**天津工业大学**

**毕业设计（论文）**

**面向中学生的“竞赛算法”可视化编程教学系统**

**姓 名 叶文滔**

**学 院 计算机科学与技术学院**

**专 业 计算机科学与技术**

**指导教师 王作为**

**职 称 讲师（高校）**

**2021 年 00 月 00 日**

摘　　要

随着信息技术的发展，计算机正被越来越多人所触及并应用。计算机作为第四次工业革命（也称工业4.0）的核心工具，其在一个国家内的普及程度将决定了第四次工业革命的广度，因此国家推出了各种各样的政策，引导计算机在青少年中的普及。其中便包括了举办计算机程序设计竞赛、对中小学校建立并推广计算机学科提供的政策及资金扶持等。这也使得越来越多的学生对计算机产生兴趣，并参与到计算机编程的学习中。

对于青少年的编程学习，一般可分为“编程思维”、“数据结构”、“算法”三个阶段。“编程思维”着重学习C语言程序的运行原理，如顺序、分支、循环三大结构；“数据结构”着重学习数据在计算机中的存放方式，使学生了解数据类型在计算机程序中的意义；“算法”则是学习使用计算机编程解决生活中的数学问题的方式。本项目主要以中学青少年计算机爱好者为目标人群，以“竞赛算法”所涉及的核心算法为教学目的，利用计算机相对成熟的技术来实现相关算法的可视化教学。并引入不同的人机交互，让学生更加直观地理解竞赛相关算法，提高学生的学习兴趣。

本课题实现了一个可即时编程并实时演示数据结构的可交互编程系统，引入面向中学生“竞赛算法”的教案，实现对教案中相关算法的可视化演示。该系统使用编译原理技术实现对C语言程序的识别和数据提取，并采用web技术实现过程数据可视化。这使得该系统可运行于各种系统与形态中，方便教师和学生的使用。该系统既可以加深学生对基本算法的学习和理解，又能够实现动态演示不同参数下的各种算法的状态及变化，实现中间的数据的输出，提高学生的动手及实践能力。

关键词：中学生；计算机；编程；C语言；竞赛算法；可视化；Web开发

*采用小四号、宋体、接排*

*小四号、黑体、顶格*

ABSTRACT

With the development of information technology, computers are known and being widely used by more and more people. As the core tool of the fourth industrial revolution (also known as Industry 4.0), the popularity of computers in a country will determine the breath of the fourth industrial revolution. Therefore, our country has launched a variety of policies to guide the popularity of computers among teenagers. These include holding computer programming competitions, providing policy and financial support for the establishment and promotion of computer science in primary and middle schools. This also makes more and more students become interested in computers and participate in the learning of computer programming.

Generally, learning process of computer programming for teenagers can be divided into three stages: logic, data structure and algorithm. “Logic” focuses on learning the operating principle of C language programs, such as the three structures of sequence, branch, and loop. “Data structure” focuses on learning how data is stored in the computer. “Algorithm” focus on learning how to solve mathematical problems in life with computer programming. This project is mainly aimed at young computer enthusiasts in middle schools, with the core algorithm in “competition algorithm” as the teaching purpose, and use some relatively mature technology to achieve the visual teaching of those algorithms. It also introduces different human-computer interactions to allow students to understand competition algorithms more intuitively and increase their learning interest.

This topic implements an interactive programming system that can be programmed instantly and show data structure in real time. With “competition algorithms” for middle school students teaching plans introduced, it can realize the visual demonstration of related algorithms in the teaching plans. The system uses compilation principle to recognize and extract data from C language programs, and uses web platform to realize process and data visualization. This allows the system to run in various systems and forms, makes it convenient for teachers and students to use. The system can not only deepen student’s learning and understanding of basic algorithms, but also can dynamically demonstrate the state and changes of various algorithms under different parameters, and the middle-state data output, which can improve student’s hands-on and practical ability.

