计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(3):95−102 [doi: 10.15888/j.cnki.csa.008382]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

C 语言代码缺陷分析辅助编程实践系统①

邱晓红, 杨瑞安, 敖紫迎, 陈佳丽

(江西理工大学 软件工程学院, 南昌 330013)
通信作者: 杨瑞安, E-mail: yra199508@163.com

摘　要: 网络课程 C 语言编程学习中, 师生互动性差、教学效率低, 学生难以凭借自身能力解决编程中常见的代码缺陷问题. 为了更好帮助学生解决学习中的难题, 辅助老师达到教学目的, 研究一款代码缺陷检测辅助学生编程实践系统. 该系统首先对易犯的代码缺陷分类, 分析编译器不易检测的语法、词法和语义缺陷; 然后构建智能分析器, 集成多种检测工具, 在系统中存储知识规则集合并扩展常见代码缺陷抽象模式; 最后检测代码并给出错误报告和修改意见, 通过配合学生模型辅助学生编程学习. 实验结果表明: 该系统能成功检测出常见代码缺陷并辅助学生编程实践.

关键词: 实践教学; 代码缺陷; 学生模型; 教学改革; 编程学习

引用格式: 邱晓红,杨瑞安,敖紫迎,陈佳丽.C 语言代码缺陷分析辅助编程实践系统.计算机系统应用,2022,31(3):95–102. http://www.c-s-a.org.cn/1003-3254/8382.html

Construction of C Language Code Defect Analyzer to Assist Programming Practice System

QIU Xiao-Hong, YANG Rui-An, AO Zi-Ying, CHEN Jia-Li

(School of Software Engineering, Jiangxi University of Science and Technology, Nanchang 330013, China)

Abstract: In the learning of online course C Language Programming, the interaction between teachers and students is poor, and the teaching efficiency is low. It is difficult for students to solve the common code defects in programming themselves. To better help students solve the problems in learning and assist teachers to achieve the teaching purpose, this study develops a practical system to assist students in programming. Firstly, the system classifies the common code defects, focusing on the analysis of syntactic, morphological, and semantic defects that are not easy to detect by the compiler. Secondly, it builds an intelligent analyzer that integrates a variety of detection tools to store the set of knowledge rules and extend the abstract pattern of common code defects. Finally, it detects the codes and gives error reports and modification suggestions. This system is capable of assisting students in programming by cooperating with the student model. The experimental results show that the system can successfully detect common code defects and thereby assist students in programming practice.

Key words: practice teaching; code defect; student model; teaching reform; programming learning

新冠疫情影响下, 各大高校学生课堂学习采用了“停课不停学”“停课不停教”的远程网络教学模式. 但与此同时, 远程网络教学也会带来弊端, 它不仅削弱了师生间学习交流的“互动性”, 造成“教”“学”分离的现象, 无形中增加了学生学习的难度. 特别是在编程类学科的学习过程中, 学生课堂学习时间短、知识点不易吸收, 课后老师能解答的问题也有限, 课余自主编程学习效率又低, 碰到难题无法及时解决. 传统的代码缺陷都是人工审查来完成, 因此存在过度依赖个人经验和低效等问题. 自动化代码检测编程实践系统就成为非计

① 基金项目: 江西省自然科学基金 (20181BAB202018); 江西理工大学研究生教改课题 (YJG2018015)
收稿时间: 2021-05-18; 修改时间: 2021-06-25; 采用时间: 2021-07-01; csa 在线出版时间: 2022-01-24

