****

本科毕业设计(论文)

GRADUATION DESIGN(THESIS)

|  |  |
| --- | --- |
| 题 目： | 面向C语言的编译错误增强提示方法研究与实现 |
| 学生姓名： | 罗龙光 |
| 指导教师： | 李洪东 |
| 学 院： | 计算机学院 |
| 专业班级： | 信息安全1801 |

本科生院制

2022年5月

# 面向C语言的编译错误增强提示方法研究与实现

# 摘 要

C语言是本科计算机相关专业学生最常见的编程入门语言之一，它简洁紧凑，灵活方便。作为强类型语言，也有利于培养初学者的良好编程习惯。但是对于初学者而言，学习编程往往并不是一帆风顺的。调试是学习编程过程中的一个重大挑战，特别是，如何学习并理解编译器的报错信息。初学者面对编译器给出的反馈信息往往不知所措，不同的学生面对同类型的问题很可能会提出多个相似的问题。我们想知道，能否通过对编译器所反馈的信息进行增强提示，来帮助学生解决调试编译错误代码的问题。

本文介绍了收集到的编译错误信息的分析结论，以及增强提示服务的开发背景及意义，同时详细描述了该服务的总体设计以及具体实现。

在CSUOJ作为数据集的背景下，我们收集并整理了中南大学21届计算机学院学生的C语言代码编译信息，提取了其中的编译错误信息，并按照出现次数、错误类型等条目进行归类整理。对归类的编译信息做出分析，以期对教学实践有所帮助。我们对现有编译错误信息改进方式进行了总结，并搜集了常见的编译错误提示消息。在GCC编译信息提示的基础上，结合所收集的信息，给出了一个服务以对C语言的编译错误信息进行增强提示。对于常见的简单错误，直接展示服务端所储存的增强提示;对于复杂错误，在服务端使用自定义逻辑的方式，具体错误具体处理。最后将服务部署在平台上，令前端展示出对编译器反馈信息的增强提示。

**关键词**： C语言 编译错误 调试 增强提示

# **Research and Implementation of C Language-Oriented Compilation Error Enhancement Prompt Method**

# **ABSTRACT**

C language is one of the most common programming entry languages ​​for undergraduate computer-related majors. It is simple, compact, flexible and convenient. As a strongly typed language, it is also conducive to cultivating good programming habits for beginners. But for beginners, learning to program is often not smooth sailing. Debugging is a major challenge in learning to program, and in particular, how to learn and understand compiler error messages. Beginners are often overwhelmed by the feedback given by the compiler, and different students are likely to ask multiple similar questions facing the same type of problem. We wondered if we could help students solve the problem of debugging code that compiles errors by enhancing the feedback from the compiler.

This paper introduces the analysis conclusion of the collected compilation error information, as well as the development background and significance of the enhanced prompt service, and describes the overall design and specific implementation of the service in detail.

We summarize the ways to improve existing compilation error messages, and collect common compilation error messages. On the basis of the GCC compilation information prompt, combined with the collected information, a service is given to enhance the prompt of the C language compilation error message. For common simple errors, the enhanced tips stored on the server are directly displayed; for complex errors, the server uses custom logic to handle specific errors. Finally, the service is deployed on the platform, so that the front end displays enhanced prompts for compiler feedback.

**Key words:** C language compilation errors debug enhancement tips

目录

[摘 要 I](#_Toc102952256)

[ABSTRACT II](#_Toc102952257)

[第一章 绪论 5](#_Toc102952258)

[1.1 编译错误增强提示的作用及意义 5](#_Toc102952259)

[1.2 背景知识介绍 5](#_Toc102952260)

[1.2.1 编译错误的分类方式 5](#_Toc102952261)

[1.2.2 C语言程序设计中常见的错误类型 6](#_Toc102952262)

[1.3 国外在增强提示方面的研究 6](#_Toc102952263)

[1.4 国内在增强提示方面的研究 7](#_Toc102952264)

[1.5 论文组织结构 7](#_Toc102952265)

[第二章 相关技术介绍 9](#_Toc102952266)

[2.1 Vue.js 9](#_Toc102952267)

[2.2 Koa 9](#_Toc102952268)

[2.2.1 中间件 9](#_Toc102952269)

[2.2.2 洋葱模型 9](#_Toc102952270)

[2.3 Navicat 10](#_Toc102952271)

[2.4 Element Plus 10](#_Toc102952272)

[2.5 Axios 10](#_Toc102952273)

[2.6 本章小结 11](#_Toc102952274)

[第三章 编译错误研究 12](#_Toc102952275)

[3.1 统计概览 12](#_Toc102952276)

[3.2 错误分析 13](#_Toc102952277)

[3.2.1 简单错误 13](#_Toc102952278)

[3.2.1 复杂错误 14](#_Toc102952279)

[3.3 增强提示方案 16](#_Toc102952280)

[3.3.1 简单错误的增强提示 16](#_Toc102952281)

[3.3.1 复杂错误的增强提示 17](#_Toc102952282)

[3.4 本章小结 17](#_Toc102952283)

[第四章 需求分析与总体设计 18](#_Toc102952284)

[4.1服务的需求分析以及细分能力 18](#_Toc102952285)

[4.1.1 需求分析 18](#_Toc102952286)

[4.1.2 服务的细分能力 19](#_Toc102952287)

[4.2服务的体系结构 19](#_Toc102952288)

[4.3服务的运行环境 20](#_Toc102952289)

[4.3.1 数据收集运行环境 20](#_Toc102952290)

[4.3.2 服务端运行环境 20](#_Toc102952291)

[4.3.3 客户端运行环境 20](#_Toc102952292)

[4.4 本章小结 20](#_Toc102952293)

[第五章 详细设计与实现 21](#_Toc102952294)

[5.1 数据收集详细设计及实现 21](#_Toc102952295)

[5.1.1 获取所有的编译器反馈信息 21](#_Toc102952296)

[5.1.2 编译错误统计处理 23](#_Toc102952297)

[5.1.3 增强提示编写 28](#_Toc102952298)

[5.2 服务端详细设计与实现 29](#_Toc102952299)

[5.2.1 http通信模块 30](#_Toc102952300)

[5.2.2 获取增强提示 30](#_Toc102952301)

[5.2.3 服务复杂度分析 32](#_Toc102952302)

[5.3 客户端详细设计及实现 32](#_Toc102952303)

[5.3.1客户端界面 33](#_Toc102952304)

[5.3.2客户端逻辑 34](#_Toc102952305)

[5.4 本章小结 35](#_Toc102952306)

[第六章 总结与展望 36](#_Toc102952307)

[6.1 工作总结 36](#_Toc102952308)

[6.2 工作的局限性 36](#_Toc102952309)

[6.3 工作展望 37](#_Toc102952310)

[致 谢 38](#_Toc102952311)

[参考文献 39](#_Toc102952312)

# 绪论

## **1.1 编译错误增强提示的作用及意义**

计算机编程已被证明给学生带来了许多困难，它导致了学生的低积极性，低成功率。调试是学习编程过程中的一个挑战，特别是，如何学习并理解编译器的报错信息。而增强提示可以对此带来一定的帮助。

1)提高学生的编写效率以及积极性。根据研究显示，85%的学生表示发现编译器反馈的错误信息是他们挫折感的来源，92%的学生表示编译器反馈的错误信息是他们进步的障碍。64%的人说发现多个编译错误信息会让他感到困惑。某些错误难以通过编译器的所给出信息而准确定位，这需要我们在给初学者的所看到的错误信息进行增强提示。

2)提高教师教学效率。对于很多初学者来说，经常遇到编译错误、运行错误等问题，所以需要向教师求助，但是由于C语言的错误类型过多，加之部分错误较为隐秘且不常见，这给教师快速定位错误带来困难。对于编译错误的增强提示可以帮助教师跳过阅读冗杂的代码，使教师直接看到代码的错误原因，从而快速准确的定位到出错代码。

