

SysY2022E 扩展语言编辑器

SysY2022E Language Support

需求规格说明书

成员信息: 李晓坤 信息安全 信安 211 U202141863

王若凡 信息安全 信安 211 U202141852

王宠爱 信息安全 信安 211 U202141853

成豪 信息安全 信安 211 U202141880

指导教师: 崔 晓 龙

目录

1. 引言	3
1.1. 编写目的	3
1.2. 背景	3
1.3. 术语定义	4
1.4. 参考资料	4
1.5. 版本更新信息	5
2. 任务概述	6
2.1. 系统定义	6
2.1.1. 项目来源及背景	6
2.1.2. 达到目标	6
2.1.3. 系统整体架构	6
2.1.4. 各组成部分结构	7
2.2. 应用环境	7
2.2.1. 设备环境	7
2.2.2. 系统运行硬件环境	7
2.2.3. 系统运行软件环境	7
2.2.4. 系统运行网络环境	8
2.2.5. 用户操作模式	8
2.2.6. 当前应用环境	8
2.3. 假定和约束	9
2.3.1. 开发假定与约束	9
2.3.2. 最终用户特点	9
2.3.3. 人员限制	9
2.3.4. 预期使用频度	10
3. 需求规定	10
3.1. 对功能的规定	10
3.1.1. IDE 集成	10
3.1.2. 语法高亮和悬浮提示	10
3.1.3. 语法检查与错误提示	12
3.1.4. 静态语义检查	13

3.1.5. 修复建议	14
3.1.6. 程序错误检查	15
3.1.7. 代码重构	16
3.2. 对性能的规定	17
3.2.1. 精度	17
3.2.2. 时间特性要求	18
3.2.3. 灵活性	19
3.3. 输入/输出要求	20
3.3.1. 输入数据类型	20
3.3.2. 输出数据类型	20
3.3.3. 硬拷贝报告	21
3.3.4. 图形或显示报告	21
3.3.5. 控制输出量说明	21
3.4. 故障处理要求	22
3.4.1. 软件故障	22
3.4.2. 硬件故障	23
3.5. 其他要求	23
3.5.1. 易读性	23
3.5.2. 可靠性	24
3.5.3. 使用方便性	24
3.5.4. 可维护性	24
3.5.5. 可补充性	24
3.5.6. 安全保密	24
4. 运行环境规定	25
4.1. 设备	25
4.1.1. 处理器及内存	25
4.1.2. 外存	25
4.1.3. 输入及输出设备	25
4.1.4. 数据通信设备	25
4.2. 支持软件	25
5. 双方签字	27

1. 引言

1.1. 编写目的

编写需求文档是为了明确和定义开发一个 SysY2022E 扩展语言编辑器所需的功能和特性，以便开发团队、产品经理、设计师和其他利益相关者之间达成一致，并作为开发过程中的参考框架。具体目的包括：

确定产品范围：需求文档帮助在项目启动阶段明确产品的范围和，指导团队在开发过程专注于满足用户需求，规避功能膨胀和过度设计。

明确用户需求：需求文档描述用户的需求和期望，有助团队了解用户需求，确保产品设计符合用户期望。

确保功能一致性：需求文档定义了每个部分的详细规格交互方式，确保团队对功能的实现方式保持一致，避免功能实现的差异性。

促进沟通和协作需求：文档作为团队成员之间沟通和协作的桥梁，帮助团队成员共同理解项目目标和需求，减少沟通失误和偏差。

评估项目进度：需求文档可以作为项目进度和质量的参考标准，帮助团队了解项目进展情况，及时调整和优化开发计划。

1.2. 背景

1) 名称：SysY2022E 扩展语言编辑器。

2) 介绍：SysY2022E 是一种 C-like 的高级编程语言，其扩展语言编辑器旨在为用户提供更加强大、灵活的编程功能，同时提升用户的开发体验。

3) 本项目的任务提出者、开发者、用户及实现该产品的单位：

- ◆ 任务提出者：北京科技大学计算机与通信工程学院；
- ◆ 开发者：李晓坤、王若凡、王宠爱、成豪；
- ◆ 用户：北京科技大学全体师生；
- ◆ 实现该产品的单位：北京科技大学计算机与通信工程学院。

4) 开发完成的 SysY2022E 扩展语言编辑器作为 vscode 的插件，将与 vscode 紧密集成，利用 vscode 提供的平台与接口，实现 SysY2022E 代码的编辑、调试和其他 IDE 功能，同时 vscode 作为宿主环境，为该编辑器插件提供必要的支持和展示界面，共同为开发者打造高效的 SysY2022E 开发环境。

1.3. 术语定义

表 1 术语定义表

术语	定义
SysY2022E 语言	SysY2022 是 C 语言的一个子集，最初是为“全国大学生系统能力大赛-编译器大赛”设计的迷你编程语言。SysY2022E 语言是在其基础上拓展的语言。
LSP	微软指定的 Language Server Protocol，它标准化了语言工具和代码编辑器之间的通信。
ANTLR4	ANTLR4 是一款功能强大的解析器生成器，用于读取、处理、执行和翻译结构化文本或二进制文件。

1.4. 参考资料

- [1] 卢慧琼, 李百龄, 张溶溶. C 语言的结构编辑器[J]. 计算机研究与发展, 1986, (07):22-31.
- [2] 宋国钦. 编辑器设计技术[J]. 现代计算机, 1996, (05):21-23.
- [3] 微软 c++ language support 文档:
<https://learn.microsoft.com/lt-lt/cpp/overview/visual-cpp-language-conformance?view=msvc-170>.
- [4] VScode 插件开发文档:
<https://rackar.github.io/vscode-ext-doccn/get-started/your-first-extension.html%E5%BC%80%E5%8F%91%E6%8F%92%E4%BB%B6>.
- [5] LSP 官方文档:
<https://vscode.github.net.cn/api/language-extensions/language-server-extension-guide#google-vignette>.
- [6] LSP 官方示例项目:
<https://github.com/microsoft/vscode-extension-samples/tree/main/lsp-sample>.
- [7] ANTLR4 官方文档:

<https://pragprog.com/titles/tpantlr2/the-definitive-antlr-4-reference/>.