算专业学生以及初学者学习编程辅助软件, 既可以提高学生的编程效率, 还可以让学生可以自己解决编程问题. 从国内外研究成果来看, 目前编程实践系统有很多种. 文献 [1] 学者研究互动测试系统, 以提高学生在计算机程式设计课程的学习效果, 解决学生在课后学习中缺乏编程和缺陷实践的问题. 文献 [2] 提出了一种基于网络的学习活动与 Scratch 工具集成的方案来帮助学习者提高编程能力, 学生通过循序渐进学习知识内容从而反向辅助自己学习程序编程. 文献 [3] 设计一款编程测试练习平台, 平台中有很多案例程序, 可以很好帮助初学者和上班族闲暇时间学习. 文献 [4] 设计并实现了一个灵活的、自适应的在线高性能计算学习平台 EasyHPC, 平台不仅涵盖学习内容和经典案例, 同时嵌入检测工具, 加强代码检测技术, 在线编程学习和解决缺陷问题. 所以研发一款辅助学习 C 语言编程实践的系统, 加强代码检测技术是当前网络教学背景下代替老师的首要任务. 根据文献 [5] 和文献 [6]中几种检测工具的比较分析, 以及 Dev、Visual C++[7] 等编译器的检测效果比较, 虽然它们可以检测识别大部分错误, 但对于初学者常犯的那些代码缺陷, 例如字符混淆、变量名写错、空语句问题、运算符优先级问题、if-else 不配对问题、Case 语句后不以 break 结尾、数组越界等, 这些错误编译器和检测工具无法正确检测. 其中, Dev 对语法、词法缺陷上识别率不高, Visual C++ 对语法、语义缺陷上检测正存在不足, 而且没有明确的解释与修改意见, 老师在远程教学中也难以面对多个学生且复杂多样的问题, 这就会导致学生无法编程学习和解决问题.

因此, 为了帮助学生更好的学习编程, 辅助老师开展教学活动, 研究“辅助编程实践系统”来辅助学生学习编程, 解决编程中的常见问题. 首先, 需要对学生在编程中容易出现错误分类, 分析代码词法、语法和语义缺陷, 归纳出不易被编译器检测的缺陷以及代码知识规则; 其次, 设计辅助编程实践系统, 构建代码缺陷分析器, 该分析器集成 CppCheck[8]、PC-Lint[9] 以及 GCC 编译环境作为智能分析器的静态检测组成工具, 对学生编程中常犯代码缺陷分析, 分析这些代码缺陷的特征抽象模式, 利用正则表达式进行抽象表示, 并扩展到代码知识规则库中; 然后对代码进行检测处理, 与知识规则库中所有缺陷模式匹配, 并给出检查分析报告; 最后, 系统会根据检测分析报告给出相应的错误原因和修改意见, 在远程教学红很方便的代替老师, 达到一个电子智慧教师的身份, 辅助学生编程实践. 同时还会提供符合的教学视频和练习习题帮助学生巩固缺少的知识点.

1 C 语言易犯代码缺陷分类

代码缺陷的本质就是程序中有不符合语言标准、违背良好的设计规范和编码规范. 代码看似没有问题, 但是运行起来违背开发者的意愿, 并有可能导致程序崩溃的陷阱. 这里探讨的陷阱分为词法、语法和语义缺陷.

1.1 词法陷阱

初学者刚刚接触编程, 对编程的规范性还不是很熟悉. 有时候在编程中因为主观意识和敲击键盘的不注意, 容易导致代码名词写错, 一般会有如下错误情况: 漏写字符; 字符重复; 字符顺序颠倒; 相似字符混淆. C 语言词法中常见陷阱如下: =和==混淆; &&、||与&、| 混淆; 字符和字符串格式混淆; 变量名写错导致未定义.

1.2 语法陷阱

各种字符、关键词组合不同就产生不同的含义, 编译器对这些组合语句有都特定的定义和识别方法. 初学者在编程过程中, 尽管代码符合语法格式, 但结合上下文, 就有可能存在语法缺陷. C 语言语法中常见陷阱如下: if、for、while 语句后接分号导致空语句; case 语句后不以 break 结尾; 运算符优先级问题; “悬挂” else 引发的配对问题; 输入函数 scanf() 漏写取地址符.

1.3 语义陷阱

在词法和语法都编写正确的情况下, 词义陷阱也是初学者常犯的错误. 词义陷阱主要是指运行过程中出现的逻辑问题, 这将导致运行结果的不正确性, 而这些问题大多也难以被编译器检测出来. C 语言语义中常见陷阱如下: 未初始化变量或没有分配合法的值; 可能存在空指针; 深拷贝与浅拷贝; 存在数组越界; 两个整数求商等于 0.

2 辅助编程实践系统

系统集成代码缺陷分析器中, 存储了 C 语言知识规则和常见缺陷特征信息, 可以智能分析检测代码并帮助学生编程实践. 本节重点论述缺陷分析器、代码缺陷检测以及偏差纠正学生模型的设计.