## **1.2 背景****知识介绍**

## **1.2.1 编译错误的分类方式**

为了便于讨论编译器的错误信息，参考爱尔兰都柏林大学团队的研究,我们引入了一个分类方法。所有的编译错误信息都属于两种类型中的一个：第一条错误信息或随后的其他错误。第一条错误信息是编译器在编译失败后报告的第一条信息。第一条信息总是与代码中的真正错误相对应。即使是新手也知道，大多数编译错误会产生一些额外的错误信息；我们不知道这些非第一条错误信息的现有术语。我们使用术语 “后续错误”来指代编译器在第一条信息之后报告的所有信息。后续错误可以依次分为两种类型。

**级联式错误**：级联式错误信息是由一个与先前的错误信息相对应的错误引起的虚假信息，并没有提供额外的（有用的）信息。简而言之，这些是编译器提示对于初学者不够人性化的结果。

**真正的后续错误**:这些是由于代码中的真正错误而产生的后续信息。与级联错误信息不同的是，这些信息不是第一条错误的级联式错误，而是代码中的其他错误相对应的第一条错误信息。

## **1.2.2 C语言程序设计中常见的错误类型**

一般错误类型有三种，分别是:语法错误、逻辑错误、系统错误。这在**C语言程序设计**中也是一样。

语法错误是指一份代码中存在有不符合语法规定的语句，如使用没有被定义过的变量，不区分大小写等。

逻辑错误，又称语义错误，指符合C语言的语法规定，却不符合需求逻辑导致错误结果的错误。如‘==’错误打成‘=’号，将判断相等的逻辑变成了赋值语句，这类错误不会通过编译器反馈出来，程序会正常执行只是结果与预期不相符。

系统错误是指程序运行环境中产生的错误，如代码运行所处的软硬件环境中产生错误导致代码无法正常运行。

## **1.3 国外在增强提示方面的研究**

国外对于这方面的研究较为丰富，国外十分重视初学者的代码实践，它们做出了很多理论上的研究。

文章：爱尔兰都柏林大学团队对于编译错误集合进行了梳理，它们发现了多数情况下一个错误对应着多条编译器错误信息，其中一些错误信息是无帮助的，甚至会产生混淆。它们基于这种情况对错误信息进行了归类分析。

文章：在文章的基础上，他们提出了同时处理多条编译器错误信息的方法，入了一个新的分类法，使研究人员和教育工作者有可能更准确地谈论这些编译错误。定义每次编译失败的第一条信息被标记为 "第一条信息"。 理论上，随后的信息可以分为三种类型：“级联信息”（虚假的错误信息，当真正的错误被修复后就会消失）；“真正的第一条信息”（对应于一个明显的错误，在编译中成为第一条信息）；以及 “被埋藏信息”（对应于一个明显的错误，但在编译后未成为第一条信息）。只考虑第一条错误信息，从全局来看是更为有效的。

文章：他们对于增强提示后的效果做出了研究，验证了增强提示的有效性， 实质性的对于调试做出了帮助。

文章、文章：对于java常见的新手错误进行了收集，对常见的错误以及教育者如何利用这些信息做出了研究，为如何将增强提示应用到新手教学中提出了指导意见。

## **1.4 国内在增强提示方面的研究**

文章：从底层阐述了编译错误的诞生，并描述到了一个编译错误诞生的生命周期，这对于对某些相对复杂的编译错误做出个性化响应有所帮助。

如何帮助学生理解和处理编译器反馈的错误，我们可以从C语言程序中计算思维着手，计算思维的本质是抽象和自动化。抽象指的是使用符号系统对问题进行精确而严格的描述;自动化指的是对这些符号系统施加一定操作并按照某种结构自动地执行。

稍早年间，国内也有学者提出了从编译原理的角度对编译错误进行分析,描述了编译器如何识别语法错误，这对于C语言中较复杂的编译错误进行代码层面的溯源有所帮助。

字节跳动的morden.js中集成了对于node代码的编写错误的自动提示以及代码补全,这类似于Visual Studio Code上面向各种语言的代码补全插件，通过对于抽象语法树的分析发现编译错误，并通过自定义逻辑对于这些错误进行提示与补全。

在智能化软件开发中,代码补全实现了由编译器根据上下文中现有代码建议下一个可能的代码Token,例如方法调用或对象字段。近年来随着深度学习的应用,我们国内在该方向也涌现出大量科研成果,推进了软件智能化发展。

## **1.5 论文组织结构**

**论文共分为六章。**

**第一章是绪论部分。在这一章主要介绍了本此工作的研究背景，研究作用和意义。介绍了相关的背景知识，国内外在增强提示方面现有的研究成果。**

**第二章是相关技术介绍。这一章主要介绍了本次工作中用到的几个关键技术，**介绍了各项技术的特性及在本次工作中的用途**。**

**第三章是编译错误研究。这一章对于从数据中收集的编译错误进行了分析讨论，提出了对于错误的归类方法，并在此基础上给出了编写增强提示的思路。**

**第四章是本次工作的需求分析和总体设计。这一章从相关课程的角度出发，分析了本次工作的需求，并一一给出了增强提示服务所需要的能力，提出了服务的运行环境。**

**第五章是增强提示方法的详细设计。这一章主要介绍了增强提示服务在开发过程中的关键设计与具体实现，阐述了数据收集的工作流程，详细说明了客户端和服务器端的设计思路及代码实现。**

**第六章是总结与展望。这一章主要总结了设计与开发工作，提出了工作的局限性，并阐述了完善增强提示方法以后所需要做的工作。**

# 第二章 相关技术介绍

## **2.1 Vue.js**

Vue是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库内容只关注视图层，不仅易于上手，还便于整合第三方库。另一方面，当与现代化的工具链结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。在本次工作中，我们使用Vue来搭建增强提示演示的前端单页应用。

## 2.2 koa

Koa基于node.js ，是一个新的 web 框架，它与以往的node服务端框架不同，通过利用 async 函数，抛弃了使用回调函数的传统方法，可以避免‘回调地狱’，从而有效的增强了错误处理。 Koa 本身没有捆绑任何中间件。在本次工作中，我们使用Koa作为web服务端框架，来提供增强提示服务。

## **2.2.1 中间件**

在广义的定义中，中间件是介于应用系统和系统软件之间的一类软件，为了能够达到资源共享、功能共享的目的，它基于系统软件所提供的基础服务，将网络上应用系统的各个部分或不同的应用的逻辑连接在一起。

在node.js和Express、Koa等web框架中，中间件的本质为一个回调函数，参数包含请求对象、响应对象和执行下一个中间件的函数，在这些中间件函数中，我们可以执行业务逻辑代码，修改请求和响应对象、返回响应数据等操作。Koa实现逻辑处理的方式正是通过中间件的拓展，通过添加不同的自定义中间件来实现不同的需求，最终组成一个Koa应用。

## **2.2.2 洋葱模型**

在 Koa 中，中间件被 ‘next’方法分成了两部分。‘next’方法之前的部分会先执行，之后的部分会在后续中间件逻辑全部执行完成后再执行。

在洋葱模型中，每一层相当于一个中间件，用于处理特定的功能，比如错误处理、响应头编辑等等。其处理顺序先是 next() 前请求（Request，从外层到内层）然后执行 next() 函数，最后是 next() 后响应（Response，从内层到外层），也就是说每一个中间件被‘next’方法划分成了两部分，都有两次处理逻辑的时机。其模型如图2.1所示。

