基本处理		



软件设计约束	无。
测试内容	1.数据完整性：数据完整性审查，编译用户的文件检查编译结果是否为完整文件。 2.测试不同状态下软件是否会崩溃。 3.抛出异常，检查软件是否可以自动处理异常

## 5.6 信息安全性测试

测试项名称	信息安全性测试	测试项标识	XA
追踪关系			
测试项描述			
软件设计约束	无。		
测试内容	信息安全性陈述：检查产品说明是否包括有关信息安全性的陈述，包括保密性、完整性、抗抵赖性、可核查性、真实性以及信息安全性的依从性，并经书面形式展示可验证的依从性证据。		

## 5.7 维护性测试

测试项名称	维护性测试	测试项标识	WH
追踪关系			
测试项描述	软件模块和功能具有可重用性、易分析性、易改变性、易测试性，并且在其他软件中复用性强。		
软件设计约束	无。		

测试内容	<p>维护性陈述：检查产品说明是否包含有关维护的陈述，包括模块化、可重用性、易分析性、易改变性、易测试性以及维护性的依从性，并经书面形式展示可验证的依从性证据。</p> <p>维护信息：检查产品说明是否有用户所需的维护信息。</p> <p>用户修改：当软件能由用户作适应性修改时，检查产品说明是否标识用于修改的工具或规程及其使用条件。</p>
------	---

## 5.8 可移植性测试

测试项名称	可移植性测试	测试项标识	YZ
追踪关系			
测试项描述	软件可以安装和移植。		
软件设计约束	无。		
测试内容	<p>可移植性陈述：检查软件产品说明是否包含有关可移植性的陈述，包括适应性、易安装性、已替换性以及可移植的依从性，并经书面形式展示可验证的依从性证据。</p> <p>安装规程：检查产品说明中是否提供安装规程信息。</p>		







## 5.9 用户文档集测试

测试项名称	用户文档集测试	测试项标识	WD
追踪关系			
测试项描述	提供用户文档，软件使用说明书等相关文档		
软件设计约束	无。		
测试内容	<p>规范性：用户文档描述规范，有版本控制修改记录。</p> <p>符合性：用户文档与需求和设计文档的符合程度。</p> <p>完整性：用户手册内容基本完整，对具体操作的说明比较详细。</p> <p>一致性：用户手册的描述与软件的实际功能基本一致，对重要功能的说明比较全面，用户手册中具有产品版本号描述。</p> <p>易理解程度：用户手册对操作有图例和文字说明，较易理解。</p>		

## 6 项目管理

### 6.1 测试审核过程

对项目的管理的时候，我们小组采取了测试用例文档审查这一手段，通过审查，退回哪些不合格的测试用例文档，通过哪些合格的测试用例文档，最终我们组的测试审查的审查文档达到 25 个之多。

 测试用例审查文档20.docx	2022-12-14 21:20
 测试用例审查文档21.docx	2022-12-14 21:20
 测试用例审查文档22.docx	2022-12-14 21:20
 测试用例审查文档23.docx	2022-12-14 21:20
 测试用例审查文档24.docx	2022-12-14 21:20
 测试用例审查文档25.docx	2022-12-14 21:20

## 6.2 测试人力资源管理

测试人员	每人每天可以投入 工时	每人每天可以产 生测试用例数	每人每天可以产 生文档数
刘睿泽	3h	3	7
陈晓璐	3h	5	7
吕泽超	2.7h	5	5
梁坤	3h	4	6
张桢	2.5h	6	6
杨宜松	2h	6	6

## 6.3 工作分工与进度安排

进行工作分解，并制定进度安排，见表 8。

表 8 工作进度安排

工作任务	负责人	工作量	完成日期
确定测试目标项目	刘睿泽	2h	2022/11/7

初步完成测试计划文档	杨宜松，吕泽超，梁坤	4h	2022/11/10
需求分析审查	梁坤	2h	2022/11/20
设计审查	梁坤	2h	2022/11/20
单元测试	刘睿泽，陈晓璐，吕泽超，梁坤，杨宜松，张桢	18h	2022/12/10
集成测试	刘睿泽，陈晓璐，吕泽超，梁坤，杨宜松，张桢	6h	2022/12/11
系统测试	吕泽超	2h	2022/12/12
回归测试	杨宜松	2h	2022/12/13
验收测试	刘睿泽，陈晓璐，吕泽超，梁坤，杨宜松，张桢	18h	2022/12/14
测试总结	刘睿泽	2h	2022/12/14