2.1 系统体系结构

辅助编程实践系统主要由人机交互接口、知识规

则库、代码缺陷分析器、教师模型、学生模型、解释器等 6 部分构成, 如图 1 所示.

(1) 人机交互接口: 人机交互接口是系统与学生数据交互的模块, 同时包括自然语言处理技术的应用、人机对话的处理、对领域知识库扩展和维护的接口.

(2) 知识规则库: 存储开源检测工具的内置缺陷规则集, 以及扩充第 1 节分类中常见的缺陷模式, 作为检测分析学生编写程序的重要知识源.

(3) 代码缺陷分析器: 该系统的分析器由开源检测工具集成 GCC 编译器构成. 分别可以对程序进行语法、语义、非语法上的错误进行检测[10].

(4) 教师模型: 其主要任务将教学视频和相关练习题并通过接口以适当的表达形式提供给学生.

(5) 学生模型: 对学生学习和编程实践过程中进行全面分析, 最终判断学生是否达到专家标准的缺陷纠正能力, 从而做出下一步教学策略[11].

(6) 解释器: 分析器检测出的缺陷报告在匹配知识库中的规则之后, 会由解释器做出详细解释报告和修改意见, 并定位缺陷位置, 反馈给学生.

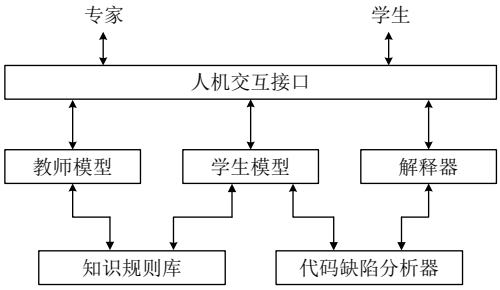

图 1 辅助编程实践系统组成结构

2.2 代码缺陷分析器

代码缺陷分析器同时利用静态检测工具和本地动态检测, 并扩展一些学生编程中常犯的缺陷, 强化静态分析技术.

2.2.1 组成结构

代码缺陷分析器由规则集加载器、静态检测工具、缺陷规则集合、本地动态检测、检测结果解释器等 5 部分构成, 如图 2 所示.

(1) 规则集加载器: 负责把知识规则库中存储的一些缺陷规则和特征信息加载到内存中, 便于在检测时候匹配错误缺陷.

(2) 静态检测工具: 集成 CppCheck、PC-Lint 和 GCC 编译环境[12]. PC-Lint 是逻辑语义分析的检测工具; CppCheck 是针对代码非语法错误的检测工具; GCC 是 C 语言编译器底层编译程序工具.

(3) 缺陷规则集合: 集合中有 CppCheck 中的内置检测类和缺陷模式、PC-Lint 代码错误警告列表和 GCC 编译器中的一些语法规则, 还有自定义的扩充检测类型和缺陷模式[13].

(4) 本地动态检测: 根据扩充的缺陷规则去实时检测代码, 并用正则表达式去匹配缺陷, 同时反馈错误提示报告给用户.

(5) 检测结果解释器: 匹配知识库中的对应规则后并给出错误原因.

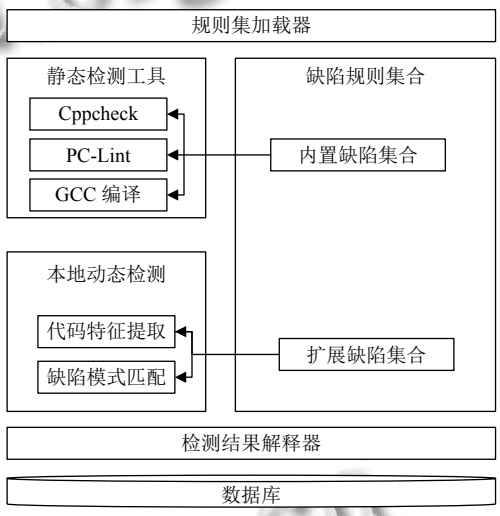

图 2 智能分析器体系结构

2.2.2 知识规则库

CppCheck: 主要存储 CppCheck 检查后相关问题的描述, 分类及建议解决方法. 如表 1 所示.