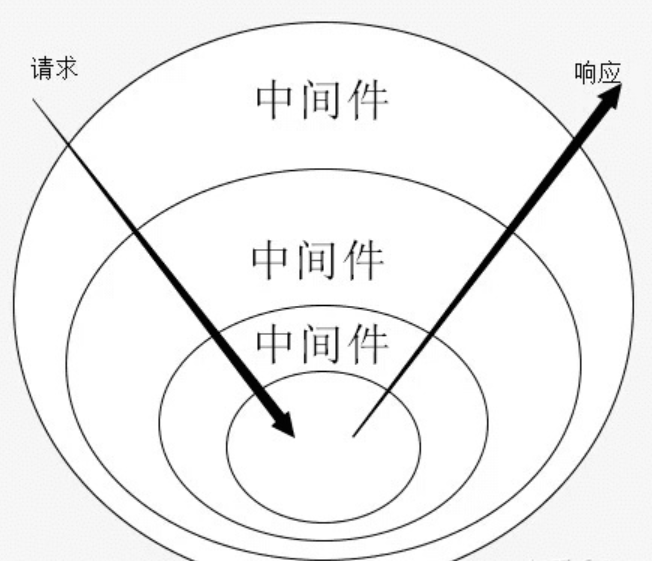


图2.1 洋葱圈模型

## 2.3 Navicat

Navicat for MySQL 是管理和开发 MySQL的理想解决方案。它是一个好用的图形化数据库管理软件，有许多便捷的可视化工具来帮助我们操作数据库。在本次工作中我们使用它来辅助进行数据库操作、提取学生的编译错误信息。

## 2.4 Element Plus

Element UI是一个前端UI库，它不依赖于Vue。但是却是当前和Vue配合较好且最为广泛使用的UI组件库之一，Element Plus是在element UI之上，基于Vue 3，面向设计师和开发者的组件库。在本次工作中我们使用它来辅助前端页面的开发。

## 2.5 Axios

Axios 是一个基于 promise 的 HTTP 库，浏览器和 node.js 是它常见的使用场景，它拥有以下特性：从浏览器中创建 XMLHttpRequests、从 node.js 创建 http 请求、支持 Promise API、拦截请求和响应、支持转换请求数据和响应数据、支持取消请求、自动转换 JSON 数据、客户端支持防御 XSRF。在本次工作中我们使用它来进行客户端与服务端之间的HTTP通信。

## 2.6 本章小结

本章介绍了面向C语言的编译错误增强提示方法研究与实现中所用到的几个关键技术，并依次介绍了各项技术的特性及在本次工作中的用途。本章为工作的进行奠定了坚实的基础。

# 第三章 编译错误研究

在CSUOJ作为数据集的背景下，我们对于14,302份来自21届学生的编译错误代码进行了分析，统计了编译器响应的编译错误，进行了去重、归类、统计所占比等处理。在统计的基础上进行了分析，为如何下一步进行增强提示提供了方案。

## **3.1 统计概览**

我们14,302份代码中一共统计出31,239条编译错误信息，首条错误信息占比为45.78%，在考虑所有编译错误信息的情况下，最频繁出现的错误(占比大于1%)如表3.1所示。

表3.1 常见编译错误(所有错误)

|  |  |  |
| --- | --- | --- |
| error | times | percent |
| (expected (.\*) before (.\*)|(expected (.\*) at end of (.\*))|(expected (.\*) after (.\*))) | 9268 | 29.67% |
| (undeclared)|(was not declared)|(has not been declared) | 5397 | 17.28% |
| stray ‘(.+?)’ in program | 4862 | 15.56% |
| ld returned 1 exit status | 1724 | 5.52% |
| 没有那个文件或目录 | 1669 | 5.34% |
| unknown type name ‘(.+?)’ | 1391 | 4.45% |
| ‘(.+?)’ has no member named ‘(.+?)’ | 871 | 2.79% |
| variable-sized object may not be initialized | 649 | 2.08% |
| request for member ‘(.+?)’ in something not a structure or union | 436 | 1.40% |
| lvalue required as left operand of assignment | 421 | 1.35% |
| ‘else’ without a previous ‘if’ | 345 | 1.10% |
| variably modified ‘(.+?)’ at file scope | 330 | 1.06% |

若我们只考虑首条错误信息，最频繁出现的错误信息会有所差异。可以观察到如‘ld returned 1 exit status’，它永远作为首条错误信息出现;如‘stray ‘(.+?)’ in program’的占比则大大降低，这代表它常常作为级联错误而出现。只考虑首条错误信息的表格如表3.2所示。

表3.2 常见编译错误(仅首条错误)

|  |  |  |
| --- | --- | --- |
| error | times | percent |
| (expected (.\*) before (.\*)|(expected (.\*) at end of (.\*))|(expected (.\*) after (.\*))) | 4517 | 31.67% |
| (undeclared)|(was not declared)|(has not been declared) | 2178 | 15.27% |
| ld returned 1 exit status | 1724 | 12.09% |
| 没有那个文件或目录 | 1660 | 11.64% |
| stray ‘(.+?)’ in program | 907 | 6.36% |
| unknown type name ‘(.+?)’ | 629 | 4.41% |
| variable-sized object may not be initialized | 497 | 3.49% |
| lvalue required as left operand of assignment | 272 | 1.91% |
| variably modified ‘(.+?)’ at file scope | 154 | 1.08% |
| ‘else’ without a previous ‘if’ | 147 | 1.03% |

## **3.2 错误分析**

在整理这些常见的编译错误后我们认为，这些错误可以简单归类为两类:一是简单的语法错误，即可以通过编译器反馈的信息来直接定位到代码中的错误，二是复杂晦涩的错误，这类错误往往是由于编译器的一条信息可能对应着多种错误场景，首条错误信息并没有直接指出错误，无法给予学生直接的帮助。我们在接下来把这两种错误简称为简单错误与复杂错误。

## **3.2.1 简单错误**

简单错误往往可以通过编译器给出的首条信息直接得知错误的原因，如统计中最常出现的错误‘(expected (.\*) before (.\*)|(expected (.\*) at end of (.\*))|(expected (.\*) after (.\*)))’，它代表了代码中符号缺失或不匹配这一类错误，这类错误很明确的指出了在某个符号前或者后，缺少了哪个对应的符号。数据库中solution\_id=1903的代码就是这类错误的代表，其代码如图3.1所示。

|  |
| --- |
| #include <stdio.h>  int main(void)  {  int a, b;    while(scanf("%d %d",&a,&b)  {  printf("%d**\n**", a + b);  }  return 0;  } |

图3.1 solution\_id=1903的代码

此时编译器的反馈信息如图3.2所示:

|  |
| --- |
| Main.c: In function ‘main’:  Main.c:7:3: error: expected ‘)’ before ‘{’ token  Main.c:11:2: error: expected expression before ‘}’ token |

图3.2 solution\_id=1903代码的编译器反馈信息

可以很清晰的看到，首条错误信息告诉我们错误在第7行，在符号‘{’前期望有一个符号‘(’。而第二条错误信息就是很明显的级联错误信息，由于缺少右括号导致代码段的括号不匹配，while循环代码段不构成一个合法的表达式。

这类错误尽管新手常犯，但我们认为它们并不是学生难以自我解决的问题，且不会占用教师过多的时间，这类错误往往只和首条错误信息相关联，我们对于首条错误信息进行增强提示即可。

## **3.2.1 复杂错误**

复杂错误是我们需要重点讨论的类型。它可以被细分为两类：

一是编译器反馈的信息不能准确反应错误:如‘stray ‘(.+?)’ in program’，它表示代码中有不可识别的字符。有经验的程序员会知道这往往是由于代码中插入了全角字符，但是仅凭编译器反馈的信息却无法确定到底是哪个字符的问题，数据库中solution\_id=2202的代码是这类问题的代表，其代码如图3.3所示:

|  |
| --- |
| #include <stdio.h>  int main ()  {  printf（"hello world"）;  return 0;  } |