[8] ANTLR4 官方 github 项目: <https://github.com/antlr/antlr4>.

[9] 编译原理教材: <https://book.douban.com/subject/26736235/>.

[10] SysY2022 编译器大赛项目参考: https://code.educoder.net/ph7n2ofui/SysyCompiler_Arm.

[11] Clangd 官方文档: <https://clangd.llvm.org/>.

[12] Clangd 官方项目: <https://github.com/clangd/vscode-clangd>.

1.5. 版本更新信息

在项目开发过程中, 使用 github 进行项目版本管理, 共迭代了 5 个版本。在每一次版本迭代时, 关于具体的修改时间、修改位置、修改内容均在 github 仓库中有详细记录。由于记录内容繁杂, 不便于在报告中展示, 因此在文档中省略关于版本更新信息的说明, 如若想要查看相关版本更新信息, 请查看本项目的 github 仓库。

表 2 版本更新表

修改编号	修改日期	修改后版本	修改位置	修改内容概述
U005	2024. 06. 08	v5. 0	见仓库	见仓库

2. 任务概述

2.1. 系统定义

2.1.1. 项目来源及背景

项目来源：SysY2022E 扩展语言编辑器的项目来源于对现有 SysY2022 编程语言编辑器的改进和扩展需求的提出，旨在提供更强大、更灵活的代码编辑和开发环境。

项目背景：SysY2022 是 C 语言的一个子集，最初是为“全国大学生系统能力大赛-编译器大赛”设计的迷你编程语言。虽然 SysY 是一种简化的 C 语言，但经过简单扩展后能够编写相对复杂的程序。本项目将以扩展的 SysY2022 语言为目标（称为 SysY2022E），开发一款支持 SysY2022E 程序编写的编辑器环境，从而为 SysY2022E 开发人员提供能更好的开发体验。

2.1.2. 达到目标

1) 市场目标：提供一款功能强大、易用的扩展语言编辑器，满足程序员和开发者的编程需求，拓展用户群体，提升市场竞争力。

2) 技术目标：

- ◆ 设计一个高效、稳定的系统架构，支持多平台（Windows、Linux、Mac）运行；
- ◆ 提供丰富的 API 接口，方便第三方插件和工具的集成；
- ◆ 实现高效的代码解析、编译和调试功能，提高开发效率；
- ◆ 提供友好的用户界面和交互体验，降低学习成本；
- ◆ 项目开源，提供完整的开发手册，方便后续开发和改进。

2.1.3. 系统整体架构

1) 系统提供的主要功能包括：

- ◆ 代码编辑：提供语法高亮、悬浮提示、语法检查、错误提示、自动补全；
- ◆ 项目管理：支持多项目、多文件的管理和切换；
- ◆ 编译和执行：将自定义语言代码编译成可执行文件或脚本，并直接运行；当检测到代码中的错误时提供修复建议；
- ◆ 调试工具：提供断点设置、变量查看、执行流程控制。

2) 涉及的接口包括：

- ◆ API 接口：提供 API 接口供第三方插件调用，扩展编辑器功能；

- ◆ 文件接口：支持多种文件格式（如.yy2022）的导入和导出；
- ◆ 网络接口：支持远程代码存储和版本控制功能。

2.1.4. 各组成部分结构

SysY2022 扩展语言编辑器作为一个独立的软件产品，也可以作为更大系统（如集成开发环境 IDE）的一个组成部分。

在本次项目开发中，开发人员严格遵守软件工程开发步骤，关于各组成部分的结构部分见系统设计的相关文档资料。

2.2. 应用环境

2.2.1. 设备环境

SysY2022 扩展语言编辑器主要设计用于标准的个人计算机（PC）设备，包括但不限于台式机、笔记本电脑和一体机等。这些设备应配备有适当的输入设备（如键盘、鼠标或触摸板）和输出设备（如显示器、扬声器等），以支持编辑器的正常输入和输出操作。

2.2.2. 系统运行硬件环境

表 3 硬件环境规定表

硬件	规定
处理器	包括但不限于 Intel Core i 系列、AMD Ryzen 系列等。最低要求为双核处理器，但建议至少使用四核处理器以获得更好的性能。
内存	建议至少拥有 8GB 的 RAM。对于大型项目或同时运行多个实例的情况，建议配置更高的内存容量。
存储	推荐使用 SSD 作为系统盘。支持从本地硬盘、固态硬盘（SSD）或网络存储设备（如 NAS）加载和保存项目。
显示	支持至少 1024x768 分辨率的显示器。为了获得更好的用户体验，建议使用高分辨率的显示器，并支持全高清（1080p）或更高分辨率。

2.2.3. 系统运行软件环境

表 4 软件环境规定表

软件	规定
操作系统	包括但不限于 Windows 10/11、macOS 以及 Linux 发行版（如 Ubuntu、Debian 等）。
插件	安装相关插件以支持项目运行，包括 antlr4、C/C++、GitHub Copilot、Java、Docker、JavaScript。
其他	可能还需要安装其他软件以支持特定的功能或插件，如版本控制系统(Git)、node.js。

2.2.4. 系统运行网络环境

表 5 网络环境规定表

网络	规定
网络连接	支持通过有线或无线方式连接到互联网，以使用户能够访问在线资源、远程存储库或与其他用户协作。
网络协议	支持标准的网络协议，如 TCP/IP、HTTP/HTTPS、FTP 等，以便与远程服务器或云服务进行通信。
网络安全性	支持安全的网络连接方式，如 VPN、SSH 隧道等，以保护用户数据和通信安全。