## 6.4 使用 GitHub 进行项目文档管理

使用 GitHub+git 管理测试过程产生的文档，避免使用微信传输，容易造成文档过期丢失。也方便整理文档的结构

Git 的使用过程如下：

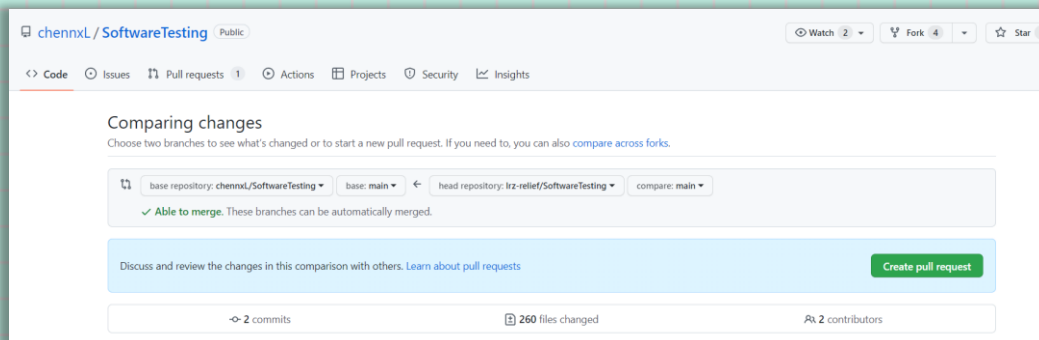


```
lrz@LAPTOP-76FA1BJ5 MINGW64 /d/codeWareHouse/SoftwareTesting (main)
$ git commit -m "add"
[main b66febb] add
260 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 使用权声明/使用权声明.docx
create mode 100644 加分申请/1.txt
create mode 100644 团队总结/1.txt
create mode 100644 文档封皮/1.txt
rename {手工测试用例文档 => 测试方案/手工测试用例文档}/001-F-PageTest
用例文档.docx (100%)
rename {手工测试用例文档 => 测试方案/手工测试用例文档}/002-F-PageTest
```

```
lrz@LAPTOP-76FA1BJ5 MINGW64 /d/codeWareHouse/SoftwareTesting (main)
$ git pull
Enter passphrase for key '/c/Users/39752/.ssh/id_rsa':
Already up to date.

lrz@LAPTOP-76FA1BJ5 MINGW64 /d/codeWareHouse/SoftwareTesting (main)
$ git push
Enter passphrase for key '/c/Users/39752/.ssh/id_rsa':
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 29.74 KiB | 2.70 MiB/s, done.
Total 9 (delta 0), reused 2 (delta 0), pack-reused 0
To github.com:lrz-relief/SoftwareTesting.git
708fd8b..b66febb main -> main
```

Github 的 pull request 过程如下:



## 6.5 使用华为云 ProjectMan 进行测试用例管理

老师建议使用基于数据库的测试用例管理工具，我们选择了华为云 Projectman，我们将我们的所有测试用例都上传上去，在云端保存，形成了本地+云端的测试用例的存储体系，也实现了利用基于数据库的测试用例管理工具

管理测试用例。

项目测试计划如下：



测试用例的列表如下，这里仅仅放出了单元测试、集成测试、系统测试与回归测试

验收测试因为其全部都是结果正确的测试，所以没有必要展示。

所有用例

请输入关键字

高级过滤

导入

新建

<input type="checkbox"/> 名称	编号	结果	级别	执行者	操作
<input type="checkbox"/> 215-F-KeyWordTest-A-D-B	215-F-KeyWor...	成功	L0	lrz-relief	
<input type="checkbox"/> 214-F-AutoParenthesesMatchTest-A-D-B	214-F-AutoPar...	失败	L0	lrz-relief	
<input type="checkbox"/> 213-F-AutoParenthesesMatchTest-A-D-B	213-F-AutoPar...	失败	L0	lrz-relief	
<input type="checkbox"/> 212-F-AutoIndentationTest-A-D-B	212-F-AutoInd...	失败	L0	lrz-relief	
<input type="checkbox"/> 211-F-KeyWordTest-A-D-B	211-F-KeyWor...	失败	L0	lrz-relief	
<input type="checkbox"/> 210-F-StringLookupTest-A-D-B	210-F-StringLo...	成功	L0	lrz-relief	