表 1 CppCheck 检查知识规则

错误代码	错误原因	修改意见
Code50001	变量为分配合法的值	为变量赋正确的值
Code50002	未初始化变量	正确初始化变量
Code50003	可能存在空指针	正确初始化指针
Code50004	存在数组越界	做好索引限制或扩容数组
Code50005	变量声明了却未使用	删除无用的变量
Code50006	内存泄露	及时释放无用的内存
Code50007	资源泄露	及时释放无用的读入资源
Code50008	变量没有定义	重新定义或者查看上下文是否都相似变量

GCC 错误代码中文对照: 主要存储相关编译错误代码其中文解释.

GCC 错误代码解决方案: 主要存储相关编译错误

代码, 相似案例代码, 以及建议解决方法.

运行时问题: 主要存储运行时问题描述、对应操作及它们之间的逻辑关系.

2.2.3 扩展常见代码缺陷

将第 1 节中的编译器和检测工具无法检测的错误整理出, 分析其缺陷模式并提取抽象模式, 利用正则表达式的进行特征表示, 如表 2 所示.

表 2 新增缺陷规则

缺陷模式	抽象模式
=和==混淆	*if *\(.*[A-Za-z0-9_]+ *= *[A-Za-z0-9_]+.*\).*
&&、||与&、|混淆	*if *\(.*[A-Za-z0-9_]+ *&& *[A-Za-z0-9_]+.*\).*
If、for、while语句后接分号导致空语句	*\(.[if, for, while]*\);.*
Case语句后不以break结尾	*case.*(?<!break;)$
“悬挂”else引发的配对问题	\s*if.*(?<!\{)$
函数scanf()漏写&取地址符	.*scanf(.*, (?!&)\w.*).*

2.3 代码缺陷智能分析检测

2.3.1 缺陷检测处理框架流程

首先通过命令行读入待检测文件的路径, 然后遍历所有文件夹下面的 c/cpp 文件, 每遍历一个 c/cpp 文件, 对读取的程序进行预处理, 去除注释语句. 静态检测工具通过字符流将程序读入到预处理后程序的首部, 对处理后的程序进行词法分析, 建立 Token 双链表, 并对 Token 链表进行语义分析, 通过循环调用检查类中定义的缺陷模式来实现缺陷的检测, 最后通过定位找出程序中存在缺陷. 运行流程如图 3 所示.

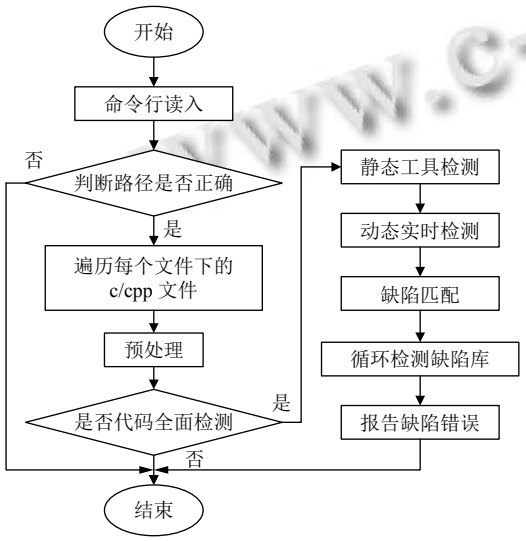

图 3 缺陷检测处理框架流程图

2.3.2 抽象缺陷模式匹配算法

利用缺陷的抽象模式, 通过正则表达式去匹配字符串, 如果匹配成功则返回 true, 否则匹配不成功, 就不存在该缺陷[14,15]. 缺陷模式匹配算法的伪代码如算法 1.

算法 1. 缺陷模式匹配算法的伪代码

```
Procedure Check(code[]/**/, detect[]/**/)
/*创建一个新的链表*/
Begin: *p=code, *q=defect, Link *s=L
While(p 是否为空)
  While(q 是否为空)
    if(p 中的内容与 q 中的内容进行正则表达式匹配是否为 true)
      newNode /*创建一个新的结点*/
      newNode->data=q->data /*将 q 中的缺陷模式赋值给 s*/
      s->next= newNode /*尾插法插入新结点 s*/
      s= newNode /*指针指向新结点*/
    End if
    q=q->next /*q 指针向后移一位*/
  End while
  p =p->next/*p 指针向后移一位*/
End while
End
Return L
```

2.4 学生模型

2.4.1 学生模型结构

根据分析, 考虑既要全面描述学生, 又便于实现, 将覆盖、偏差、认知模型进行复核和简化, 得到一个综合学生模型结构, 并集成在系统中. 其数据结构关系模型见图 4.

