SysY2022E 扩展语言编辑器

SysY2022E Language Support

系统测试说明书

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

1.1. 编写目的

本测试说明文档的编写目的是为 SysY2022E 编辑器系统提供全面、系统的测试指南，确保编辑器的各项功能能够按照需求描述中的要求正确实现，并发现可能存在的问题。本文档适用于测试团队、开发人员以及项目管理人员，旨在保证软件质量，并为后续版本更新和优化提供参考。

1.2. 范围

本文档覆盖了 SysY2022E 编辑器系统的全部功能测试，包括 IDE 集成、语法高亮与悬浮提示、语法检查与错误提示、静态语义检查、修复建议、程序错误检查、代码重构等。本文档在整个项目中充当测试环节的指导手册，旨在确保编辑器满足预定的功能需求并达到预期的性能标准。

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. 引用标准

1）IEEE 829-1998（已被 IEEE 829-2008 取代）：软件测试文档标准

提供了编写软件测试计划、测试用例、测试设计规格和测试过程规格等文档的指南，确保测试文档的完整性和规范性。

2）ISO/IEC/IEEE 29119-1:2013：软件与系统工程——软件测试——第 1 部分：概念和定义

定义了软件测试的基本概念、原则和术语，为软件测试活动提供了统一的语义和理解基础。

3）ISO/IEC/IEEE 29119-2:2013：软件与系统工程——软件测试——第 2 部分：测试过程

描述了软件测试的过程，包括测试计划、测试设计与实现、测试执行、测试评估等活动，为规范的软件测试流程提供了指导。

4）ISO/IEC/IEEE 29119-3:2013：软件与系统工程——软件测试——第 3 部分：测试文档编制

规定了软件测试过程中应编制的文档类型和内容，包括测试计划、测试用例规格、测试过程规格等，确保测试文档的一致性和可追溯性。

5）ISO/IEC/IEEE 29119-4:2015：软件与系统工程——软件测试——第 4 部分：测试技术

提供了软件测试中常用的测试技术，包括黑盒测试、灰盒测试和白盒测试等，以及测试数据的选择和测试用例的设计方法。

6）ISO/IEC/IEEE 29119-5:2013：软件与系统工程——软件测试——第 5 部分：关键字驱动测试

介绍了基于关键字的测试方法，通过定义和使用关键字来指导和自动化测试用例的设计和执行，提高测试的效率和质量。

1.5.参考资料

[1] IEEE. (1998). IEEE 829-1998 (Superseded by IEEE 829-2008, with potential newer versions available): Standard for Software Test Documentation. New York: Institute of Electrical and Electronics Engineers.

[2] ISO/IEC/IEEE. (2013). ISO/IEC/IEEE 29119-1:2013: Software and systems engineering -- Software testing -- Part 1: Concepts and definitions. Geneva: International Organization for Standardization; Washington, D.C.: International Electrotechnical Commission; New York: Institute of Electrical and Electronics Engineers.

[3] ISO/IEC/IEEE. (2013). ISO/IEC/IEEE 29119-2:2013: Software and systems engineering -- Software testing -- Part 2: Test processes. Geneva: International Organization for Standardization; Washington, D.C.: International Electrotechnical Commission; New York: Institute of Electrical and Electronics Engineers.

[4] ISO/IEC/IEEE. (2013). ISO/IEC/IEEE 29119-3:2013: Software and systems engineering -- Software testing -- Part 3: Test documentation. Geneva: International Organization for Standardization; Washington, D.C.: International Electrotechnical Commission; New York: Institute of Electrical and Electronics Engineers.

[5] ISO/IEC/IEEE. (2015). ISO/IEC/IEEE 29119-4:2015: Software and systems engineering -- Software testing -- Part 4: Test techniques. Geneva: International Organization for Standardization; Washington, D.C.: International Electrotechnical Commission; New York: Institute of Electrical and Electronics Engineers.