图3.3 solution\_id=2202的代码

此时编译器的反馈信息如图3.4所示:

|  |
| --- |
| Main.c: In function ‘main’:  Main.c:4:11: error: stray ‘\357’ in program  printf（"hello world"）;  Main.c:4:12: error: stray ‘\274’ in program  printf（"hello world"）;  Main.c:4:13: error: stray ‘\210’ in program  printf（"hello world"）;  Main.c:4:14: error: expected ‘;’ before string constant  printf（"hello world"）;  Main.c:4:27: error: stray ‘\357’ in program  printf（"hello world"）;  Main.c:4:28: error: stray ‘\274’ in program  printf（"hello world"）;  Main.c:4:29: error: stray ‘\211’ in program  printf（"hello world"）; |

图3.4 solution\_id=2202代码的编译器反馈信息

我们可以看出，代码的第四行是中文符号的左右括号，这是违反语法规定的。但是编译器反馈的信息中却并不是提示的中文括号识别失败。

参考文档可知，这是不同的编码方式而导致的提示不明确，比如一份代码中有一个字符•,它的Unicode值为2022(十六进制)，但当编译器将其读取为ASCII码时，会得到\342\200\242(3个八进制值)，从而导致编译器反馈的信息是不符合预期的。

面对这种错误，新手是很难以自己解决的，他们或许暂时解决了当前代码的编译错误，却没能真正理解到错误发生的根本原因。

二是编译器反馈的某条信息可能对应着多种可能的错误场景: 如‘ld returned 1 exit status’，这个错误的本意是C语言编译的构建过程中的链接步骤遇到了一些错误，通常退出状态0表示成功，退出状态>0表示错误。但是这个本意太模糊了，对于定位代码中的错误毫无帮助。实际上产生该错误的原因可能是库函数拼写错误。如：printf，scanf等；定义的函数名在调用时，函数名拼写错误；在本地运行时，同一份代码还在运行中，再次执行了编译，由于文件正在被调用，程序没法执行编译导致报错。

这类错误最为棘手，需要新手积累经验才可以有效的解决，且会遇到这类问题会占用教师大量的时间，尽管编译器反馈信息相似，但是解决方案却难以复用。

## 3.3 增强提示方案

结合上文的统计概览与错误分析来看，学生作为新手所犯的编译错误有相似度较高且逻辑较为简单的特点。在统计中出现次数大于1%的编译错误中，我们定义的复杂错误仅有两条。

## 3.3.1 简单错误的增强提示

对于简单错误，编译器反馈的信息能够准确的定位到错误产生在代码的哪一行，是什么错误逻辑，甚至编译器还可以给出修改建议。我们的增强提示服务大体思路上是对编译器的反馈信息做出翻译，并给出错误代码修改示例即可辅助学生解决这类问题。以solution\_id=2400的代码为例，它的代码部分如图3.5所示:

|  |
| --- |
| #include <stdio.h>  ing main()  {  printf("Hello World**\n**");  return 0;  } |

图3.5 solution\_id=2400的代码

此时编译器的反馈信息如图3.6所示:

|  |
| --- |
| Main.c:2:1: error: unknown type name ‘ing’; did you mean ‘int’? |

图3.6 solution\_id=2400代码的编译器反馈信息

可以很清晰的看到编译器提示在第二行，‘ing’是一个未知的类型名，并且提示用户的本意是不是使用int这个类型。故我们的增强提示对于编译器的反馈信息进行翻译即可。

## 3.3.1 复杂错误的增强提示

对于常见的复杂错误，我们采用每种错误都通过服务端自定义逻辑的方式进行单独处理，这类错误直接关联性不强且它们的增强提示难以复用，我们在编写增强提示的时候进行分类讨论、按例分析。如这类错误‘stray ‘(.+?)’ in program’，我们给出的增强提示为:“在代码中打入了全角字符，请检查是不是某个字符没用英文输入法。请检查报错行是否包含了全角的字符。\\\*\*\*并不一定代表了一个具体的字符，它可能是一个全角符号的Unicode值转为了多段ASCII码表示”。

## 3.4 本章小结

本章列举了从CSUOJ的数据库中收集的编译错误。并对于这些错误进行了分析讨论，将成果作为教学实践的参考。提出了对于错误的归类方法，并在此基础上给出了编写增强提示的思路。

# **第四章 需求分析与总体设计**

在C语言教学环节中, 调试错误的能力往往被忽略, 很多教师在辅导的过程中, 认为能写出没有错误的程序就是好学生。诚然, 能写出没有语法错误和逻辑错误程序的学生肯定是基础知识比较扎实, 语法掌握比较好的学生。但在实际应用中, 程序错误是普遍出现的现象。为了帮助学生更好的通过编译器反馈信息从而对自己的代码进行调试，我们给出了一个面向C语言的编译错误增强提示方法，用于满足未来的教学实践需要。

## 4.1 服务的需求分析以及细分能力

## 4.1.1 需求分析

《计算机程序设计基础（C语言）》是我院学生在大一年级的一门专业必修课程，该课程作为程序设计的入门课程其重要性不言而喻。一般来说随着课程的进行，代码实践是不可或缺的环节。故在需求的开发过程前，应充分从该课程的特点、学生实践环境、教师教学的角度来进行分析。

从课程的特点来看，《计算机程序设计基础（C语言）》理论性较强，但同时也对学生的实践能力提出了较高的要求，学生在课程学习后进行编码是掌握这门课程的一个重要环节。理论的学习不可能脱离代码的实践，这意味着学生在本课程的学时中会进行大量的编码，他们大多会经历模仿抄写教科书上的代码再到独立完成简单问题的过程。这个过程里学生代码中产生的编译错误往往是类似的，甚至对应到某道练习题上往往只能收集到几个不同编译错误。故我们可以先将数据库中所有的编译错误信息提取并去重，根据文献以及我们的开发经验，编写常见编译错误信息所对应的增强提示。

从学生实践的环境中来看，我们并没有要求学生在相同的环境下编写代码，这意味着发生系统错误是不可控的。但是我们可以确定的是最终我们是通过Online Judge来判断学生的代码是否正确。所有学生的代码会通过同一版本的GCC编译器进行编译。既然流程会收口于Online Judge上，未来我们可以考虑将增强提示服务与Online Judge集成，这样当学生提交的代码发生Compile Error时，我们在Online Judge不仅显示编译器的报错信息，还同时显示在后台部署好的增强提示信息，从而帮助学生的实践。

从教师教学的角度来看，我们之前提到了学生作为新手所产生的错误往往具有一致性，但是我们并不能保证每位学生都能通过增强提示服务所反馈的信息来改正自己的编译错误。加之有复杂错误的存在，个别的复杂错误下增强提示服务可能给出的并不是一个对于当前错误合格的解释。所以不可避免的，仍会有学生找到教师来求取，为了节省教师的时间，我们对于高频错误不仅需要提供增强提示，还需要提供高频错误的解决方案示例，示例包含错误代码示范和修改方案。

## 4.1.2 服务的细分能力

为了增强提示能成功的在Online Judge上相应，我们的服务需要以下这些能力：

1. 从Online Judge的数据库中获取所有的编译错误信息，并进行同类错误合并、相同错误去重计数的能力。制作一个新的表单，以每个错误的hash值作为唯一键，新建一张错误-增强提示的表单。
2. 从编译器反馈信息中提取首条错误以及级联错误的能力。对于Online Judge上新的犯有编译错误的代码所对应的编译器反馈中提取错误，用于匹配服务端所储存的错误-增强提示的表单，获取对应的增强提示。
3. 编写增强提示的能力。对于常见的错误编写增强提示，提供入口来修改或是补充增强提示。
4. 前端展示的能力。提供一个增强提示演示的单页应用，用于展示服务的具体效果，同时也可以用于课程中的教学演示或是作为学生学习的小工具。未来可将增强提示前端并入Online Judge中，让学生提交代码编译失败后立即获得帮助。