学生基本信息 (学生编号, 姓名)
专家知识库 (知识点编号, 知识内容, 难度, 知识特性)
易犯错误案例库 (知识点编号, 错误编号, 掌握难度)
学习策略库 (知识点编号, 呈现方式, 教学方式)
认知能力表 (知识点编号, 识记, 理解, 应用, 分析, 综合, 评价)
学习测评表 (试题编号, 对应知识点, 测试分值, 错误编号)
错误类型表 (错误编号, 错误类型, 错误描述, 学习提示)

图 4 学生模型的数据结构 (关系模型)

在该模型中, 一方面吸收了覆盖模型的成分, 用于计算和记录学生的学习进度, 保证后续提供的学习内容和知识点不会重复. 二是吸收了偏差模型的成分, 每次编程学习并测试后都要计算学习偏, 即找到本次编程中缺少的知识点和认识错误的知识, 系统确定需要弥补和加强的知识, 并提出需要重新学习的内容[16]. 如此反复编程实践直到掌握缺少的知识. 三是引入了认知模型的成分, 每次测试后都需要分析计算认知能力, 调整重新学习内容的难度和所有缺少知识点的侧重点.

上述数据结构描述了学生模型的静态结构, 而图 5 则描述了学生模型的动态结构. 动态结构表现了学习中数据的动态过程. 可以看出, 它跟踪学生的学习活动, 通过对学生学习行为的分析, 记录并调整学生的知识结构、学习能力等描述学生个性化特征的信息, 从此得出教学策略. 其中, 决策机处于中心地位, 它的功能是: 有关数据库的维护、决策的生成、决策的冲突消除等[17].

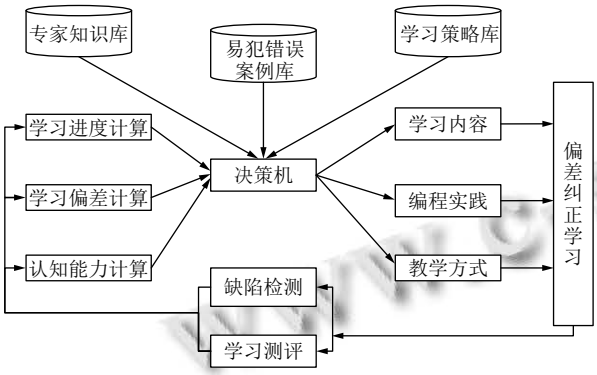

图 5 学生模型的动态结构图

2.4.2 偏差纠正学习

根据缺陷错误报告合理的找出需要学习的知识点, 并在视频库中选出符合该学生学习的教学知识点讲解视频, 同时在习题库中智能匹配出覆盖全部知识点的测试题试卷. 偏差纠正学习流程如图 6 所示.

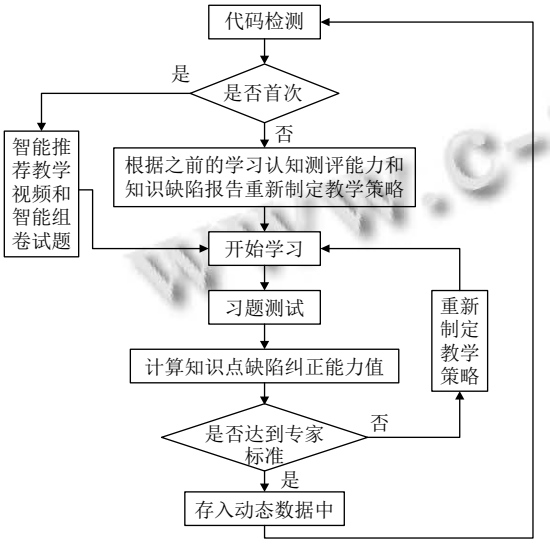

图 6 偏差纠正学习流程

学生进入系统代码检测之后, 首先判断该学生是否是第一次使用系统进行检测, 若是第一次使用就直接推荐代码中知识缺陷教学讲解视频, 然后开始学习, 配合做习题进行测试, 系统会在交卷时给出相应知识点掌握程度和缺陷纠正能力值, 若没有达到专家的标准, 则重新制定教学策略再次学习. 如果达到了专家标准可以选择继续练习编程, 发现新的知识缺陷. 如果不是第一次使用系统, 则系统会根据之前的学习认知测评能力和缺陷报告重新制定教学策略和教学内容.