**Key words：Middle school students, computer, programming, C language, competition algorithm, visualization, web**

目 录

目录

[摘　　要 2](#_Toc71553793)

[ABSTRACT 3](#_Toc71553794)

[目 录 4](#_Toc71553795)

[目 录 6](#_Toc71553796)

[第一章　　绪论 1](#_Toc71553797)

[1.1 课题研究背景 1](#_Toc71553798)

[1.2 国内研究现状 1](#_Toc71553799)

[1.4 本文主要结构 2](#_Toc71553800)

[第二章　　相关开发技术介绍 4](#_Toc71553801)

[2.1 TypeScript 4](#_Toc71553802)

[2.2 React 5](#_Toc71553803)

[2.3 webpack 6](#_Toc71553804)

[2.4 Node.js 7](#_Toc71553805)

[第三章　　系统总体设计 9](#_Toc71553806)

[3.1 需求分析 9](#_Toc71553807)

[3.2 系统功能设计 9](#_Toc71553808)

[3.3系统整体结构设计 10](#_Toc71553809)

[第四章　　重要模块设计——代码服务 14](#_Toc71553810)

[4.1 数据结构 14](#_Toc71553811)

[4.2 方法说明 15](#_Toc71553812)

[第五章　　核心模块设计——编译器 17](#_Toc71553813)

[5.1 编译原理概述 17](#_Toc71553814)

[5.2 编译器的结构 17](#_Toc71553815)

[5.3 语言及其文法基本概念 19](#_Toc71553816)

[5.3.1 字母表 19](#_Toc71553817)

[5.3.2 文法的定义 20](#_Toc71553818)

[5.3.3 文法的分类 21](#_Toc71553819)

[5.3.4 CFG的分析树 22](#_Toc71553820)

[5.4 词法分析 23](#_Toc71553821)

[5.4.1 有穷自动机 23](#_Toc71553822)

[5.4.2 使用DFA进行C语言词法分析 24](#_Toc71553823)

[5.5 语法分析 26](#_Toc71553824)

[5.5.1 移入-归约语法分析技术 27](#_Toc71553825)

[5.5.2 LR分析法 27](#_Toc71553826)

[5.5.3 C语言识别文法 28](#_Toc71553827)

[第六章　　其余模块设计 30](#_Toc71553828)

[5.1 教案 30](#_Toc71553829)

[5.2 教案加载器 30](#_Toc71553830)

[5.3 页面视图 31](#_Toc71553831)

[第负一章　　模板 32](#_Toc71553832)

目 录

*（三号、黑体、居中、目录两字空两格、与正文空一行）*

第一章　　绪论*………………………………………………………*×

1.1☆☆☆☆*（小四号宋体）*………………………………………………………×

1.2☆☆☆☆**………………………………………………………………………**×

1.3☆☆☆☆………………………………………………………**………………**×

………………

参考文献*（四号、宋体）………………………………………*………×

附录*（四号、宋体）……………………………………………*……**…**×

致谢*（四号、宋体）……………………………………………………*×

*不标页码*

第一章　　绪论

1.1 课题研究背景

正文开始标注页眉，宋体五号居中

C语言作为计算机领域的一门经典语言，在过去的十几年间被持续广泛地使用，也被国内各大高校和各类计算机培训机构作为首选编程语言。其实践性高、综合性强，相对于其他高级语言更贴近计算机的运行原理，大部分语法亦可应用到其他语言中去，因此相当多人选用此语言作为计算机编程基础知识学习以及系统级的程序编写工具。这也使得C语言得以在Tiobe编程语言排行榜[1]中长期位列第一或第二。（在过去的20年间亦有很多年由Java承担榜首的位置，这与网站后端开发与移动端开发有较大关系。）

正因如此，国内对青少年的计算机编程教学，目前大多数使用的是C语言。尽管C语言相对其他大多数高级语言具有更低的功能性和更高的学习难度，但由于其相对贴近计算机的运行原理，语法与其他大多数语言相近，因此可作为青少年学习编程的首选语言。