在拥有了上述几项能力后，服务可以正常的响应，满足教学以及辅助学生实践的需要。

## 4.2 服务的体系结构

服务的上游与数据库相关联收集Online Judge中的编译错误信息，下游接收单次编译后编译器的反馈信息并给出对应的增强提示。在给出服务的接口文档后，此服务可作为插件的形式接入各种场景。如增强提示演示的单页应用、Online Judge代码提交的响应页面。我们只关注增强提示服务本身，使其有很强的灵活性以及可拓展性。

## 4.3 服务的运行环境

服务的核心本身只包含了服务端。但是由于还涉及了数据收集和增强提示的前端单页应用展示，故从数据收集，客户端，服务端三个部分来进行介绍。

## 4.3.1 数据收集运行环境

数据收集是在Windows环境下编写的node.js脚本进行的数据处理。运行环境与服务端的运行环境类似。

## 4.3.2 服务端运行环境

服务采用node.js进行编写，需要服务所运行的系统中有node环境并版本高于v14.16.1。开发过程中利用了typescript和import等新的语法特性，但是最终通过babel进行了转义，在必要的情况下产物最终可以为.js文件，便于服务的部署。

## 4.3.3 客户端运行环境

这里的客户端指的是增强提示演示界面，客户端采用Vue3来进行开发，最终通过Vite来进行打包，最终的产物是可在浏览器中运行的JavaScript文件。

Vite利用的是ES Module，现代浏览器大多支持，但是由于使用了Vue3，它不再支持IE11，故最终产物可以在除了IE浏览器的主流浏览器上运行。

## 4.4 本章小结

本章首先从《计算机程序设计基础（C语言）》这一门课程的角度出发分析了本次服务的需求，并一一给出了服务所需要的能力，其次介绍了服务的体系架构，以及服务的运行环境。

# 第五章 详细设计与实现

为了实现本次服务，大体上分为了数据收集、服务端搭建、客户端搭建这三步。通过收集的数据来编写增强提示，将编写好的增强提示组合上响应编译器反馈信息的逻辑构成服务端的增强提示响应服务，最后将增强提示显示在作为演示页面的客户端上。

## 5.1 数据收集详细设计及实现

## 5.1.1 获取所有的编译器反馈信息

在CSUOJ的数据库架构中，一份代码的编译信息、编译器响应状态与源码并不在同一张表中，它们的数据库结构如图5.1所示。

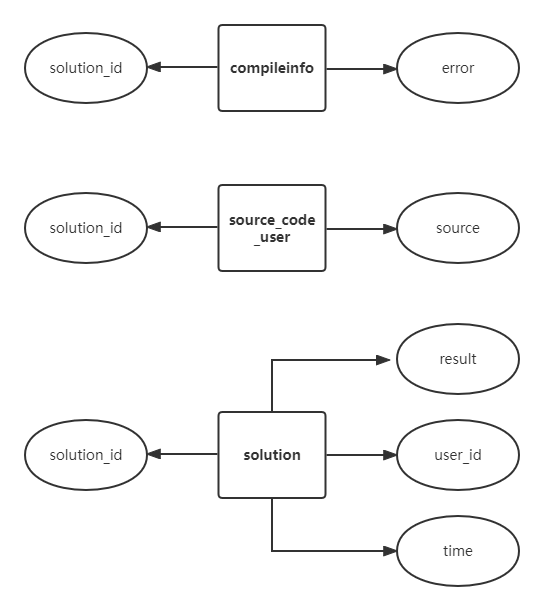


图5.1 CSUOJ数据库表单实体图

表‘compileinfo’中存储了编译器的反馈信息、表‘source\_code\_user’中存储了每份提交的源代码，表‘solution’中存储了每次提交的运行结果‘result’、用户ID‘user\_id’,代码运行时间‘time’等程序运行的相关信息。

故我们需要通过sql语句查询，将这几张表单的数据进行关联。观察这几张表可知，它们的信息可以通过唯一键‘solution\_id’关联起来，从而提取出运行结果为编译器反馈信息的编译器反馈信息。如图5.2所示:

|  |
| --- |
| **SELECT**  error,  user\_id,  nick,  source\_code\_user.solution\_id,  source\_code\_user.source  **FROM**  compileinfo a,  solution b,  source\_code\_user  **WHERE**  a.solution\_id = b.solution\_id  **AND** b.result = 11  **AND** a.solution\_id=source\_code\_user.solution\_id |

图5.2 提取所有编译器反馈信息

最终提取的部分表单如图5.3所示。

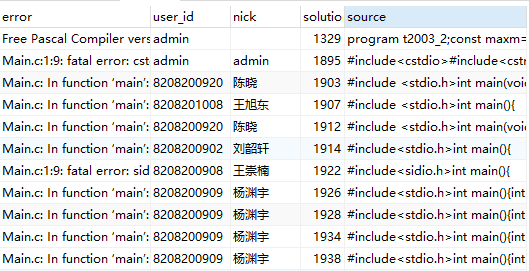


图5.3 获取编译器反馈信息

接来下利用Navicat进行数据提取的辅助，将所有编译器反馈信息通过内置工具以json的格式提取出来，以便于编写node.js脚本对于所有编译器反馈信息进行统计处理。表单数据转换为json的操作如图5.4所示。

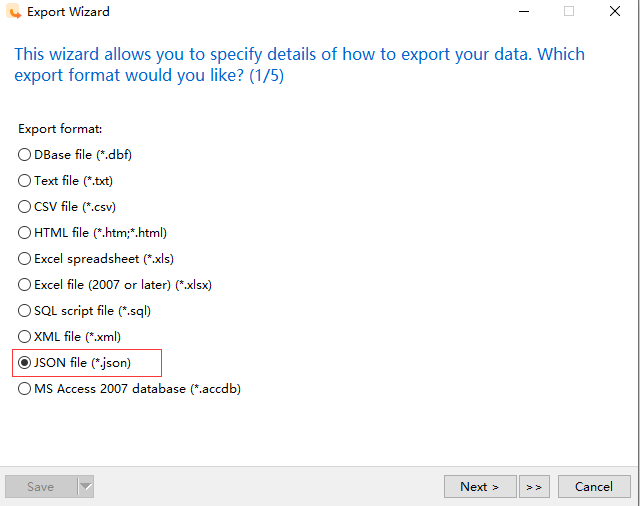


图5.4 所有编译器反馈信息转换为json格式

最终将所有编译器反馈信息储存在文件‘all\_error\_message.json’中。

### 5.1.2 编译错误统计处理

在收集完所有编译器反馈信息后，需要从这些信息中提取所有的编译错误信息。我们采用node.js编写脚本处理，通过正则匹配的方式，从每条编译器反馈信息中提取编译错误信息。如图5.5所示。

|  |
| --- |
| const fs = require("fs");  let path = require("path");  import { highFrequencyErrors } from "./highFrequencyErrors.js";  const filepath = "../resource/all\_error\_message.json";  let fileStr = fs.readFileSync(path.resolve(\_\_dirname, filepath), "utf-8");    let jsonstr = JSON.parse(fileStr);  let allError = [],  errors = [],  count\_err = 0;  const reg = /error:(.+?)\n/g; //获取error:xxx到\n间的报错信息    let getPercent = (x) => {  return `${((x / errorSize) \* 100).toFixed(2)}%`;  };    jsonstr.data.forEach((item) => {  (item.error.match(reg) || []).forEach((str,key) => {  //if (key>0) return ;//统计首条错误信息  allError.push(  str  .toString()  .replace(/(error: )/g, "")  .replace(/(\n)/g, "")  );  });  }); |

图5.5 获取所有编译错误

正则表达式‘/error:(.+?)\n/’表示获取error:xxx到\n之间的报错信息，利用Reg对象原型链上的‘match’方法，对每条编译器反馈信息进行匹配，就可以获取所有的编译错误信息，之后将所有编译错误信息存入‘allError’这个变量中。