[6] ISO/IEC/IEEE. (2013). ISO/IEC/IEEE 29119-5:2013: Software and systems engineering -- Software testing -- Part 5: Keyword-driven testing. Geneva: International Organization for Standardization; Washington, D.C.: International Electrotechnical Commission; New York: Institute of Electrical and Electronics Engineers.

2. 测试项目

1) 用户能够在 VS Code 中通过 VISX 安装 sysY 扩展插件。

2) 用户使用编辑器窗口能看到语法高亮和悬浮提示。

3) 编辑器能自动对代码进行语法检查和错误提示。

4) 编辑器能进行静态语义检查并给出修复建议。

5) 编辑器能生成程序错误检查报告。

6) 编辑器能实现自动辅助编辑功能，如补全代码、查看引用等。

7) 编辑器能实现代码重构功能。

2.1. 单元测试——接口测试用例

1) 被测试对象的介绍

本次测试的对象是 SysY2022E 扩展语言编辑器的两个外部接口：接口 A（vscode 扩展）和接口 B（编辑器窗口）。接口 A 主要负责通过 VSIX 安装 sysY 的扩展插件，使得用户能够在 Visual Studio Code 环境中使用 sysY 提供的功能。接口 B 则是用户在编辑器窗口中进行编码时，与插件进行人机交互的接口，包括语法高亮、语法语义错误提示、修复建议、程序错误检查、代码补全、代码重构等功能。

2) 测试范围与目的

测试范围主要集中在接口 A 的安装功能以及接口 B 的编辑器交互功能。测试目的是验证这些功能是否能够按照预期工作，包括但不限于：扩展插件能否成功安装、编辑器中的语法高亮是否正确、鼠标悬停提示信息是否准确、错误检查、修复建议和代码重构是否有效，以及代码补全和查找引用等辅助编辑功能是否正常。

3) 功能测试用例

表 2 单元测试用例表

接口 A（外部接口）	**vscode 扩展**	
输入/动作	期望的输出/相应	实际情况
从 VSIX 安装扩展	成功安装 sysY 扩展插件，无错误提示，插件可在 VS Code	吻合，成功安装插件，可在 VS Code 中看到并使用 sysY

	中正常使用	插件
接口 B（外部接口）	**编辑器窗口**	
输入/动作	期望的输出/相应	实际情况
键入关键字、变量、函数等不同类型的元素	编辑器根据元素类型显示不同颜色、字体，实现语法高亮，便于用户区分代码结构	吻合，各类元素均正确高亮显示
鼠标悬停在标识符上	显示该标识符的定义、类型等相关信息或提示，帮助用户理解代码	吻合，鼠标悬停时能够正确显示标识符信息
鼠标悬停在错误语法上	显示错误原因，并提供可能的修复建议，帮助用户快速定位和解决问题	吻合，能够准确指出语法错误并提供修复建议
鼠标悬停在错误语义上	识别语义错误，并给出相应的错误原因和修复建议	吻合，能够识别语义错误并提供帮助信息
输入命令 Run Analysis	生成程序错误检查报告，列出代码中可能存在的问题和改进建议	吻合，成功生成详细的错误检查报告
键入新的函数声明、变量声明等代码内容	编辑器自动辅助编辑，实现如代码补全、查找引用等功能，提高编码效率	吻合，编码时能够自动提供代码补全和查找引用等功能
鼠标右键函数，点击重构，选择具体的代码重构方式	编辑器根据选择的方式自动进行代码重构	吻合，成功进行代码重构

2.2. 集合测试——功能测试用例

1） 被测试对象的介绍

本次功能测试针对 SysY2022E 扩展语言编辑器的核心功能，包括代码编辑、语法高亮、错误检查、修复建议、代码补全、代码重构等。sysY 扩展旨在为用户提供一套高效、智能的代码编辑环境，帮助用户提升编程效率。