2.2.5. 用户操作模式

表 6 用户操作模式规定表

操作模式	规定
图形用户界面 (GUI)	提供直观的图形界面，使用户能够轻松地编辑、编译、调试和管理 SysY2022E 项目。
命令行界面 (CLI)	对于高级用户或需要自动化脚本的场景，提供命令行接口以支持脚本编写和自动化任务。
远程操作	支持通过远程桌面、SSH 等协议进行远程编辑和调试操作。

2.2.6. 当前应用环境

当前 SysY2022E 扩展语言编辑器的应用环境主要针对专业开发人员、教育者以及对 SysY2022E 语言感兴趣的个人用户。这些用户通常需要在各种设备和操作系统上安装和使用编辑器，并且需要能够灵活地配置和管理他们的项目。因此，SysY2022 编辑器应具备跨平台兼容性、易用性和可扩展性等特点，以满足不同用户的需求。

2.3. 假定和约束

2.3.1. 开发假定与约束

1) 经费限制:

- ◆ 假定项目经费为有限资源, 需在整个开发周期内合理分配;
- ◆ 约束条件包括人力成本、设备采购、软件测试与市场推广等费用。

2) 开发期限:

- ◆ 假定项目需要在软件工程课程设计这个特定期限内完成, 以满足市场需求;
- ◆ 约束条件包括开发阶段的时间分配、关键里程碑的达成时间以及最终交付时间。

3) 技术选型:

- ◆ 假定采用成熟的软件开发框架和技术栈, 以确保产品的稳定性和可维护性;
- ◆ 约束条件包括技术栈的选择、第三方库的兼容性以及技术更新的频率。

4) 硬件兼容性:

- ◆ 假定产品需要支持多种操作系统和硬件配置, 以满足不同用户的需求;
- ◆ 约束条件包括不同操作系统和硬件的兼容性测试、性能优化以及适配性调整。

5) 法律与标准:

- ◆ 假定产品需遵守相关法律法规和行业标准, 如数据保护、软件许可等;
- ◆ 约束条件包括法律条款的遵守、标准规范的遵循以及合规性检查。

2.3.2. 最终用户特点

1) 技术背景: 最终用户可能具备不同程度的编程和软件开发经验, 从初学者到专业开发者均有可能。

2) 使用需求: 用户需要使用 SysY2022E 扩展语言进行编程, 并期望编辑器提供高效、便捷的代码编辑和调试功能。

3) 学习意愿: 用户可能愿意投入一定的时间和精力学习使用本产品, 但希望学习过程简单直观。

2.3.3. 人员限制

1) 操作人员:

- ◆ 教育水平: 操作人员应具备基本的计算机使用能力, 对编程和软件开发有一定了解;

- ◆ 技术专长：熟悉 SysY2022E 扩展语言的语法和特性，掌握基本的代码编辑和调试技巧。

2) 维护人员：

- ◆ 教育水平：维护人员应具备计算机科学或相关专业的高等教育背景；
- ◆ 技术专长：熟练掌握软件开发和测试技术，具备深厚的编程能力和问题解决能力。同时，应熟悉 SysY2022E 扩展语言编辑器的内部结构和功能，以便进行故障排除、性能优化和功能更新等工作。

2.3.4. 预期使用频度

假定 SysY2022E 扩展语言编辑器将成为用户的日常开发工具之一，因此预期使用频度将较高。

根据用户反馈和市场调研，预计每天平均使用时长在数小时至数十小时不等，具体取决于用户的编程需求和项目规模。

3. 需求规定

3.1. 对功能的规定

3.1.1. IDE 集成

- ◆ 功能编号：F001
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：高
- ◆ 功能定义：编辑器与主流 IDE 的集成。
- ◆ 功能描述：SysY2022 扩展语言编辑器必须能够与一款主流 IDE（如 Visual Studio Code、Eclipse 等）集成，允许用户通过该 IDE 直接编辑、运行和调试 SysY2022E 程序。集成后，编辑器应提供与 IDE 一致的界面风格和用户体验，同时保留 SysY2022E 语言的特定功能。

3.1.2. 语法高亮和悬浮提示

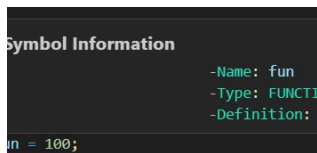
- ◆ 功能编号：F002
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：高
- ◆ 功能定义：对 SysY2022E 程序进行语法高亮和悬浮提示。

- 功能描述：SysY2022 扩展语言编辑器必须能够实时对 SysY2022E 程序进行语法高亮，使得代码更易于阅读和理解。同时，当鼠标悬停在标识符（如变量名、函数名等）上时，编辑器应能够显示相应的信息或提示，这些信息包括但不限于该标识符代表的对象和类型。

表 7 语法高亮规定表

代码类型	颜色	
关键字	#C586C0	
定界符	#DCDCAA	
操作符	#569CD6	
整型常量	#B5CEA8	
浮点数常量	#D4D4D4	
标识符	#9CDCFE	
双引号字符串	#CE9178	
行内注释	#608B4E	
块注释	#608B4E	
结构体名称	#25dfb6	

表 8 悬浮提示规定表

	提示内容	提示方式	示例
变量名	<ul style="list-style-type: none"> 变量名 变量类型 变量的简短描述 	<p>a) 使用一个小的悬浮框显示上述信息，该悬浮框应位于鼠标指针附近，且不影响用户的视线。</p> <p>b) 变量名和类型应以粗体或加长的字</p>	 <pre> Symbol Information -Name: fun -Type: FUNCTION -Definition: fun = 100; </pre>