接下来需要对所有的编译错误进行去重处理，并进行计数。需要注意的是，所有编译错误中有很多的错误是属于某种相同类型的，它们的内容虽然略有差异，但是本质上是同一种错误。我们通过现有的经验，收集了若干常见的错误类型，将可以匹配它们的正则表达式存在了‘highFrequencyErrors.js’这一表单中，如图5.6所示。

|  |
| --- |
| export const highFrequencyErrors = [  "stray ‘(.+?)’ in program",  "unknown type name ‘(.+?)’",  "(undeclared)|(was not declared)|(has not been declared)",  "(expected (.\*) before (.\*)|(expected (.\*) at end of (.\*))|(expected (.\*) after (.\*)))",  "too few arguments to function ‘(.+?)’",  "too many arguments to function ‘(.+?)’",  "expected declaration specifiers",  "‘(.+?)’ does not name a type",  "‘(.+?)’ has no member named ‘(.+?)’",  "没有那个文件或目录",  "variably modified ‘(.+?)’ at file scope",  "redeclaration of ‘(.+?)’",  "redefinition of ‘(.+?)’",  "parameter ‘(.+?)’ is initialized",  "label ‘(.+?)’ used but not defined",  "conflicting types for ‘(.+?)’",  "request for member ‘(.+?)’ in something not a structure or union",  "array size missing in ‘(.+?)’",  "no matching function for call to",  "note:",  ]; //高频类似错误合并集 |

图5.6 所有高频错误的正则表达式

有了高频错误的增则匹配表单后，我们对于普通从错误进行去重，对于属于高频错误表单中的编译错误进行合并，最后按照单个错误在所有编译错误中所占的比例从高到底进行排序，给出整理后的所有编译错误统计信息。核心代码部分如图5.7所示。

|  |
| --- |
| result.data.forEach((item) => {  if (  !highFrequencyErrors.some((reg) => {  if (item.error.match(reg)) {  if (!errorExiested.get(item.error)) {  errorExiested.set(item.error, true);  if (!highFrequencyErrorsArr.get(reg))  highFrequencyErrorsArr.set(reg, []);  highFrequencyErrorsArr.get(reg).push({  error: item.error,  times: item.times,  percent: getPercent(item.times),  uid: item.uid,  });  highFrequencyErrorsMap.set(  reg,  (highFrequencyErrorsMap.get(reg) || 0) + item.times  );  }  return true;  }  })  ) {  resultWithDeduplication.data.push(item);  }  });  highFrequencyErrors.forEach((item) => {  resultWithDeduplication.data.push({  error: item,  times: highFrequencyErrorsMap.get(item),  percent: getPercent(highFrequencyErrorsMap.get(item)),  errors: highFrequencyErrorsArr.get(item),  });  });  resultWithDeduplication.data.sort((a, b) => b.times - a.times); |

图5.7 统计所有编译错误

经过统计，生产了两份数据‘errorsWithtips.json’和‘errorsWithDeduplication.json’，前者是没有对于同类错误进行合并，只进行了去重的所有编译错误统计文件，后者是进行了同类错误合并的所有编译错误统计文件。第一个文件用于下一步增强提示的编写，一个具体的错误对应一个增强提示，此json文件的数据结构如图5.8所示。

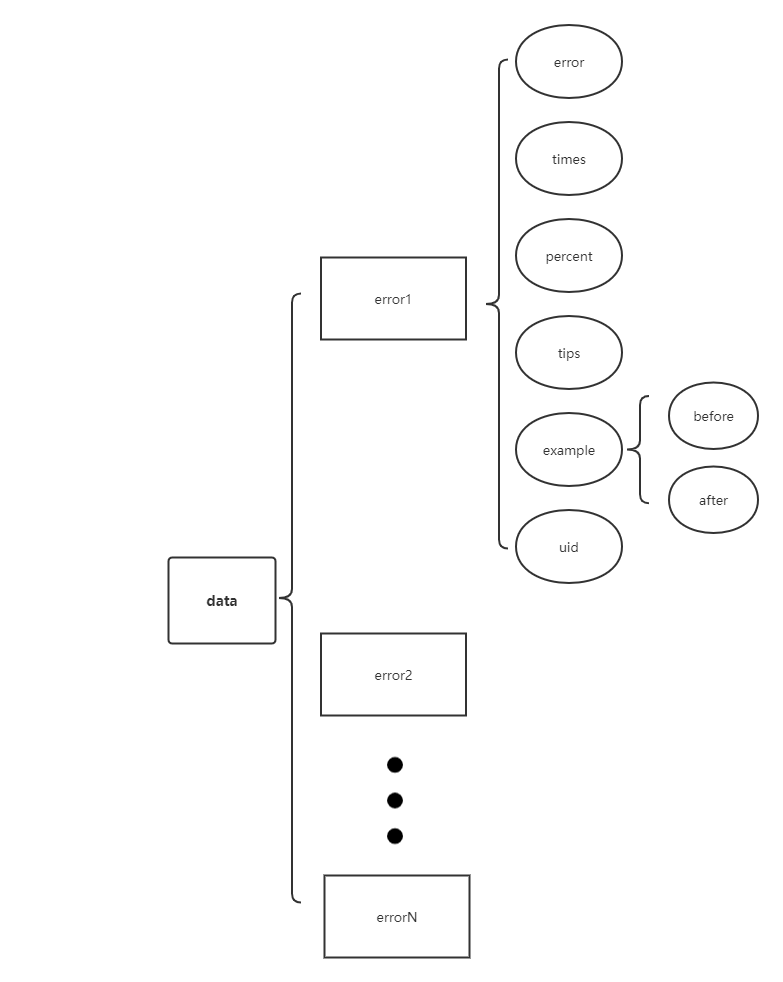


图5.8 errorsWithtips.json文件数据结构

此json文件中依次存储了所有内容不一样的编译错误，每个错误包含的属性含义如下:

Error:错误内容，编译器反馈信息的错误内容截取。

Times:当前错误在数据集中总计出现的次数。

Percent:当前错误在数据集中总计出现的次数占所有错误的比例。

Tips:通过我们的编码经验编写的增强提示。

Example:高频错误的解决方案示例,before表示错误代码示范after表示修改方案。

Uid:当前错误的唯一标识符，按照频率从自然数一开始编号，可以作为增强提示储存于数据库中的主键。

第二个文件‘errorsWithDeduplication.json’用于数据的统计分析还有研究结果的展示，它可作为一个教学参考，展示了学生们在实践中常犯的错误是什么，并可以针对性的做出辅导。

### 5.1.3 增强提示编写

在收集完所有编译错误之后，对应出现频率大于0.1%的错误进行编译错误的增强提示编写，这样可以覆盖数据集中93%的错误场景。

对于简单错误而言，直接按照经验进行增强提示。对于复杂错误，每种错误都通过服务端自定义逻辑的方式进行单独处理。特别的，对于常见的高频错误，为了节约工作量，可以编写脚本，一次性完成同类错误的增强提示。转换函数集合如图5.9所示。

|  |
| --- |
| export const func = [  (item) => {  if (item.error.match("stray ‘(.+?)’ in program")) {  item.tips =  "在代码中打入了全角字符，检查一下是不是某个字符没用英文输入法";  }  },  (item) => {  if (  item.error.match(  "(undeclared)|(was not declared)|(has not been declared)"  ) &&  item.tips.length === 0  ) {  item.tips = "该错误表示报错变量未定义";  }  },  (item) => {  if (  item.error.match(  "(expected (.\*) before (.\*)|(expected (.\*) at end of (.\*))|(expected (.\*) after (.\*)))"  ) &&  item.tips.length === 0  ) {  item.tips =  "请检查报错行表达式是否缺少了对应的标识符，代码段括号是否闭合";  }  },  (item) => {  if (item.error.match("没有那个文件或目录") && item.tips.length === 0) {  let reg = /(.+?):/g;  let name = item.error.match(reg)[0].replace(/:/, "");  item.tips = `检查头文件是否引用了#include<${name}>,\r\n该错误表示编译器没找到这个库，请检查是否选择了正确的语言及正确的编译器版本进行代码提交。`;  }  },  ]; |