但随着计算机行业的发展，国内计算机教育的问题也逐渐显现了出来——计算机学科由于其本质的枯燥性，使得多数学生无法独立进入此领域。尽管一些民营计算机教育中心拥有较良好的教育方式能使学生快速上手，但在传统学校中，受到国内“成绩决定未来”的恶性气氛影响，学校配备的计算机相关师资力量仍不足以使学生对计算机学科具有较好的理解。因此改进计算机学科的教学方法，是推进计算机教育的重要一环。

1.2 国内研究现状

编程语言的学习一般分为被动式学习和主动自学两种。相对于其他语言，国内C语言学习者大多数情况下的学习方式是被动学习，即接受高等教育学校提供的相关教学。这种教学方式往往更注重基础能力的积累，不注重新功能的传授，因此学习速度并不高。但由于存在任务性需求，加之“教师为主体、学生被动参与”的传统学习方法已沿用多年，因此绝大多数人都可以接受。

对于接受编程思想能力更强的人来说，脱离传统教学方式的自主学习往往是更优的方案。由于C语言的火爆性，目前能给C语言初学者提供的自学书籍教材相当多，如C语言之父合著编写的书籍——《C程序设计语言》，全面地讲述了C语言的基础语法和见解；《C语言程序设计：现代方法》，以一种独特的视角介绍C语言及其变化；面向C语言专业研究者的《C Primer Plus》，给读者更好地讲述C语言的深层用法及原理；数据结构介绍书籍《大话数据结构》，以通俗有趣的方式介绍数据结构的概念；算法类书籍《啊哈！算法》，从实际应用出发，以幽默的语言和插图来讲解算法；《算法导论》，深入浅出、全面地介绍了计算机算法。

相比书籍，现在更为流行的自学方法是通过编程自学网站平台进行学习。这些网站有以教学为主，在线编辑器为辅的菜鸟教程（ruboob.com）、w3school（w3school.com.cn）；以在线编辑器动手操作优先的ITbegin（itbegin.com）、以真题练习为主的力扣（leetcode-cn.com）、牛客网（newcoder.com）等。它们能提供除C语言外更多的编程语言及框架的教案，扩展性和时效性都更好。人们可以根据自己的喜好和学习环境，选择自己喜欢的方式、喜欢的教材进行学习。

同时，也有通过视频教学进行自主学习的方案可选。视频教学方案结合了传统教学和其他几种自主学习方式的优点，在讲授式教学的同时可以随时调整速度和进度，可以适应更多场景，解决了上课可能不太方便带电脑进行当场实践的问题，是一种不错的选择。相关网站有慕课网（imooc.com）、中国大学MOOC（icourse163.org）等。

1.3 课题研究内容

本课题针对现有的C语言在线编程环境无法为用户提供调试功能、IDE无法提供图文教学功能、教案书籍无法提供与程序运行实时对应的图文演示功能进行了功能整合，借鉴了国内C语言学习环境的研究成果，针对学生和教室对C语言的学习和教学需求，设计了一套能在网页和桌面客户端内完成程序编写、程序解释单步执行、断点调试、实时图形展示的系统。

本课题重点对如何实现C语言解释编译器的关键技术做了研究，通过此套系统提供程序运行的变量信息和程序运行状态，为上层服务；并在此基础上使用Web技术实现页面框架的搭建，最终可使用这套平台为教案和程序搭起一座桥梁，进而完成本系统的功能实现。

1.4 本文主要结构

本文内容分为以下几个章节：

第一章为绪论。首先对本课题的研究背景和国内的研究现状做了简要介绍，然后说明本课题的主要研究内容及内容安排。

第二章为相关开发技术介绍。简要介绍了本工程应用到的几项重点技术，包括工程语言 TypeScript、前端开发框架React、工程打包处理框架webpack，以及基于V8引擎的JavaScript运行时系统node.js。