	<ul style="list-style-type: none"> 变量初始化的值 	<p>体显示，以便快速识别。</p> <p>c) 如果变量有注释或文档字符串，可以显示其简短描述。</p>	
函数名	<ul style="list-style-type: none"> 函数名 函数的返回类型 函数的参数列表（包括参数名和类型） 函数的简短描述或文档字符串 	<p>a) 使用一个较大的悬浮框或悬停预览窗口显示上述信息，因为函数的信息通常较多。</p> <p>b) 函数名和返回类型应以粗体或加长的字体显示。</p> <p>c) 如果有参数，可以列出所有参数并注明类型，参数之间用逗号分隔。</p> <p>d) 如果函数有文档字符串，可以显示其简短描述。</p>	
类名、结构体名	<ul style="list-style-type: none"> 类名或结构体名 继承的类或实现的接口（如果适用） 类的成员列表 类的简短描述或文档字符串 	<p>a) 使用一个悬停预览窗口来展示类的详细信息，因为类的内容通常比较丰富。</p> <p>b) 类名应以粗体或加长的字体显示。</p> <p>c) 列出类的所有成员，并注明成员的类型（属性）或返回类型（方法）。</p> <p>d) 如果有文档字符串，可以显示其简短描述。</p>	

3.1.3. 语法检查与错误提示

- 功能编号：F003
- 所属产品编号：SysY2022E Language Support
- 优先级：高
- 功能定义：对 SysY2022E 程序进行语法检查并提示错误。
- 功能描述：
 - 语法检查：编辑器内置的语法检查器会实时分析代码，检测语法错误。这包括但不限于变量声明、函数定义、表达式书写方面的错误。语法检查器能够识别出不符合 SysY2022E 语言规范的代码片段，并及时给出反馈。
 - 错误提示：当检测到语法错误时，编辑器会在代码中相应的位置标出错误，并提供错误提示信息。这些提示信息旨在帮助开发者理解错误的性质，包括错误类型、可能的原因以及建议的解决方案。错误提示既可以是文本形式的，也可以通过悬浮提示、状

态栏消息等方式展现。

- c) 代码高亮：为了更好地辅助错误识别和代码理解，编辑器通常会提供代码高亮功能。通过不同的颜色或样式区分代码的不同部分（如关键字、变量、函数等），可以帮助开发者快速定位语法错误和逻辑错误。
- d) 自动补全：虽然严格来说不属于语法检查和错误提示的范畴，但自动补全功能可以在编写代码时提供语法结构和代码片段的建议，从而减少语法错误的发生。
- e) 代码格式化：编辑器可能还包括代码格式化工具，它可以自动调整代码布局和风格，以符合 SysY2022E 的编码规范，进一步减少因格式问题导致的错误。
- f) 集成开发环境（IDE）支持：对于 SysY2022E 扩展语言，可能还有专门的 IDE 或插件支持，这些工具提供了更为丰富的功能，包括但不限于上述提到的功能，以帮助开发者更有效地编写、调试 SysY2022E 程序。

3.1.4. 静态语义检查

- ◆ 功能编号：F004
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：中
- ◆ 功能定义：SysY2022 扩展语言编辑器应提供静态语义检查功能，以在代码编辑阶段自动检测 SysY2022E 程序中潜在的逻辑错误和不符合语言规范的地方。
- ◆ 功能描述：静态语义检查是代码质量保障的重要一环，SysY2022-Editor 通过内置的静态语义分析器，在编译之前对 SysY2022E 程序进行详细的语义检查。这些检查不仅限于语法层面，还包括对代码逻辑、类型、作用域等方面的深入分析。
- ◆ 以下是静态语义检查功能的具体描述：
 - a) 类型检查：
 - ◆ 编辑器会检查程序中变量的类型声明和使用，确保所有变量的类型都符合 SysY2022E 语言的规范；
 - ◆ 如果发现类型不匹配或未声明的类型，编辑器会立即标识出错误位置，并提供修复建议。
 - b) 作用域检查：
 - ◆ 编辑器会分析程序中变量、函数和类的作用域，确保它们在使用时处于正确的上下文环境中；
 - ◆ 如果发现变量未声明、函数或类访问了超出其作用域的内容，编辑器会提示相应的错误。
 - c) 逻辑错误检测：

- ◆ 静态语义检查还能够发现一些潜在的逻辑错误，如无限递归、不可达代码、未使用的参数等；
 - ◆ 编辑器会利用静态分析技术来检测这些错误，并为用户提供修复建议。
- d) 实时反馈：
- ◆ 静态语义检查是在用户编写代码的过程中实时进行的，一旦检测到错误，编辑器会立即在用户界面上标识出错误的位置和类型；
 - ◆ 用户可以根据编辑器的提示快速定位并修复问题，提高编程效率。
- e) 修复建议：
- ◆ 对于每个检测到的静态语义错误，编辑器都会提供详细的修复建议；
 - ◆ 这些建议可能包括修改类型声明、调整作用域、重构代码逻辑等；
 - ◆ 编辑器还支持根据用户选择的建议自动完成修复过程，进一步简化用户的操作；
 - ◆ 通过静态语义检查功能，SysY2022E Language Support 能够帮助用户及时发现并修复潜在的逻辑错误和不符合语言规范的地方，提高代码的质量和可维护性。

3.1.5. 修复建议

- ◆ 功能编号：F005
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：中
- ◆ 功能定义：该功能旨在自动分析 SysY2022E 程序中检测到的语法错误和静态语义错误，并基于这些错误提供针对性的修复建议，以帮助用户快速、准确地解决问题。
- ◆ 功能描述：当 SysY2022 扩展语言编辑器（SysY2022-Editor）在代码编辑或编译过程中检测到语法错误或静态语义错误时，它会立即启动修复建议功能。这一功能基于内置的错误分析引擎和最佳实践数据库，能够智能地为用户提供一系列可能的修复方案。
- ◆ 以下是修复建议功能的具体描述：
 - a) 错误检测与识别：
 - ◆ 编辑器在代码编辑或编译时实时检测语法错误和静态语义错误；
 - ◆ 一旦发现错误，编辑器会立即在用户界面上标识出错误的位置和类型。
 - b) 智能修复建议：
 - ◆ 对于每个检测到的错误，编辑器都会基于内置的错误分析引擎和最佳实践数据库提供一系列修复建议；
 - ◆ 这些建议可能包括修改语法结构、调整变量类型、更改函数参数等；
 - ◆ 建议会根据错误的严重性和常见性进行排序，以便用户能够优先处理最重要的问题。