2) 测试范围与目的

测试范围涵盖 sysY 编辑器的语法高亮和悬浮提示、错误检查、修复建议、代码重构、自动辅助编辑等核心功能。测试目的是验证这些功能是否能够正常运行，并且符合用户的期望。

3) 功能测试用例

表 3 集合测试用例表

功能 A 描述	**语法高亮和悬浮提示**	
用例目的	测试编辑器的语法高亮和悬浮提示功能是否正常工作	
前提条件	sysY 扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
新建 test.sy，输入 int fun(int a,int b) { int c = a + b ; return c; }	编辑器根据变量类型自动进行语法高亮	吻合，各类代码元素均正确高亮显示
将鼠标悬浮在 fun 上面	悬浮窗 Symbol lnformation 显示 name,type,definition 等信息	吻合，各项信息均能在悬浮窗中正确显示
功能 B 描述	**语法检查与错误提示**	
用例目的	测试编辑器能够检查语法错误并提示出错的位置等信息	
前提条件	sysY 扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况

在 test.sy 中输入：
const int qq = 200;
qq = 300;
	在第二个 qq 处有红线提示语法错误，鼠标悬浮可提示具体的错误原因	与期望相吻合，提示出常量不能修改的信息

在 test.sy 中输入：m=1;	字符 m 处红线提示错误，鼠标悬浮提示变量未定义	与期望相吻合，提示出正确的报错信息
功能 C 描述	**静态语义检查**	
用例目的	测试编辑器能够进行类型和作用域等静态语义检查	
前提条件	sysY 扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
在上述 test.sy 代码基础上写入 fun();	fun 红线提示错误，鼠标悬浮提示函数的形参数量错误	与期望相吻合，提示出正确的报错信息
在上述 test.sy 代码基础上写入： float e = 3.14;	fun 红线提示错误，鼠标悬浮提示函数的形参的变量类型错误	与期望相吻合，提示出正确的报错信息

int b=1000;fun(e,b);		

功能D描述	**修复建议**	
用例目的	测试对于发现语法错误和静态语义错误，编辑器能够提供修复建议，并能够根据用户选择的建议自动完成程序修复	
前提条件	sysY扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
将鼠标移动到fun(e,b);	悬浮窗显示修复建议，检查变量e的类型	与期望相吻合，成功对语义错误给出修复建议

功能E描述	**程序错误检查**	
用例目的	测试编辑器能够检查程序中的常见错误，包括但不限于死循环、无用变量、死代码段、数组越界等。	
前提条件	sysY扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
写下完整的test.sy测试代码，包含多种程序错误类型，命令输入Run Analysis	生成错误检查txt文档，显示数组越界、无用变量等程序错误类型	与期望相吻合，成功生成文档并记录程序错误与类型

功能F描述	**代码重构**	
用例目的	测试编辑器能够支持代码重构功能，包括变量/函数改名、抽取新函数、内联函数、提取局部变量等。	
前提条件	sysY扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
在上述test.sy代码的基础上添加函数 void myFunction() { int temp = 100;}右键函数名，点击重构，选择抽取为新函数	成功抽取新函数， void void newFunction() {myFunction()} myFunction() {int temp = 100;}	与期望相吻合，代码重构成功
右键myFunction，点击重构，选择转换为大写	函数名myFunction成功变为大写函数名	与期望相吻合，代码重构成功
右键myFunction，点击重构，选择内联函数	main函数中调用myFunction处变为myFunction的内容	与期望相吻合，代码重构成功

功能G描述	**辅助编辑**	
用例目的	验证编辑器自动辅助编辑，实现如代码补全、查找引用等功能，提高编码效率	
前提条件	sysY扩展插件已正确安装并激活	
输入/动作	期望的输出/相应	实际情况
右键main函数的fun(c,b);	成功转到fun函数定义处	与期望相吻合，辅助编辑成