第三章为系统总体设计。首先根据课题进行了需求分析，并根据需求设计出相应的软件系统解决方案。然后从两个视角描述了系统的整体结构设计，并简要介绍了系统运行的流程。

第四章为重要模块设计——代码服务。代码服务负责为代码编辑器和编译器提供中间支持，是这两者之间承上启下且不可或缺的角色。

第五章为核心模块设计——编译器。编译器用于对C语言程序进行编译操作，以为代码编辑器提供语法高亮、变量表、指令队列等功能。它是本系统的低层核心功能，包含了词法分析器和语法分析器。

第六章为其他模块设计，分别介绍了教案、教案加载器、页面视图三个模块的设计。

第二章　　相关开发技术介绍

本章主要介绍了本系统实现过程中使用到的相关开发技术，包括工程语言 TypeScript、前端开发框架React、工程打包处理框架webpack，以及基于V8引擎的JavaScript运行时系统node.js。

2.1 TypeScript

TypeScript是JavaScript的超集，它可以编译成纯JavaScript，是一款可以在几乎任何浏览器、任何计算机和任何操作系统上运行的开源编程语言[2]。



图2-1　　TypeScript图标

在基于浏览器的编程语言发展历程中，JavaScript于1998年由Netscape（网景）[3]推出后大获成功，并于后期浏览器开发语言的竞争中取得优势地位，由ECMA国际标准化组织统一了ECMAScript标准[4]，并占据了所有浏览器编程语言的市场。简单来说，目前在主流浏览器上支持的语言是JavaScript，它基于ECMAScript标准持续迭代。

JavaScript作为一款专用于浏览器环境的脚本语言，对小型项目支持良好，对新接触程序开发的程序员表现友好，因此它能在网络大普及的年代迅速地占领市场。它支持面向对象程序设计、命令式编程，以及函数式编程。它提供语法来操控文本、数组、日期以及正则表达式等[5]。但其动态类型、基于原型等设计方式则在大型项目中存在明显的短板。动态类型使得开发人员在开发过程中无法获得来自编辑器和编译器的提示，发生错误的时机被推迟到了生产环境，容易造成不可预知的后果，只能通过精确的文档去尽量避免这个问题。加之原型链式设计方法并不是面向对象编程的良好方法，因此非常有必要引入一些更优的方案。

TypeScript便是一款基于JavaScript的编程语言。它通过在JavaScript的基础上添加静态类型定义构建而成。类型提供了一种描述对象形状的方法，可以帮助提供更好的文档和代码提示，可以让TypeScript在开发过程中就验证代码可以正常工作，对于大型工程的代码可读性和可维护性起到强大的作用。而它的类型便不是强制的，而是渐进式的。这允许在编写TypeScript的时候忽略类型推导，并在后续的开发过程中再进行定义与引用，或者直接使用类型推断，即可使用IDE快速地完成类型处理的工作，无需编写额外的代码即可获得大量功能。

TypeScript拥有活跃的社区。很多JavaScript的第三方库都提供TypeScript的类型定义文件。我们可以通过一些数据来了解TypeScript的流行趋势。