- c) 详细解释与示例：
 - ◆ 对于每个修复建议，编辑器都会提供详细的解释和示例代码，帮助用户理解建议的意图和如何应用它们；
 - ◆ 这有助于用户更好地理解错误的原因和如何修复它们，从而提高代码质量。
- d) 自动修复：
 - ◆ 编辑器支持用户根据提供的建议自动完成修复过程；
 - ◆ 用户只需选择一个建议并点击相应的按钮，编辑器就会自动修改代码以修复错误；
 - ◆ 这一功能大大减少了用户手动修复错误的时间和复杂性。
- e) 用户反馈与改进：
 - ◆ 编辑器允许用户对修复建议进行评估和反馈；
 - ◆ 用户可以标记某个建议是否有效、是否满意等，以便编辑器不断完善其修复建议算法；
 - ◆ 这些反馈数据将被用于优化未来的修复建议功能，提高其准确性和实用性；
 - ◆ 通过这一功能，SysY2022-Editor 不仅能够帮助用户及时发现并修复代码中的错误，还能通过智能的修复建议提高用户的编程效率和代码质量。

3.1.6. 程序错误检查

- ◆ 功能编号：F006
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：中
- ◆ 功能定义：该功能旨在自动检测 SysY2022E 程序中存在的常见错误，以帮助用户提高代码质量和减少潜在的运行时问题。
- ◆ 功能描述：SysY2022E 扩展语言编辑器（SysY2022-Editor）集成了一套全面的程序错误检查机制，能够实时地检测 SysY2022E 程序中可能存在的常见错误。
- ◆ 以下是该错误检查功能的具体描述：
 - a) 实时错误检测：
 - ◆ 在用户编写或修改代码时，编辑器将自动运行错误检查器，实时地分析代码并标识出潜在的问题；
 - ◆ 错误检查器将检查代码的语法、语义和逻辑结构，以发现可能导致程序运行错误的代码片段。
 - b) 常见错误检查：
 - ◆ 死循环：编辑器将分析程序中的循环结构，以检测是否存在无限循环（即死循环），这些循环可能导致程序无法正常终止；

- ◆ 无用变量：编辑器将识别并标记出那些已声明但未在代码中使用的变量，这些变量可能是不必要的，或者需要被重新使用或删除；
 - ◆ 死代码段：编辑器将检测出那些永远不会被执行的代码段（死代码），这些代码可能是冗余的，或者应该被移动到更合适的位置；
 - ◆ 数组越界：编辑器将分析数组访问操作，以确保索引值在有效范围内，避免潜在的数组越界错误。
- c) 详细的错误信息：
- ◆ 当编辑器检测到错误时，它将提供详细的错误信息，包括错误类型、错误位置和可能的修复建议；
 - ◆ 这些信息将直接显示在代码编辑器中，以便用户能够快速定位并修复问题。
- d) 修复建议：
- ◆ 除了提供错误信息外，编辑器还可能为某些错误提供自动修复建议或代码重构建议；
 - ◆ 这些建议将帮助用户快速解决问题，并改进代码的质量和可维护性；
 - ◆ 通过集成这一功能，SysY2022-Editor 能够显著提高 SysY2022E 程序的开发效率和质量，减少因代码错误而导致的运行问题。

3.1.7. 代码重构

- ◆ 功能编号：F007
- ◆ 所属产品编号：SysY2022E Language Support
- ◆ 优先级：低
- ◆ 功能定义：该功能旨在为 SysY2022E 程序提供全面的代码重构支持，帮助用户改善代码结构、提高代码质量和维护性。
- ◆ 功能描述：SysY2022 扩展语言编辑器应提供代码重构功能，以帮助用户优化和重构 SysY2022E 程序。
- ◆ 以下是该功能所涵盖的具体操作及其详细描述：
 - a) 变量/函数改名：
 - ◆ 允许用户更改程序中变量或函数的名称，以确保代码命名的一致性和可读性；
 - ◆ 提供全局搜索和替换功能，以确保所有对该变量或函数的引用都被正确地更新；
 - ◆ 支持批量改名，减少用户手动操作的复杂性。
 - b) 抽取新函数：
 - ◆ 识别代码块中的可重用代码段，并允许用户将这些代码段提取为新的函数；
 - ◆ 提供智能建议，指导用户选择最佳的函数名和参数列表；

- ◆ 提取后，自动更新所有对该代码段的调用，以使用新创建的函数。
- c) 内联函数：
 - ◆ 允许用户将小型、简单的函数内联到其调用点，以减少函数调用的开销；
 - ◆ 评估内联的潜在收益，并为用户提供是否进行内联的建议；
 - ◆ 在内联后，自动删除原始的函数定义和所有对该函数的调用。
- d) 提取局部变量：
 - ◆ 识别表达式或代码块中频繁使用的常量或表达式，并允许用户将其提取为局部变量；
 - ◆ 提供智能命名建议，并根据上下文自动为变量分配合适的类型；
 - ◆ 提取后，自动更新所有对该常量或表达式的引用，以使用新创建的局部变量。
- e) 其他重构操作：
 - ◆ 支持循环重构，如将嵌套循环转换为迭代器等；
 - ◆ 提供条件语句的重构选项，如合并或拆分条件分支；
 - ◆ 允许用户自定义重构规则，以满足特定项目的需求；
 - ◆ 通过这些重构操作，SysY2022-Editor 帮助用户更好地组织代码，减少冗余和复杂性，提高代码的可读性和可维护性。此外，编辑器还提供了可视化的重构界面和详细的重构预览，使用户能够轻松查看重构前后的代码差异，并做出明智的决策。