具体的一个测试用例在华为云 projectman 上的截图：

001-F-PageTest-U-D-B

保存

×

详情

结果记录

缺陷列表

操作历史

描述

运行整个程序，直接观察界面的最上面是否存在菜单栏与工具栏。以及菜单栏与工具栏是否符合测试需求中提到的习惯性原则。

前置条件

程序可以正确运行且已经可以显示界面

测试步骤

步骤	步骤描述	预期结果	操作
1	(1) 运行程序 (2) 观察界面	菜单栏与工具栏的位置是界面的最上面而且对齐方式为左对齐	

基本信息

状态: 完成

编号: 001-F-PageT...

用例等级: L2

执行方式: 手工测试

模块: --

处理者: --

版本号: 002

迭代: --

标签:

类型: 功能性测试

关联需求: 关联需求

附件: 添加附件

用例通过率与用例完成率图



6.6 测试人员激励办法

6.6.1 理论方法

- (1) 表扬和奖励
- (2) 站在测试小组一边
- (3) 提高士气
- (4) 支持合理的工作方式

### 6.6.2 实践方法

针对小组内的 6 个成员，组长采用了不同的方法激励队员，其中包括：

- (1) 以加分的方式来激励队员，每设计出 1 个测试用例可以加 50 分。



最后可以通过算总分的方式来作为小组成员打分的一个标准。

- (2) 以抢红包的方式激励同学们努力投入测试工作，这个方法借鉴自单纯老师的抢答



总体来说组长尝试各种方法去激励自己的组员，让自己的组员做到效率最大。

6.7 测试团队的知识共享与培训

为了节省组员的时间也为了给组员传授知识，我通过录视频的方式代替开会，对组员进行培训与指导我自己录制了四个视频，对组员进行培训与指导。



6.8 跟踪与控制

在测试过程中测试监督人员对测试按如下方式进行监督：

表 9     测试监督内容与方式

阶段	跟踪内容	跟踪方式	跟踪频度
测试需求及测试策划	测试项划分情况	例会	一周两次
测试设计、文档审查及代码审查	用例设计的情况和进度；文档审查、代码审查的情况和进度；环境构建情况和进度；存在的问题；测试人员	例会	一周两次
测试执行	每日已测用例数；每日发现问题数统计；用例调整、环境问题。	总结会	两周一次

测试总结	信息的完整性。	例会	一周两次
------	---------	----	------

过程跟踪表参照《软件测评过程监督程序》中的跟踪表，其中的具体工作内容可根据具体情况细化。

在生成用例中，监督人员负责审核相应的用例的合理性和充分性，单元测试具体分工参见表 10。

表 10 测试分工与需求追踪的关系

记录项目中的测试跟踪报告（不是全部）：

### 测试用例跟踪文档

#### 22/12/01 测试用例跟踪

人员	预测设计的测试用例数	通过测试的测试用例数	未通过测试的测试用例数	未测试的测试用例数
刘睿泽	5	3	2	0
陈晓璐	5	4	1	0
吕泽超	0	0	0	0
梁坤	0	0	0	0
张桢	5	1	4	0
杨宜松	15	7	8	0

未测试的原因：

无未测试用例



**未设计测试用例原因：**

梁坤：为小组初步定义测试用例文档模板，没有时间写测试用例

**22/12/02 测试用例跟踪**

人员	预测设计的测试用例数	通过测试的测试用例数	未通过测试的测试用例数	未测试的测试用例数
刘睿泽	0	0	0	0
陈晓璐	0	0	0	0
吕泽超	0	0	0	0
梁坤	0	0	0	0
张桢	5	4	1	0
杨宜松	0	0	0	0