图2-2　　JavaScript和TypeScript百度搜索量对比（近10年）

2.2 React

React（也可称React.js或ReactJS），是一个用于构建用户界面的 JavaScript 库[6]。

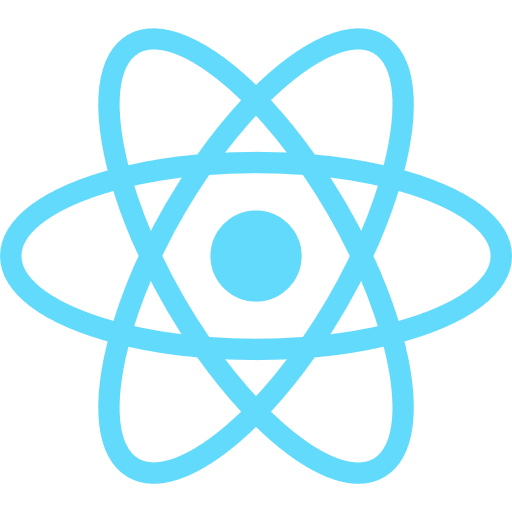


图2-3　　React图标

React具有如下特点：

1. 声明式。相对于命令式编程方法，此编程方法能更轻松地更新组件的状态，当数据改变时能有效地更新并渲染组件。这使代码更加可靠，方便调试。
2. 组件化。可通过React创建拥有各自状态的组件，再由这些组件构成更加复杂的UI。React组件通过使用JavaScript或JSX编写，相比于Vue中使用的模板式写法，可以更轻松高效地控制数据与组件的关联关系与渲染逻辑，并使得状态与DOM分离。
3. 多平台。React不仅可用于网页构建，还可使用Node.js进行服务器渲染，或者使用React Native开发原生移动应用。这使React可实现“一次学习，随处编写”。

React使用了一种新的文件格式“JSX”。这是一种将JavaScript与HTML结合起来构建React组件（后续Vue也通过插件支持了这种格式）的语法。它允许开发者将HTML标签直接作为JavaScript对象使用，亦可在HTML标签中直接穿插JavaScript逻辑，其逻辑类似于Servlet。使用JSX编写的代码经过Babel[7]编译后即可转换成JavaScript代码。这使得开发人员在控制渲染逻辑上拥有了极大的便利性。