3.2. 对性能的规定

3.2.1. 精度

数据精度的要求主要关注于编辑器在处理、显示、传输与 SysY2022E 程序相关的数据时的准确性和一致性。以下是关于输入、输出以及传输过程中数据精度的具体要求：

表 9 数据精度要求表

数据		精度规定
输入数据	文本输入	准确接收用户输入，包括字符、数字、特殊符号等，确保无乱码、丢失或重复字符的现象。
	数值输入	数值的输入精度与 SysY2022E 语言的规范一致。包括但不限于整数、浮点数、科学计数法等。
	其他输入	准确解析并处理，确保与 SysY2022E 语言的规范相符。包括但不限

		于布尔值、枚举类型。
输出数据	代码显示	包括字符、格式、缩进、注释等，不应出现乱码、格式错误或丢失信息的情况。
	错误信息	输出详细的错误信息，包括错误的类型、位置、描述等，确保信息的准确性和清晰度。
	提示建议	提供准确的信息和建议，帮助用户更好地理解 and 修改代码。
传输过程	文件传输	确保文件内容的完整性和准确性，不应出现数据丢失或损坏的情况。
	网络通信	确保传输过程中的数据精度和安全性。包括使用合适的数据编码格式、加密传输等。
	外部接口	确保通过这些接口传输的数据的精度和一致性。

3.2.2. 时间特性要求

在开发 SysY2022 扩展语言编辑器时，对时间特性的要求至关重要，因为它们直接影响到用户的使用体验和编辑器的性能。以下是针对响应时间、更新处理时间、数据转换和传送时间、计算时间等关键时间特性的具体要求：

表 10 时间精度要求表

时间		精度规定
响应时间	用户界面响应	控制在毫秒级，确保用户界面的流畅性。迅速响应用户的各种操作，包括但不限于键入代码、点击按钮、选择菜单项。
	错误提示和反馈	尽快向用户显示错误提示和反馈。对于严重错误，编辑器应能够立即中断当前操作并显示错误信息。
更新处理时间	代码自动补全	毫秒级内完成代码分析和建议生成；快速提供代码自动补全和提示功能。
	实时语法检查	实时进行语法检查和静态语义检查，尽快向用户显示检查结果。
	文件保存加载	保存和加载文件时具有较快的处理速度。对于大型文件，采用增量保存和加载技术，减少不必要的等待时间。
数据转换和传送时间	外部数据导入	具有较快的数据转换和导入速度，高效地将外部数据转换为 SysY2022E 程序可识别的格式，并尽快将其加载到编辑器中。
	数据导出和共享	具有较快的处理速度，支持多种数据格式和导出选项。
计算时间	代码分析和优化	采用高效的算法和数据结构，以减少计算时间并提高准确性。
	实时预览和调试	具有较快的计算速度以支持实时反馈。高效处理代码并生成

		结果。
--	--	-----

3.2.3. 灵活性

1) 操作方式的变化

- ◆ 编辑器应支持多种操作方式，包括但不限于键盘操作、鼠标操作、触控操作等，以适应不同用户群体的操作习惯。
- ◆ 提供可定制的用户界面和快捷键设置，允许用户根据自己的喜好调整编辑器的操作方式。
- ◆ 设计直观的交互反馈机制，确保用户在不同操作方式下都能获得一致的体验。

2) 运行环境的变化

- ◆ 编辑器应支持跨平台运行，能够在 Windows、Linux、macOS 等多种操作系统上稳定运行。
- ◆ 适配不同的硬件环境，包括不同性能的计算机、显示器和输入设备。
- ◆ 编辑器应具备自动检测和适应运行环境的能力，确保在不同配置下都能提供稳定的性能。

3) 同其它系统的接口变化

- ◆ 提供开放的 API 和插件接口，允许用户或第三方开发者根据需求扩展编辑器的功能。
- ◆ 支持多种版本控制系统和项目管理系统，方便用户在不同系统间协同工作。
- ◆ 编辑器应能够与其他开发工具（如 IDE、编译器、调试器等）无缝集成，提高开发效率。

4) 精度和有效时限的变化

- ◆ 编辑器应能够支持 SysY2022E 语言规范中不同精度的数据处理需求，如整数、浮点数、高精度计算等。
- ◆ 对于需要长时间运行的复杂任务（如代码重构、性能分析等），编辑器应能够支持异步处理和进度显示，避免用户等待过长时间。

5) 计划的变化或改进

- ◆ 编辑器应具备可扩展的架构和模块化的设计，方便在未来进行功能扩展和性能优化。
- ◆ 设立完善的版本控制机制，确保在更新和升级过程中不会破坏现有功能和数据。
- ◆ 提供用户反馈渠道和技术支持服务，及时响应并解决用户在使用过程中遇到的问题。

为了提供这些灵活性而进行的专门设计部分：

1) **模块化设计：**将编辑器的功能划分为多个独立的模块，每个模块负责实现特定的功能。这种设计方式使得在需要添加新功能或改进现有功能时，只需修改或扩展相应的模块，而无需对整个编辑器进行重构。

2) **插件机制**: 通过提供开放的插件接口和 API, 允许用户或第三方开发者根据需求定制编辑器的功能。这种机制可以极大地扩展编辑器的功能范围, 并满足不同用户群体的个性化需求。

3) **配置化选项**: 提供丰富的配置选项, 允许用户根据自己的喜好和需求调整编辑器的各项参数和设置。这些配置选项可以覆盖编辑器的操作方式、界面风格、性能参数等方面, 从而为用户提供更加灵活和个性化的使用体验。