图5.9 转换函数集合

完成编写后得到了包含增强提示的编译错误数据，存储于‘errorsWithtips.json’文件中。

## 5.2 服务端详细设计与实现

增强提示服务用于接收单次编译后编译器的反馈信息以及部分可选信息后给出对应的增强提示，用于辅助教学和学生的编码实践。服务采用node.js来编写。用于展示的增强提示演示单页应用是一个前后端分离项目，通过Koa中间件的方式处理请求、监听路由，以其内建的对于http模块封装与客户端进行http通信。服务的接口约定如表5.1、5.2所示。

表5.1 接口输入字段约定

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 是否可选 | 备注 |
| error | string | 必选 | 编译器反馈信息 |
| source\_code | string | 可选 | 程序源码 |
| time | number | 可选 | 程序运行时间(ms) |
| result | number | 可选 | 程序运行结果的状态码(与CSUOJ数据库一致) |

表5.2 接口输出字段约定

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 是否可选 | 备注 |
| tips | string | 必选 | 增强提示信息 |
| lines | number | 可选 | 错误所在行 |
| example\_before | string | 可选 | 错误示例代码 |
| example\_after | string | 可选 | 修改错误示例 |

## 5.2.1 http通信模块

通过Koa的实例方法‘listen’监听指定端口，当请求指定路径/api时表示访问增强提示服务，在‘use’方法中获取请求实例ctx，设置http请求的响应头的‘Access-Control-Allow-Origin’、‘Access-Control-Allow-Methods’、‘Access-Control-Allow-Headers’字段来解决浏览的CORS策略所导致的跨域问题，并调用‘getTips’方法来获取请求所对应的增强提示，最后响应。相关代码如图5.10所示。

|  |
| --- |
| import Koa from "koa";  const app = new Koa();  const koaBody = require("koa-body");  import getTips from './getTips'    app.use(koaBody());    app.use(async (ctx) => {  if (ctx.request.path === "/api") {  console.log("do");  ctx.set("Access-Control-Allow-Origin", "\*");  ctx.set("Access-Control-Allow-Methods", "OPTIONS, GET, PUT , POST");  ctx.set("Access-Control-Allow-Headers", "\*");  ctx.response.type = "json";  ctx.response.body = {  data: {  tips: getTips(ctx.request.body.compileErrMessage),  },  };  }  });    app.listen(7777);  console.log("app started at port 7777..."); |

图5.10 http通信模块

### 5.2.2 获取增强提示

获取增强提示的代码集成在‘getTips’方法中，大致步骤为首先从http请求体中获取编译器反馈信息，从编译器反馈信息中提取首条错误信息和级联错误信息。将首条错误信息与‘errorsWithtips.json’表单中的每条错误进行匹配，如果是简单错误，直接返回表中该错误对应的tips属性对应的数据；若是复杂错误、且有自定义逻辑的话，结合报文中的可选提交信息，走自定义逻辑来处理复杂错误。相关代码如图5.11所示。

|  |
| --- |
| const reg = /error:(.+?)\n/g; //获取error:xxx到\n间的报错信息  const filepath = "./enhanceTips.json";  let fileStr = fs.readFileSync(path.resolve(\_\_dirname, filepath), "utf-8");  let jsonstr = JSON.parse(fileStr);  const getErrorList = (msg) => {  if (!msg) return "";  let allError = [];  (String(msg).match(reg) || []).forEach((str) => {  allError.push(  str  .toString()  .replace(/(error: )/g, "")  .replace(/(\n)/g, "")  );  });    return allError;  };  //简单错误处理逻辑  const defaultGetTips = (errorList) => {  const firstError = errorList[0];  console.log("firstError", firstError);  if (!Array.isArray(errorList)) return "错误解析失败";  let tip = undefined;  jsonstr.data.forEach((item) => {  if (item.error == firstError) {  console.log("get!", item.tips);  tip = item.tips;  }  });  return tip;  };  export default (msg) => {  const errorList = getErrorList(msg);  let tip = defaultGetTips(errorList);  return tip || "该错误未收集";  }; |

图5.11 获取增强提示

### 5.2.3 服务复杂度分析

考虑整个服务流程，服务的时间复杂度瓶颈处在获取增强提示函数中的将首条错误信息与‘errorsWithtips.json’表单中的每条错误进行匹配这一步。采用枚举的方法进行匹配，单词查询的时间复杂度为‘errorsWithtips.json’表单中所有编译错误字符长度相加。出现频率大于0.2%的错误一共有84条，字符总数为3447，时间复杂度可以接受。经测试在不考虑网络通信的情况下，即服务器搭建在本地，单次请求的响应时间稳定小于100ms，服务响应迅速，可以满足生产环境的要求。

考虑可拓展性，当错误信息的字符总数增多到十万这个数量级的场景下，会出现响应时间大于1000ms的情况。此时可以对编译错误信息进行哈希，将hash值、增强提示这一组键值对存入哈希表中，每次请求所提取的错误也进行hash取值，从哈希表中查询增强提示信息，可将时间复杂度优化到对字符总数取对数。

### 5.3 客户端详细设计及实现

为了展示服务的能力以及教学参考的需要，我们提供了增强提示服务功能展示的前端单页应用，主要功能是根据编译器的反馈信息，来提供对应的增强提示。

前端界面运用Vue进行开发，作为前后端分离的前端项目，使用Axios来进行http通信，布局与页面组件使用Element Plus协助开发。最终效果如图5.12所示。



图5.12 增强提示演示

### 5.3.1客户端界面

页面的布局使用Vue提供的<template>组件进行布局，使用Element Plus中的输入输出框，布局方式为flex布局。利用Vue的<style scoped>标签保证前端页面的CSS样式不会具有全局污染性质。相关代码如图5.13所示。

|  |
| --- |
| <template>  <div class="hello">  <h1>{{ msg }}</h1>  <el-input  v-model="compileErrMessage"  placeholder="Please input"  type="textarea"  :autosize="inputSize"  />  <el-card class="box-card tip-card">  <span class="pre-wrap">  {{ tips }}  </span>  </el-card>  </div>  </template>  <style scoped lang="less">  h3 {  margin: 40px 0 0;  }  ul {  list-style-type: none;  padding: 0;  }  li {  display: inline-block;  margin: 0 10px;  }  a {  color: #42b983;  }  </style> |

图5.13 客户端布局

为了输入C语言源码和展示错误修改建议，我们利用Ace.js搭建了C语言的代码编辑框，每一个编辑框是一个‘ace-builds’的实例。在代码编辑框中我们可以完成输入代码并监听代码框中数据的变化以进行上传、高亮代码框中的某一行以提示错误所在的位置、在代码框中写入代码以展示修改建议的操作。

### 5.3.2客户端逻辑

为了响应输入框中的编译器反馈信息，将编译错误输入框中的数据与变量‘compileErrMessage’进行双向绑定，通过Vue组件中的watch方法，响应式监听‘compileErrMessage’变量，每当输入框中的信息发生变化，便通过监听发送输入框中内容到服务端，通过Axios进行http通信，从而获取对应的增强提示，将得到的增强提示信息与展示框的数据变量‘tips’进行双向绑定，从而达到了前端页面响应式的效果。相关代码如图5.14所示。

|  |
| --- |
| <script lang="ts">  export default defineComponent({  props: ["msg"],  data(): homeData {  return {  compileErrMessage: "",  tips: "",  };  },  watch: {  compileErrMessage(message: string) {  console.log("compileErrMessage:", message);  axios({  method: "post",  url: "http://localhost:7777/api",  data: {  compileErrMessage: message,  },  })  .then((response) => {  console.log("response", response);  this.tips = response.data.data.tips;  })  .catch((error) => {  console.log("error:", error);  });  },  },  methods: {  changeTips(mes: string) {  this.tips = mes;  },  },  });  </script> |