React由Facebook创建。起初仅在Facebook内部使用，后来此框架由于太受欢迎，因此于2013年开源。此后快速占据前端主流框架的榜首。

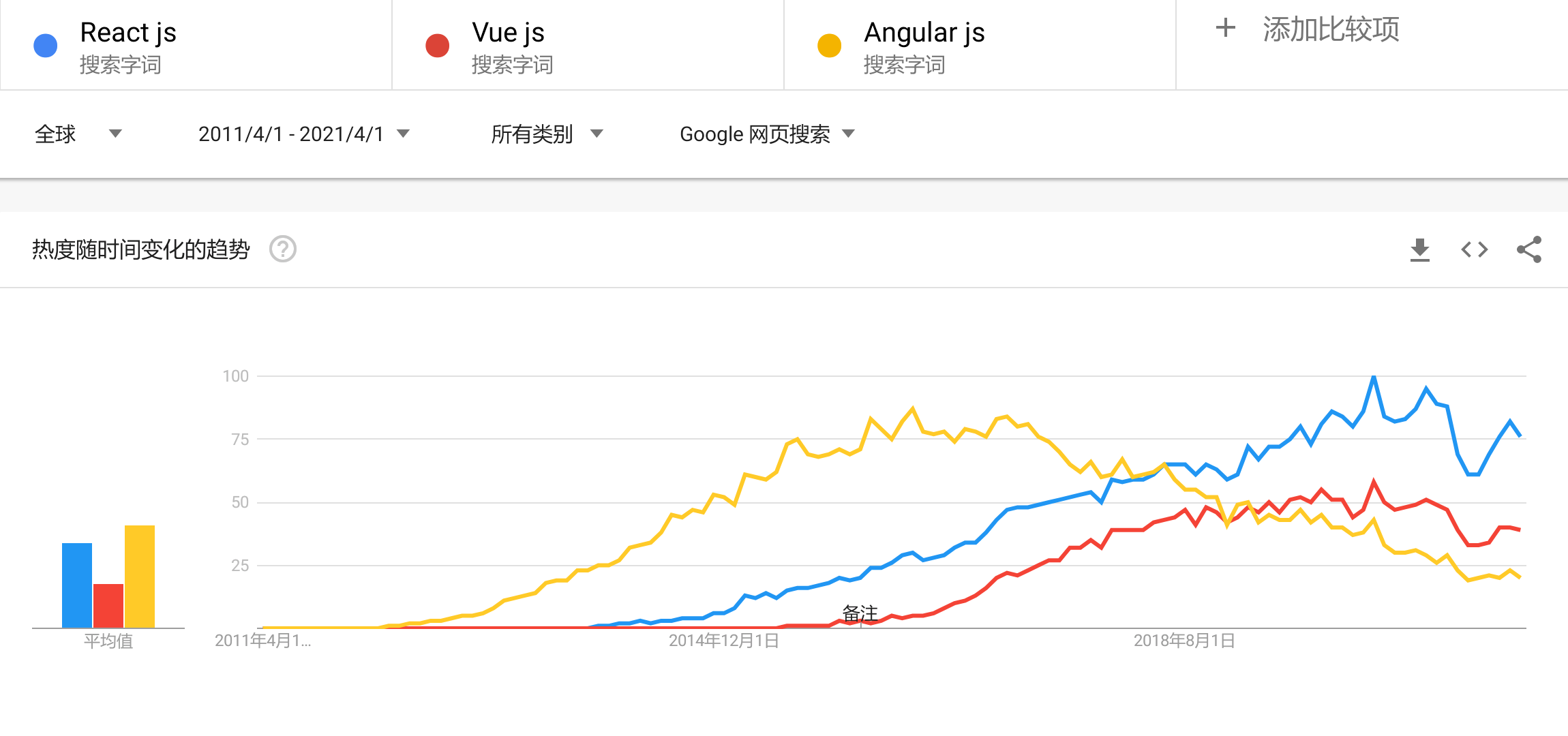


图2-3　　ReactJS、VueJS、AngularJS 谷歌搜索量对比（近10年）

2.3 webpack

webpack是当前前端最热门的前端资源模块化管理和打包工具[8]。

传统的前端脚本引入方式是在DOM中新增script标签，并将其src属性设置为需要引入的JavaScript文件地址。在实际开发中，我们往往会依据模块化的思想，将单个JavaScript文件作为一个模块。而模块与模块之间往往具有较强的依赖性，即模块之间会形成一套依赖链网络。依靠开发者自行计算并导入费时费力，而且每需要一个JavaScript文件便要多进行一次HTTP请求，对性能有影响，因此最佳的方式便是将所有文件都打包起来，一次引入。webpack 的工作便是对JavaScript工程的文件进行打包工作的库。