点击转到定义		功
编写 test.sy 测试代码，利用编辑器的自动缩进、代码折叠等功能辅助编辑	成功使用各种功能，提升代码编写效率	与期望相吻合，辅助编辑成功

2.3. 系统测试

系统测试是在真实的系统工作环境下通过与系统的需求定义作比较,检验系统的文档和系统与文档和用户需求文档和系统设计文档和系统设计是否符合与之对应的各种文档的描述,发现其中的文档和系统设计是否符合与之对应的各种问题和与之对应的各种与之对应的各种错误。本次系统测试的目的是确保 SysY2022E 扩展语言编辑器的整体性能和稳定性，以及各个模块之间的协调性和互操作性。

测试范围包括整个编辑器的所有功能，如语法高亮、错误检查、代码补全、代码重构等，并测试在不同操作系统、不同硬件配置下的表现。同时，还会对编辑器的性能进行测试，如启动速度、文件打开速度、代码编辑响应速度等。

测试方法主要包括接口测试和功能测试。通过输入各种有效和无效的数据来验证编辑器的功能和性能，确保代码的质量和稳定性。

根据测试用例的步骤，小组成员完成了系统测试，证实了 SysY2022E 扩展语言编辑器各个功能的有效性和正确性,确保为用户提供了一个高效、智能的代码编辑环境。以下是各个测试步骤的截图记录。