4) **可扩展的架构**: 采用可扩展的架构和标准化的接口设计, 使得编辑器能够方便地与其他系统和工具进行集成和互操作。这种架构可以确保编辑器在未来能够应对各种变化和挑战, 并保持其竞争力和生命力。

3.3. 输入/输出要求

3.3.1. 输入数据类型

1) 文本输入

- ◆ 媒体: 键盘、文本文件、剪贴板等;
- ◆ 格式: UTF-8、ASCII 等字符编码格式;
- ◆ 数值范围: 无特定数值范围, 支持全字符集;
- ◆ 精度: 字符级别, 确保字符的完整性和准确性。

2) 代码输入

- ◆ 媒体: 同文本输入;
- ◆ 格式: SysY2022E 语言的语法规则;
- ◆ 数值范围: 无特定数值范围, 依赖于代码逻辑和变量定义;
- ◆ 精度: 根据 SysY2022E 语言的规范, 支持整数、浮点数、布尔值等, 并确保其精度。

3) 配置文件输入

- ◆ 媒体: 配置文件 (如 INI、XML、JSON 等);
- ◆ 格式: 配置文件指定的格式, 如键值对、树形结构等;
- ◆ 数值范围: 根据配置文件的定义, 支持各种数据类型和数值范围;
- ◆ 精度: 根据配置文件的定义和数据类型, 确保精度。

3.3.2. 输出数据类型

1) 代码编辑结果

- ◆ 媒体: 显示器;

- ◆ 格式：文本格式，遵循 SysV2022E 语言的语法高亮和格式；
- ◆ 数值范围：无特定数值范围，依赖于编辑的代码内容；
- ◆ 精度：确保代码的准确性和格式的一致性。

2) 错误信息

- ◆ 媒体：显示器、日志文件等；
- ◆ 格式：文本格式，包含错误类型、位置、描述等信息；
- ◆ 数值范围：无特定数值范围，但错误代码可能有预定义的编号范围；
- ◆ 精度：确保错误信息的准确性和清晰度。

3) 状态信息

- ◆ 媒体：显示器、状态栏等；
- ◆ 格式：文本或图标格式，显示编辑器的当前状态（如保存状态、运行模式等）；
- ◆ 数值范围：无特定数值范围，状态信息通常为定性描述；
- ◆ 精度：确保状态信息的准确性和一致性。

3.3.3. 硬拷贝报告

1) 正常结果输出：当编辑器完成代码编辑或执行时，可以将结果打印出来。输出格式可以是纯文本、富文本或 PDF 等，包含代码内容、执行结果等信息。

2) 状态输出：编辑器的状态信息也可以打印成报告，如编辑器的配置信息、使用统计等。输出格式同样可以是文本或 PDF 等。

3) 异常输出：当编辑器遇到错误或异常情况时，可以将错误信息和异常栈打印出来，帮助用户或开发者定位问题。输出格式通常是文本或日志文件。

3.3.4. 图形或显示报告

1) 编辑器：通过图形界面展示代码的结构、执行流程、变量关系等。这些图形可以是流程图、语法树、UML 图等。

2) 输出媒体：显示器，格式可以是位图、矢量图等，根据具体需求选择合适的图形表示方式。

3) 精度要求：确保图形的准确性和清晰度，能够清晰地展示代码的结构和逻辑。

3.3.5. 控制输出量说明

1) 语言特性设定：编辑器可以提供不同的编译和运行选项，允许用户根据需求选择输出详细程度。例如，可以通过命令行参数或者图形界面设置来启用或禁用调试信息的输出。

2) 日志级别控制：编辑器可能支持不同的日志级别，如“错误”、“警告”、“信息”和“调试”。用户可以根据需求选择输出的日志级别，以控制输出量。

3) 编译器反馈: SysY2022 的编辑器可能包括对源代码的静态分析功能, 提供编译前错误检查、代码优化建议等。这些功能可以通过用户界面中的选项来开启或关闭, 从而影响输出的信息量。

4) 可视化工具: 编辑器可能包含图形界面的可视化工具, 如语法树可视化、中间代码表示等。这些工具通常可选使用, 用户可以根据需要来决定是否生成这些额外信息。

5) 性能分析工具: 对于更高级的使用场景, SysY2022 编辑器可能包括性能分析工具, 帮助用户理解程序执行的具体细节, 如执行时间、内存使用等。这些工具的输出也是可控的。

6) 用户自定义设置: 高级用户或开发者可以通过修改编辑器的配置文件来自定义输出设置, 比如定义特定的输出格式、选择性地记录某些事件或操作的细节等。

3.4. 故障处理要求

3.4.1. 软件故障

1) 崩溃或无响应

- ◆ 后果: 用户无法进行正常的编辑工作, 可能导致数据丢失或工作进度受阻。
- ◆ 处理要求:
 - a) 实现自动保存功能, 定期保存编辑内容, 减少数据丢失风险。
 - b) 监控软件运行状态, 出现异常时提供重启或恢复选项。
 - c) 收集崩溃日志, 便于后续分析和问题定位。

2) 功能异常

- ◆ 后果: 如代码高亮错误、自动补全失效等, 影响用户的使用体验和工作效率。
- ◆ 处理要求:
 - a) 提供详细的错误提示, 帮助用户理解问题所在。
 - b) 设计故障恢复机制, 如重置功能设置、清理缓存等。
 - c) 定期更新和维护软件, 修复已知问题和提升性能。

3) 兼容性问题

- ◆ 后果: 在特定操作系统或硬件环境下, 软件无法正常运行或存在功能缺陷。
- ◆ 处理要求:
 - a) 在多种操作系统和硬件环境下进行测试, 确保软件的兼容性。
 - b) 提供兼容性指南或解决方案, 帮助用户解决兼容性问题。
 - c) 及时更新软件以支持新的操作系统和硬件。