2.4.3 偏差纠正认知能力计算

学生在学习完对应的教学视频之后, 首先会通过相应的练习测验题来巩固所学的知识, 然后再进行测试. 该知识点测试题目一般由单选题、判断题、填空题、编程题四大部分组成. 题库中至少要能测试“识记”“理解”“应用”“分析”“综合”“评价”几项认知能力中的一项能力. 具体过程如下:

(1) 学生在做测试题时对于每一道题都有各项认知能力指标, 如果这道题答对则能力值为 1, 答错能力值为−1, 不答就为 0. 学生在做完每一套测试题后都会得到一张测试记录表, 如表 3 所示.

表 3 不同认知能力 (R_{ij}) 的测试记录表

题号	标记	理解	应用	分析	综合	评价
1	−1	1	−1	0	1	0
2	0	−1	1	1	−1	1
3	1	−1	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	−1	0	1	1	0	0

(2) 学生答完测试题就可以得到每道题的各项能力的正确率矢量:

$$A_i=(a_1,a_2,a_3,a_4,a_5,a_6),\ a_i\in[0,1],\ 0\leqslant i\leqslant 6 \tag{1}$$

每一道题的 6 项指标都如这样表示就可以构成整张试卷所有题目认知能力的评价矩阵:

$$\begin{cases} G=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{16} \\ a_{21} & a_{22} & \cdots & a_{26} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n6} \end{bmatrix}, \\ a_{ij}\in[0,1], 1\leqslant i\leqslant n, 1\leqslant j\leqslant 6 \end{cases} \tag{2}$$

其中, a_{11}–a_{16} 表示第 1 道题的 6 项认知能力评价值, a_{21}–a_{26} 表示第 2 道题的 6 项能力评价值, 以此类推.

(3) 定义每道题的相对权值为$W=(W_1,W_2,\cdots,W_n)$.

$$\sum_{i=1}^{n}W_i=1 \tag{3}$$

(4) 每道题的权值由专家或者出题者给出:

权重$w_1, w_2, \cdots, w_n$, 其中, w_i为第 i 道题的权重, $w_i \in [0,1]$, $0 \leqslant i \leqslant n$.

$$W_i = \frac{w_i}{\sum_{j=1}^{n} w_j}, 0 \leqslant i \leqslant n, 0 \leqslant j \leqslant n \tag{4}$$

(5) 计算测试后的最终认知能力评价结果矢量:

$$v = w * G = (v_1, v_2, v_3, v_4, v_5, v_6) \tag{5}$$

(6) 最后计算出该学生对于该部分的知识点和知识缺陷纠正综合能力值:

$$M = \sum_{i=1}^{6} v_i \times Q_i \tag{6}$$

其中, Q 为各项指标能力的综合权值, 这也由专家给出. 将计算出学生的认知模型和试卷专家给出的难易程度以及知识点覆盖率做出综合判断, 最后决定该学生是否已经掌握该部分知识点和知识缺陷纠正能力.