图5.14 客户端逻辑

为了响应代码编辑框中的C语言源码，我们在‘ace-builds’的实例中监听了代码框中文本的变化，当点击提交按钮后，将源码通过Axios提交到服务端。服务端响应后从返回数据中提取出编译器反馈信息、增强提示信息、修改示例，并分别绑定在变量‘gccResponse’、‘tips’、‘example’上，最后在前端进行展示。

## 5.4 本章小结

本章依次从数据收集、服务端设计、客户端设计的角度，展示了该服务的详细设计与实现过程，结合了开发过程中的核心代码段、图片介绍了该服务的设计思路与最终成果。

# 第六章 总结与展望

## 6.1 工作总结

随着计算机科学的发展，对于编译信息的分析与理解越来越重要。如何让初学者能更快的理解这些信息也就成为了重中之重。编译结果的形象化表示，将大大提高初学者调试程序的效率。如何做出增强提示在业内已有了很多的研究，针对我校样本的数据集，使得我们可以在之前研究的基础上做出本地化的结果，进而帮助教学工作的开展，在提效方面的成果值得期待。

本文提出的增强提示方法与实现是基于CSUOJ的数据集，在前人研究的基础上做出的本地化工作，它主要有以下功能和特点:

1. 实现了从数据库中提取所有的编译错误，并进行同类合并、去重。为教师教学、调研学生实践的具体效果提供了参考资料。
2. 对于常见的错误，结合实际教学的情况、初学者的编码行为，编写了增强提示。这些增强提示对于当下的教学具有很强的时效性，同时也提供了修改某条编译错误对应的增强提示的能力，为后续的改进增添了较大的灵活性。
3. 封装了轻量、灵活的增强提示服务，它本身时一个node模块，可以作为服务接入各系统的服务端。如CSUOJ、增强提示服务演示前端单页应用。特别的，在有需要的情况下，该服务可以被抽象为一个云函数，通过http的方式直接访问与调用。
4. 该方法具有可拓展性，并不是每一个编译错误都能通过文字描述的增强提示来解决。服务中集成了若干自定义逻辑用于处理复杂错误，随着未来教学的进行，会有更多的复杂错误需要加入增强提示中。服务提供了指引文档和代码，来帮助添加更多的自定义逻辑，以处理特殊的复杂错误。

## 6.2 工作的局限性

本次工作的局限性主要源自于数据集，为了方便教学工作的开展，本次研究的CSUOJ的数据集中大多是新手的代码，不过我们也主要是对新手感兴趣。由于GCC版本与每位学生在本地进行开发时的版本可能不一致，学生看到的编译器反馈信息可能与CSUOJ上提供的反馈信息不一致，这可能会导致学生面对反馈信息的反应不同，对增强提示的预期效果有所影响。

最后，我们的样本数据其实并不够大，仅仅分析了21届学生在CSUOJ上的代码，不同届学生、不同的授课班级、不同的专业都有进行C语言的学习，他们的表现可能会与现有数据集中的样本有所区别。

## 6.3 工作展望

由于人员和时间的限制，本次设计虽然完成了预期中的大部分工作，但是仍有许多地方可以进行改进。以后可以在本次工作的基础上进行改善，列举几点如下:

* 1. 将服务整理到CSUOJ的服务端，改写OJ的前端界面，使学生提交代码后可以参考增强提示。
  2. 验证增强提示服务的效果，改写OJ的服务端接口，收集学生在接收到增强提示后是否能自主修改错误，与不提供增强提示的样本进行对比，产出报告分析增强提示用于实践中的效果。
  3. 对于错误的研究可以不仅仅面向所有错误来开展，还可以从面向学生个人的角度进行拓展，收集每位学生的在OJ上产生了编译错误的代码，建立个人常见编译错误数据集，用于教师的教学的参考。
  4. 增强提示演示页面的前端功能不够丰富，可以增加新的功能：提供补充信息的填写入口、将增强提示演示页面与OJ进行关联，可以让学生修改错误后直接提交代码。
  5. 设计多语言增强提示开发模板，方便快速开发其他语言的增强提示。

# 致 谢

四年时光，聚散有时。行文至此，百感交集。完成这篇致谢，意味着我的本科生生活将要结束在2022年的这个盛夏。回望来路，自己的学识和思想在这大学四年间都有了很大的进步，在此我要多谢一路陪伴我走来的人们，是你们的陪伴让我的大学生活丰富多彩。

一朝沐杏雨，一生念师恩。感谢我的指导老师盛羽教授、李洪东教授。您们严谨治学的态度让我受益良多。感谢您们指出我论文中的不足之处，帮助我查漏补缺。同时感谢所有四年间的授课老师们，为我传授了诸多知识，并在学习生活中给予我极大帮助与鼓励。祝愿老师们未来工作顺利，平安幸福。

山河不足重，重在遇知己。感谢李彬学长在毕业设计中对于我工作内容的指导，并详细的给出了修改建议。感谢白一飞学长与我对于毕业设计的总体设计与方案进行讨论，帮助我梳理工作思路。感谢我的同学、室友在四年大学生活中对我的照顾以及支持，初见乍惊欢，久处仍怀然，人海茫茫，得以相遇，何其有幸。祝你们未来前程似锦，平安喜乐。

慈母手中线，游子身上衣。感谢我的父母与家人在生活上对我无微不至的照顾，让我在优良的环境下进行自己的学业。二十余年的养育之恩不是一朝一夕，希望我的父母身体健康，常乐久安。

以梦为马，不负韶华。最后感谢一路走来的自己，希望我的未来不忘初心，砥砺前行。

# 参考文献

[1]杨微微.计算机C语言程序的编辑分析[J].电脑知识与技术,2016,12(10):106-108.DOI:10.14004/j.cnki.ckt.2016.1051.

[2] Becker B A, Denny P, Pettit R, et al. Compiler error messages considered unhelpful: The landscape of text-based programming error message research[M]//Proceedings of the working group reports on innovation and technology in computer science education.

[3] 高云云. C语言编译系统的研究与实现[D].南京邮电大学,2019.

[4] Becker B A, Murray C, Tao T, et al. Fix the first, ignore the rest: Dealing with multiple compiler error messages[C]//Proceedings of the 49th ACM technical symposium on computer science education. 2018: 634-639.

[5] Becker B A, Goslin K, Glanville G. The effects of enhanced compiler error messages on a syntax error debugging test[C]//Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 2018: 640-645.

[6] Becker B A, Goslin K, Glanville G. The effects of enhanced compiler error messages on a syntax error debugging test[C]//Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 2018: 640-645.

[7] Becker B A, Goslin K, Glanville G. The effects of enhanced compiler error messages on a syntax error debugging test[C]//Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 2018: 640-645.

[8] Peter Mortensen. Compilation error: stray ‘\302’ in program［EB/OL］.https://stackoverflow.com/questions/19198332/compilation-error-stray-302-in-program-etc

[9] 颜晖. C语言程序设计实验指导[M].北京:高等教育出版社2008.

[10]丁春芳.C语言程序设计教学中程序调试探析[J].邢台职业技术学院学报, 2010 (3) :19-21.

[11]刘慧.编译原理中处理语法错误问题的研究[J].计算机教育,2007(05)

[12]李彩玲.基于Web的C语言教学系统的设计与实现[J].无线互联科技,2012(06):180-181.

[13],姚琳,武航星,张敏.C语言程序设计课程中的计算思维探析[J].中国大学教学,2014(09):59-62.

[14]李振.代码自动生成及代码上下文分析研究综述[J].数据通信,2020(02):44-49+53.

[15]王婷婷. 基于语义上下文的代码补全[D].哈尔滨工业大学,2021.DOI:10.27061/d.cnki.ghgdu.2021.001750.

[16]王希杰,王瑞庆.C语言教学改革的几点建议[J].中国科技信息,2006(24):213.