**未测试的原因：**

无未测试用例

**未设计测试用例原因：**

刘睿泽：个人为小组继续补充测试计划。没有时间设计测试用例。

陈晓璐：个人为小组搭建仓库，补充会议记录，没有时间写测试用例

梁坤：为小组初步定义测试需求文档模板，没有时间写测试用例



## 22/12/03 测试用例跟踪

人员	预测设计的测试用例数	通过测试的测试用例数	未通过测试的测试用例数	未测试的测试用例数
刘睿泽	0	0	0	0
陈晓璐	8	3	5	0
吕泽超	5	5	0	0
梁坤	6	2	4	0
张桢	5	1	4	0
杨宜松	0	0	0	0

### 未测试的原因：

无未测试用例

### 未设计测试用例原因：

刘睿泽：个人为小组继续补充测试计划，补充了测试管理过程。没有时间设计测试用例。

## 7 测试风险管理计划

### 7.1 项目风险检查表

类别	内容	示例
----	----	----

人员风险	测试人员状态、责任感与行为规范	个人因为工作疏忽而漏掉缺陷;某人因为生病而不能进行测试,造成资源不足,使测试不充分、或缩小测试范围
环境风险	多数情况下,测试环境史一个模拟环境,它很难与真实环境一致	测试环境不能模拟实际环境,运行环境存在垃圾数据。
测试范围风险	很难完成 100%的测试覆盖率,有些边界范围容易被忽视	很难覆盖模块之间接口参数传递、成千上万的操作组合等测试。
测试深度风险	对系统容量、可靠性等测试深度不够	对于应用的操作行为与习惯等研究不够,测试时达不到实际的用户数。
回归测试风险	一般都不会运行所有的测试用例,而是运行部分测试用例,越到测试后期,回归测试执行的用例数就越少	修正一个缺陷之后,处理验证这个缺陷之外,测试人员往往根据自己的经验来确定回归测试的范围,而且这个范围很小。
外界风险	疫情等外界因素	同学因为疫情返乡,不能投入工作;同学可能因为疫情封控等原因工作效

		率下降。
--	--	------

## 7.2 定义风险等级

严重：导致测试项目无法开展的风险或者开展之后无法进行；如：测试计划不明确、测试方案不确定、关键技术人员缺失（可能因为疫情原因）、新的测试工具的学习时间比预期长、决策层决策错误或者不能明确给出意见。

中等：测试项目可以开展，但由于以下因素导致开发进度缓慢；如：测试规划不清、方案不佳、测试项目关键里程碑计划缺失或延期、测试标准缺失、项目管理计划书缺失、测试人员技术水平不足

微小：项目开展正常，但由于存在以下因素将导致项目开发质量无法达到预期要求；如：测试输入输出规范缺失、不能达到预期目标。

## 7.3 风险应对方法

- (1) 采取措施避免可以避免的风险
- (2) 高风险转移为低风险
- (3) 设法降低不可避免的风险
- (4) 做好风险管理计划
- (5) 制定处理风险的应急、有效方案
- (6) 计划时，对于估算资源、时间、预算留有余地
- (7) 制定文档标准，建立机制，保证文档及时产生

## 8 缺陷管理

### 8.1 定义缺陷优先级

立即解决 (P1 级): 缺陷导致系统几乎不能使用或测试不能继续, 需立即修复

高优先级 (P2 级): 缺陷严重, 影响测试, 需要优先考虑

正常排队 (P3 级): 缺陷需要正常排队等待修复

低优先级 (P4 级): 缺陷可以在开发人员有时间的时候被纠正。

### 8.2 缺陷报告表

分类	项目	描述可跟踪信息
可跟踪信息	缺陷 ID	唯一的、自动产生的缺陷 ID, 用于识别、跟踪、查询
软件缺陷基本信息	缺陷状态	可分为“打开或激活的”、“已修正”、“关闭”等
	缺陷标题	描述缺陷的最主要信息
	缺陷的严重程度	一般分为“致命”、“严重”、“一般”、“小”等四种程度
	缺陷的优先级	描述处理缺陷的紧急程度, 1 是优先级最高的



		等级，2 是正常的，3 是优先级最低的
	缺陷的产生频率	描述缺陷发生的可能性 1%-100%
	缺陷提交人	缺陷提交人的名字（会和邮件地址联系起来）， 般就是发现缺陷的测试人员或其他人员
	缺陷提交时间	缺陷提交的时间