3 实验测试

3.1 代码缺陷检测

下面程序的主要功能是判断从控制台输入 flag 来分别执行不同的功能, 功能有求两个数的最大公约数和最小公倍数、计算一个数是否为质量、求解两个数的商, 对一个数组进行冒泡排序.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>
//求最大公约和最小公倍
int dump(double m, double n){
int k=0, t, a, b;
a=m;b=n;
if(m<n);{
  t=m;m=n;n=t;
}
while(k!=0)
k=m%n;
  m=n;n=k;
printf("最大公约数: %d\n", m);
printf("最小公倍数: %d\n", a*b/m);
return 0;
}
//求质数
intzhishu(double k){
for(inti=2;i<k;i++);{
 int s=i%2;
 if(s=0){
   printf("%s", "不是质数");break;
 }
}
return 0;
}
//求两个数的商
intalg(double m, double n){
if(m>0&n<0){
 printf("%d", m/n);
}else{
 printf("请输入两个正整数: ");
}
}
//冒泡排序
Intinit_change(){
int array[8] = {49, 38, 65, 97, 76, 13, 27, 49};
inti, j, al;
int key;
for (i = 0; i< 8; i++){
key=0;
 a1=array[9];
for (j = 0; j+1<8-i; j++){
if (array[j] > array[j+1]){
key=1;
swap(&array[j], &array[j+1]);
    }
  }
if (key==0)
break;
 }
return 0;
}
int main(){
floatm, n;
scanf("%f", m);
scanf("%f", n);
int flag;
scanf("%d", &flag);
if(flag&&m&&n!=0){
 switch(flag){
 case 1:init_dump(m, n);
 case 2:init_zhishu(m);
 case 3:init_alg(m, n);
 case 4:init_change();
 default:printf("%s", "结束");
}
}
return 0;
}
```

使用 Dev 去编译程序代码, 编译结果显示是没有

任何警告和错误, 说明编译器 Dev 无法检测出学生编程中非语法错误. 使用 Visual C++编译器去编译程序代码, 编辑结果显示出一些警告信息, 其中包含了部分代码中存在的错误和语法规则问题, 但是同时也检测出一些不应该存在的错误, 并且只说明了错误原因, 没有解释说明如何修改错误. 所以 Visual C++在检测正确识别率和覆盖率性能上都不是很乐观. Dev 与 Visual 编译结果如图 7 所示.

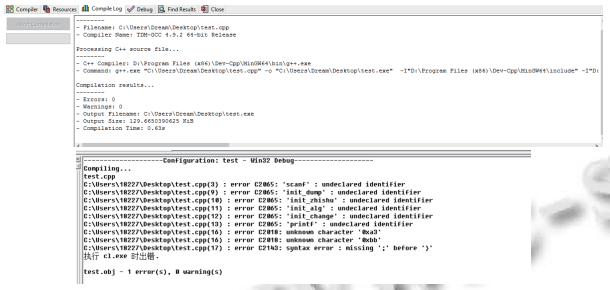

图 7 Dev 与 Visual 编译结果效果图

而使用智能分析器能检测出很多语法和逻辑上的错误, 第 1 节总结的缺陷都是初学者容易犯的错误, 编译器根本无法全部检测这些缺陷, 这款辅助工具就可以帮助初学者检测出问题, 说明在检测覆盖率和识别正确率得到了很大的提升. 代码检测报告如图 8 所示.

图 8 检测结果报告

分别对 Dev、Visual C++、本文中的智能分析器检测新增缺陷的性能做出比较, 性能比较结果如表 4 所示.

表 4 性能对比分析表

工具	缺陷覆盖率	识别正确率
Dev	无法检测新增缺陷	无法识别
Visual C++	覆盖部分新增缺陷错误	部分识别正确
智能分析器	几乎覆盖学生易犯缺陷	正确率比Visual C++要高一些

3.2 辅助学生编程实践学习

对于上一步检测结果报告每一个知识点错误都有对应的习题测试和教学视频链接, 学生点击链接之后, 可以通过学习教学视频和习题测试巩固学习缺失的知识点. 效果图如图 9 和图 10 所示.

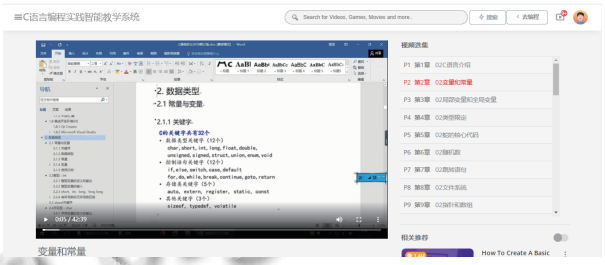

图 9 视频学习截图

视频学习过程中同时可以返回继续编程并修改自己的错误, 系统会重新检查代码, 匹配重新编程之后的错误教学视频. 视频学习之后学生可以开始习题测试, 测试结束之后会由系统计算出缺陷纠正认知能力值从而判断学生对这些知识点是否已经掌握, 如果没有达到专家的标准, 系统就会重新匹配符合学生能力的测试题继续学习, 如果达到了专家标准学生就可以继续编程发现其他的知识点缺陷.

图 10 习题测试截图

4 结论与展望

目前, 该系统在测试中获得了学生的一致好评, 检测能力强, 智能分析错误并给出修改意见, 大部分学生自主编程实践能力得到提升. 但是代码缺陷分析器主要还是针对编程初学者的易犯错误给出了智能提示和相应修改建议, 相比其他的编译器在覆盖率和识别率都有显著提高, 但对于有几年编程经验的程序员可能就不太适用. 这时就需要重新扩展知识库并丰富数据库的推理规则, 做到和现在很多高级语言的编译器一样能够提示很多逻辑语义上的错误. 因此, 该系统还存

在一些不足之处, 需要更好的扩展知识库和强化推理机制, 让更多的初学者体会到智能编程实践的优点.

参考文献

1 Yang TC, Yang SJH, Hwang GJ. Development of an interactive test system for students' improving learning outcomes in a computer programming course. 2014 IEEE 14th International Conference on Advanced Learning Technologies. Athens: IEEE, 2014. 637–639. [doi: 10.1109/ICALT.2014.186]