4) 安全漏洞

- ◆ 后果：可能导致用户数据泄露、软件被恶意攻击或滥用。
- ◆ 处理要求：
 - a) 实施严格的安全审查和测试流程，确保软件的安全性。
 - b) 及时修复已知的安全漏洞，并发布安全补丁。
 - c) 提供用户账户管理和权限控制功能，保护用户数据安全。

3.4.2. 硬件故障

1) 输入设备故障

- ◆ 后果：如键盘、鼠标等无法正常工作，影响用户的编辑操作。
- ◆ 处理要求：
 - a) 提供多种输入方式选项（如触摸屏、手写板等），确保用户在不同输入设备故障时仍能进行编辑。
 - b) 提供外部设备连接接口（如 USB 接口），支持用户连接外部输入设备。

2) 显示设备故障

- ◆ 后果：如显示器无法显示或显示异常，影响用户查看编辑内容和界面操作。
- ◆ 处理要求：
 - a) 提供软件界面缩放和字体大小调整功能，适应不同分辨率和显示比例的显示器。
 - b) 提供语音提示和音频输出功能，帮助用户在无法查看显示器时进行操作。

3) 存储设备故障

- ◆ 后果：可能导致用户数据丢失或无法访问，影响编辑工作的连续性。
- ◆ 处理要求：
 - a) 实现数据备份和恢复功能，定期将用户数据备份至云端或外部存储设备。
 - b) 提供数据迁移和导入导出功能，支持用户在不同存储设备间迁移数据。

3.5. 其他要求

3.5.1. 易读性

为了提高用户的使用体验，SysY2022 扩展语言编辑器需要具备良好的易读性。这包括：

- ◆ 直观的界面设计：采用简洁、直观的界面设计，使用户能够轻松上手并快速找到所需功能。
- ◆ 清晰的代码高亮：提供多种代码高亮方案，帮助用户快速识别代码结构和语法错误。
- ◆ 友好的错误提示：当用户在编辑代码时出现错误时，提供详细且易于理解的错误提示信息。

3.5.2. 可靠性

SysY2022 扩展语言编辑器需要保证高度的可靠性，以满足用户的需求。这包括：

- ◆ 严格的测试：在开发过程中进行严格的单元测试、集成测试和系统测试，确保编辑器的稳定性和性能。
- ◆ 错误处理：为编辑器设计完善的错误处理机制，当遇到异常情况时能够自动恢复或提供解决方案。
- ◆ 数据备份：定期备份用户数据和配置信息，以防止数据丢失或损坏

3.5.3. 使用方便性

- ◆ 设计直观的用户界面，简化操作流程。
- ◆ 提供丰富的帮助文档和教程，便于用户学习和使用。
- ◆ 增加自动化功能，比如代码补全、错误提示、一键编译运行等，减少用户的重复劳动。

3.5.4. 可维护性

- ◆ 采用模块化的设计，便于对系统的维护和升级。
- ◆ 编写清晰的代码和文档，确保开发人员能够快速理解和维护。
- ◆ 提供良好的技术支持和用户反馈机制，快速响应用户的问题和需求。

3.5.5. 可补充性

- ◆ 允许用户或开发者添加新的功能或插件。
- ◆ 提供开放的 API 接口和扩展框架，容易集成第三方工具和库。
- ◆ 设计软件时预留足够的扩展点，便于未来的功能扩展和技术升级。

3.5.6. 安全保密

- ◆ 实施严格的访问控制和用户身份验证机制。
- ◆ 对敏感数据进行加密，确保数据传输和存储的安全性。
- ◆ 定期进行安全审计和漏洞扫描，保持软件的安全性。
- ◆ 提供安全的更新机制，确保软件的安全补丁能够及时部署。

4. 运行环境规定

4.1. 设备

4.1.1. 处理器及内存

1) 处理器型号：例如 Intel Core i7 系列，提供强大的计算能力和多任务处理能力，适用于运行复杂的编辑器或 IDE；

2) 内存容量：8GB 以上，足够的内存可以确保编辑器流畅运行，并处理大型文件或多任务操作。

4.1.2. 外存

1) 外存容量：1TB 以上，用于快速加载和保存文件，提高整体系统响应速度；

2) 类型：联机存储设备，直接连接到计算机系统上，可随时访问；

3) 设备型号：例如 Samsung 980 PRO NVMe SSD

4) 数量：1 块，可根据需要增加更多存储空间或组建 RAID 阵列。

4.1.3. 输入及输出设备

1) 输入设备

- ◆ 键盘：标准 USB 键盘，联机设备，型号可根据用户偏好选择；
- ◆ 鼠标：高精度光电鼠标，联机设备，型号可根据用户偏好选择；

2) 输出设备

显示器：至少一台高清显示器，例如 Dell Ultrasharp 系列，联机设备；

4.1.4. 数据通信设备

1) 网络适配器：集成或独立的千兆以太网适配器或 Wi-Fi 6 适配器，用于连接到互联网或局域网，进行数据传输和更新等操作；

2) USB 端口：多个 USB 3.0 或以上版本的端口，用于连接外部设备如 U 盘、移动硬盘等。

4.2. 支持软件

1) 操作系统：Windows

2) 编译程序：

- ◆ Visual Studio Code: 用于编写源代码。
- ◆ antlr4: 解析器生成器, 用于构建编译器、解释器和语言分析工具, 由 Terence Parr 开发, 能够将语法定义转换为能够识别和处理相应语言的解析器代码。
- ◆ mingw: 自动化构建工具, 用于控制编译过程, 确保所有必要的文件都被正确编译和链接。
- ◆ gcc/g++: 作为 C/C++ 编译器, 用于编译由 antlr4 生成的 C 语言源代码, 以及其他相关的 C/C++ 代码。

3) 测试软件: Visual Studio Code, SysY2022E 扩展语言编辑器。

5. 双方签字

需求方（需方）：

开发方（供方）：

日期：