1) 在 vscode 中通过本地 vsix 安装 sysY 扩展插件，确保 vsix 插件的可用性与兼容性。

图 1 SysY 插件在 vscode 安装成功

2) 语法高亮和悬浮提示

- 新建 test.sy 文件，输入代码，查看语法高亮与悬浮提示信息。

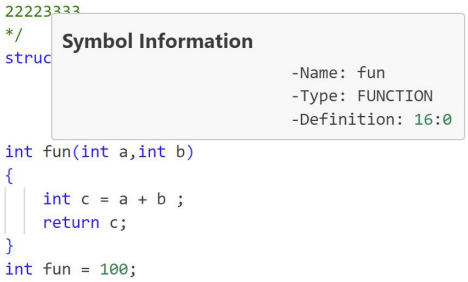

图 2 语法高亮和悬浮提示

3) 语法检查与错误提示

- 定义整型常量 qq，尝试修改常量，测试语法检查与错误提示功能。

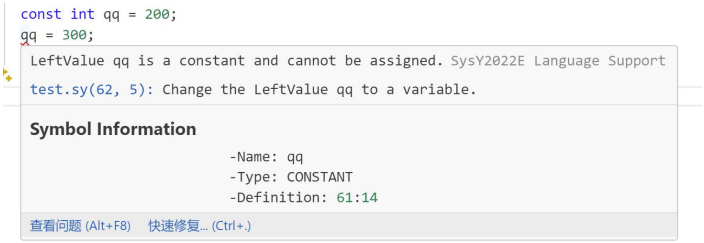

图 3 修改常量错误提示

- 在未定义变量 m 的情况下给 m 赋值，测试语法检查与错误提示功能。

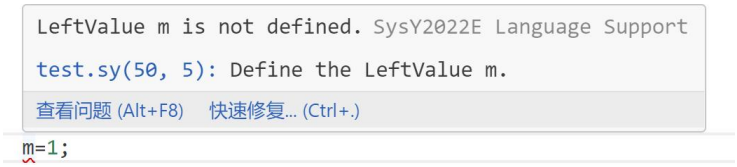

图 4 变量未定义的错误提示

4) 静态语义检查

- 定义函数 fun(int,int);在 main 函数中尝试用错误的形参数量调用函数。

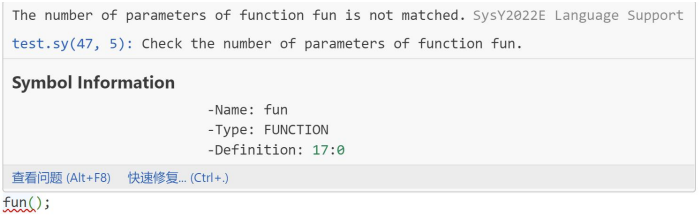

图 5 形参数量错误提示

- 在 main 函数中尝试用错误的形参类型调用 fun 函数。

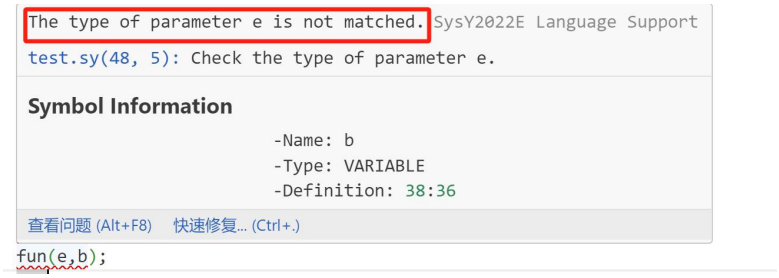

图 6 形参类型错误提示

5) 修复建议

- 查看 fun(e,b);的报错信息，已自动提示修复建议。

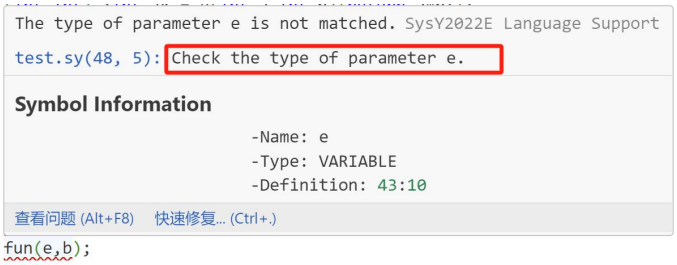

图 7 修改错误的形参类型的建议

6) 程序错误检查

- 保存完整的 test.sy 代码，输入命令 Run Analysis 启动程序错误检查。

图 8 检查出警告与错误信息

- 点击自动生成的 check_output.txt 文档，查看检错报告。

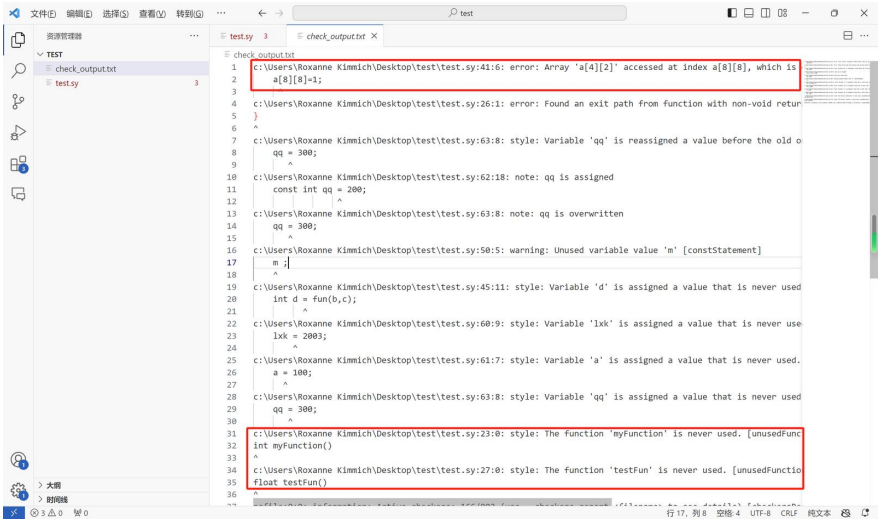

图 9 查看具体的检测报告

7） 代码重构

- 右键函数 myFunction，点击重构，选择具体的代码重构类型，查看多种形式的重构效果。

图 10 右键函数并点击重构

图 11 选择重构类型

```
void void newFunction() {
myFunction()
}
myFunction()
{
  int temp = 100;

}
```