2 Su JM, Wang SJ. A Web-based learning activity integrated with scratch tool to support programming learning. 2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media). Pattaya: IEEE, 2017. 1–4. [doi: 10.1109/UMEDIA.2017.8074137]

3 Karabulut M, Mayda İ. Web based algorithm exercise and assessment management system for computer programming students: AlgoBug. 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). Istanbul: IEEE, 2020. 1–7. [doi: 10.1109/ISMSIT50672.2020.9254480]

4 Zou ZP, Zhang YX, Li J, *et al*. EasyHPC: An online programming platform for learning high performance computing. 2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE). Hong Kong: IEEE, 2017. 432–435. [doi: 10.1109/TALE.2017.8252374]

5 古可, 刘超, 金茂忠. C++代码缺陷自动检测工具的研究与实现. 计算机应用研究, 2009, 26(5): 1628–1631.

6 Fatima A, Bibi S, Hanif R. Comparative study on static code analysis tools for C/C++. 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST). Islamabad: IEEE, 2018. 465–469. [doi: 10.1109/IBCAST.2018.8312265]

7 Arusoaie A, Ciobâca S, Craciun V, *et al*. A comparison of open-source static analysis tools for vulnerability detection in C/C++ code. 2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). Timisoara: IEEE, 2017. 161–168. [doi: 10.1109/SYNASC.2017.00035]

8 张仕金, 尚赵伟. Cppcheck 的软件缺陷模式分析与定位. 计算机工程与应用, 2015, 51(3): 69–73.

9 姜文, 刘立康. 基于持续集成的 PC-Lint 静态检查. 计算机技术与发展, 2016, 26(11): 31–36.

10 赵正旭, 梅成芳, 张强. 嵌入式软件静态测试方法研究. 计算机技术与发展, 2019, 29(3): 64–68.

11 Thai-Nghe N, Schmidt-Thieme L. Multi-relational factorization models for student modeling in intelligent tutoring systems. 2015 Seventh International Conference on Knowledge and Systems Engineering (KSE). Ho Chi Minh City: IEEE, 2015. 61–66. [doi: 10.1109/KSE.2015.9]

12 朱亚伟, 左志强, 王林章, 等. C 程序内存泄漏智能化检测方法. 软件学报, 2019, 30(5): 1330–1341. [doi: 10.13328/j.cnki.jos.005715]

13 周风顺, 王林章, 李宣东. C/C++程序缺陷自动修复与确认方法. 软件学报, 2019, 30(5): 1243–1255. [doi: 10.13328/j.cnki.jos.005729]

14 邱景, 苏小红, 马培军. 一种使用静态分析的汇编代码缺陷检测方法. 哈尔滨工业大学学报, 2013, 45(2): 53–59.

15 罗琴灵, 蒋朝惠. 多策略软件代码缺陷检测方法研究. 贵州大学学报 (自然科学版), 2015, 32(3): 113–118.

16 陆申明, 左志强, 王林章. 静态程序分析并行化研究进展. 软件学报, 2020, 31(5): 1243–1254. [doi: 10.13328/j.cnki.jos.005950]

17 Taufik R, Nurjanah D. An intelligent tutoring system with adaptive exercises based on a student's knowledge and misconception. 2019 IEEE International Conference on Engineering, Technology and Education (TALE). Yogyakarta: IEEE, 2019. 1–5. [doi: 10.1109/TALE48000.2019.9226001]