图 12 抽取为新函数

```
void MYFUNCTION()
{
  int temp = 100;

}
```

图 13 转换为大写

```
void myFunction()
{
  int temp = 100;

}
float testFun()
{
  return 100 ;
}
int main()
{
    Person person;
    person.age = 20;
    person.score = 20;

    (int,int)->int jk = ^(int a,int b){return a*b;};
    //jk(person.age,person.score);

    int a[4][2] = {1,2,3,4,5,6,7,8};
    a[1000][0]= 100;

  int temp = 100;
```

图 14 内联函数

8） 辅助编辑

- 定义好 fun 函数后，在 main 函数中调用 fun 函数，右键可转到定义。

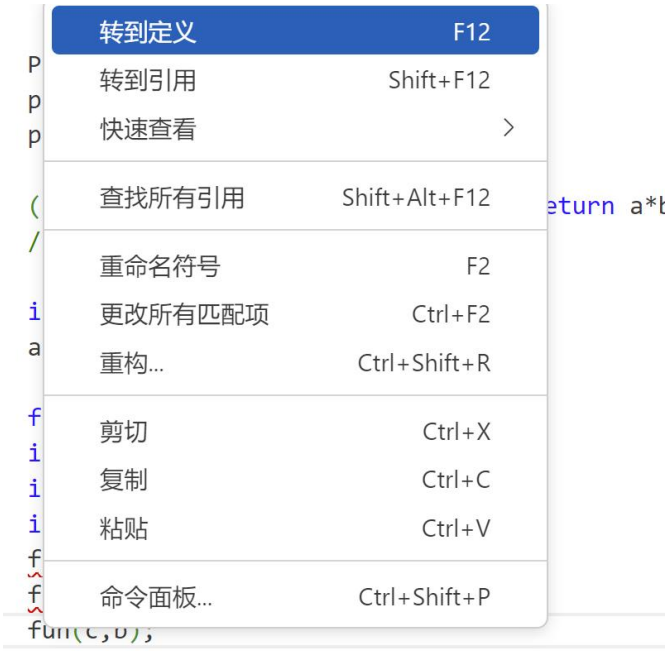

图 15 转到定义

- 点击代码块旁边的“>”符号，进行代码块折叠，便于清晰查看代码结构。

```
> struct Person{ ···
  };

  int fun(int a,int b)
> { ···
  }
```

图 16 代码折叠

3 评价

3.1 软件能力

SysY2022E 扩展语言编辑器在软件能力方面展现出了卓越的表现。从开发者和用户的角度来看，这款编辑器不仅提供了全面的基本代码编辑功能，更融入了众多前沿的智能化特性，从而极大地提升了编程的效率和体验。

SysY2022E 的语法高亮功能非常精确，无论是变量、函数还是关键字，都能以不同的颜色准确标注，使得代码结构一目了然。这一功能能够帮助开发者更快地理解和修改代码，减少错误和调试时间。

通过鼠标悬停提示信息功能，SysY2022E 为用户提供了极大的便利。当鼠标悬停在某个

函数或变量上时，编辑器会自动显示出其相关的详细信息，如参数列表、返回值类型等。这不仅有助于程序员更快地理解代码细节，还能在编写过程中提供即时的参考和帮助。

同时，SysY2022E 还具备强大的错误检查和修复建议功能。在编写代码的过程中，编辑器会实时检查语法和逻辑错误，并在发现问题时给出明确的提示和修复建议。这一功能极大地提升了代码的质量和可靠性，使得程序员能够在编写过程中及时发现并修正错误，避免后续出现更复杂的问题。

除了上述功能外，SysY2022E 还提供了代码重构和辅助编辑功能。代码重构能够帮助程序员在不改变代码外在行为的前提下，优化其内部结构，提高可读性和可维护性。而辅助编辑功能则包括自动补全、代码块折叠等，能够进一步提高编程的效率和便捷性。

总的来说，SysY2022E 扩展语言编辑器以其全面的功能和智能化的特性，充分满足了开发者在代码编辑、调试和优化方面的需求。它不仅提升了编程的效率和体验，还提供了强大的技术支持和个性化的编程服务。然而，尽管 SysY2022E 在软件能力方面表现出色，但我们也需要关注其可能存在的缺陷和限制，以便更全面地评估其性能和适用性。

3.2 缺陷和限制

1）数据资源与测试限制

虽然 SysY2022E 在功能设计和实现上表现出了高度的智能化和实用性，但其在数据资源和测试方面仍存在一定的限制。由于软件开发和测试过程中可利用的数据资源有限，可能导致某些特定场景或边缘情况下的问题未被充分暴露和测试。为了缓解这一问题，我们将在未来收集更多实际使用场景的数据，以便对编辑器进行更全面的测试和优化。

2）硬件与运行环境限制

SysY2022E 主要是针对标准的计算机环境进行设计，对于低配置设备或特定操作系统可能存在兼容性问题。虽然其在大多数主流操作系统和硬件配置上表现良好，但在一些特定环境下可能会出现性能下降或功能受限的情况。因此，在使用前，我们建议用户确认其系统配置和环境要求是否符合编辑器的运行需求。

3）网络依赖

虽然 SysY2022E 的核心编辑功能不依赖网络，但在扩展插件版本更新时需要网络连接。在无网络环境下，版本更新功能可能无法正常使用，从而影响用户的编程体验。为了应对这

一问题，我们建议用户在有网络连接的情况下使用扩展插件，并及时进行版本更新。

3.3 软件质量属性

1）用户界面与交互设计

SysY2022E 的用户界面设计简洁直观，降低了用户的学习成本。其交互设计充分考虑了程序员的使用习惯和需求，使得各项功能操作便捷、高效。通过合理的布局和直观的图标设计，编辑器为用户提供了愉悦的视觉体验和顺畅的操作流程。

2）系统响应与性能表现

在系统响应和性能表现方面，SysY2022E 展现出了优异的表现。无论是在代码高亮、错误检查还是代码重构等方面，编辑器都能提供即时的反馈和流畅的操作体验。其高效的代码解析和渲染能力确保了即使在处理大型项目时也能保持稳定的性能表现。

3）可扩展性与维护性

SysY2022E 具有良好的可扩展性和维护性。通过模块化设计和清晰的代码结构，编辑器使得未来功能的增加和修改变得更加容易。同时，其提供的丰富的 API 和插件接口也为用户提供了更多的自定义和扩展空间。这种设计思路不仅有助于满足用户多样化的需求，还降低了软件的维护成本。

3.4 总结

综上所述，SysY2022E 扩展语言编辑器凭借其强大的功能和优异的性能在编程领域展现出了显著的竞争力。它不仅提供了丰富的编辑和调试工具，还通过智能化的设计优化了编程流程，从而大幅提升了开发效率。其简洁直观的用户界面、高效的系统响应以及良好的可扩展性和维护性都使得这款编辑器成为了程序员们的得力助手。

然而，我们也应看到 SysY2022E 在数据资源与测试、硬件与运行环境以及网络依赖等方面存在的限制和挑战。为了进一步提升编辑器的性能和适用性，我们建议在未来加强与实际使用场景的对接，收集更多的测试数据以优化算法和模型；同时，也应关注不同硬件和操作系统环境下的兼容性测试以确保更广泛的用户群体能够顺畅使用编辑器。

展望未来，随着技术的不断进步和用户需求的日益多样化，我们相信 SysY2022E 将持续

进化并融入更多创新功能以满足程序员的专业需求。通过不断优化现有功能和引入新技术手段如机器学习、自然语言处理等，SysY2022E 有望在未来编程领域发挥更大的作用并推动行业的持续发展。