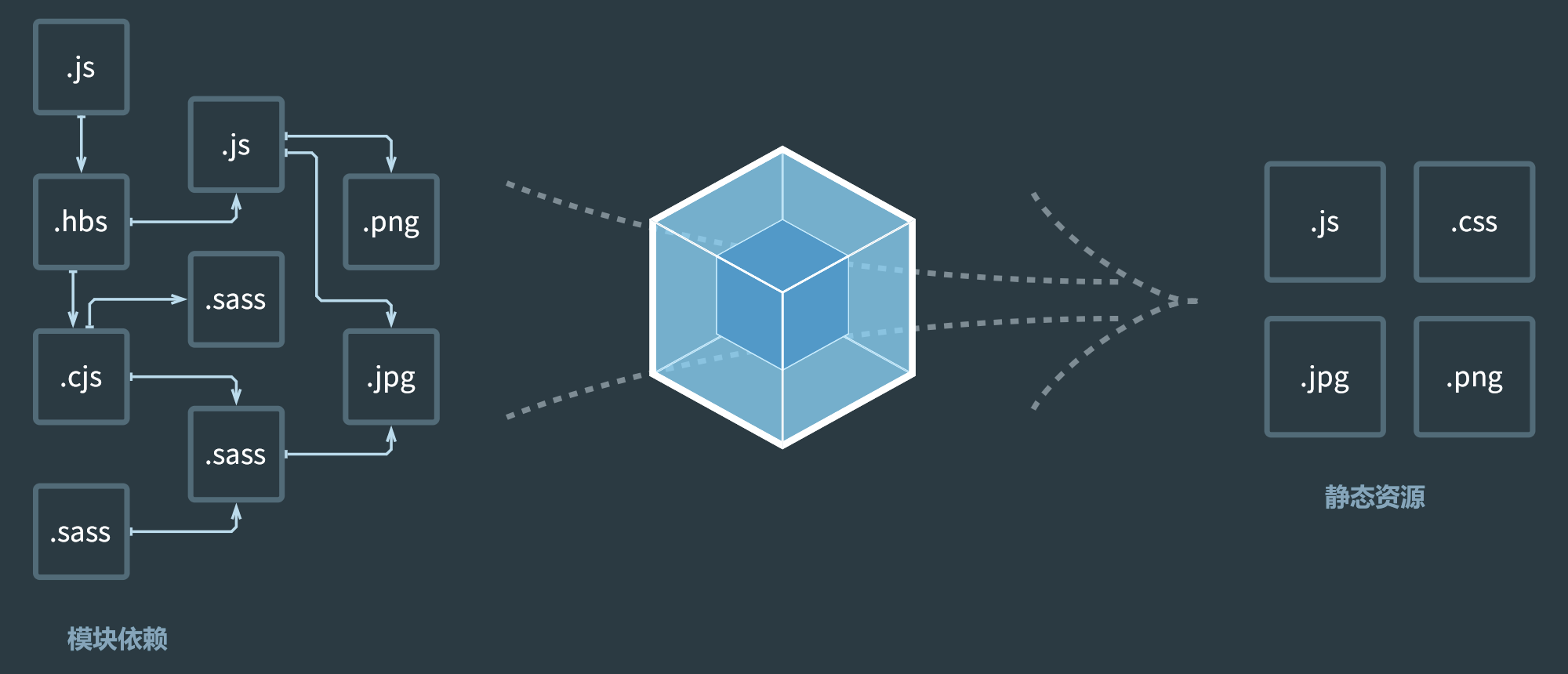


图2-4　　webpack能力示意图

webpack可与其他前端框架高度集成，支持单页面和多页面工程的打包，支持自定义文件格式（或模块）的引入，支持按文件格式分发loader（如TypeScript、Babel、Sass、Stylus等）进行引入，支持插件对代码进行各种后处理，支持生成map文件以供调试，支持热更新，以及更多此处未提及的功能。

这些功能使得一个庞大的工程在浏览器引入之前便被浓缩成少量几个文件，开发人员无需关注模块之间的具体导入顺序，可以如同其他编译型语言一样自由地编写模块；在需要进行调试时，热重载和map文件生成功能可使得本身仅支持JavaScript的开发者工具能支持任意语言的调试，并可在代码改动后在页面内直接生效，无需重载页面。这对于工程开发是极为有利的功能。

webpack现已被众多JavaScript框架选为首选打包工具。如VueJS的vue-cli[9]、ReactJS的create-react-app[10]等。开发人员无需知道JavaScript及框架打包的内部细节，直接使用此工具即可快速地生成对应框架的应用。

2.4 Node.js

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行时[11]。



图2-5　　node.js图标

Chromium中使用的V8引擎V8（谷歌提供的JavaScript引擎）广受大众好评，node.js使V8引擎可以脱离于浏览器运行，实现更多依赖本地的功能，如Web服务器，以及本文需要使用到的本地代码编译。

JavaScript通常被认为是一门解释型的语言，但是现代的JavaScript引擎不再只是解释JavaScript，也会对其进行编译。自2004年Google地图的引入以来，JavaScript已经从一门通常执行几十行代码的语言演变为能在浏览器中运行具有成千上万行代码的完整的应用程序。在这个新世界中，编译JavaScript非常有意义，因为尽管可能需要多花费一点时间来为JavaScript做准备，但是一旦完成，则它会比纯解释型的代码具有更高的性能。上文中提到的TypeScript、React、webpack便是在前端工程工具链中的重要成员，他们相互合作，将开发者的代码在node.js平台上进行编译。可以认为，这是一套使用JavaScript将其他语言编译成JavaScript的平台。

Node.js自带一个包管理器——npm，是一个下载和管理Node.js依赖包的软件，也是一个存放公共包的平台[12]（也可以用于公司的私有软件包访问）。使用npm，可以便捷地连接到Node.js的开源世界，分享和使用成千上万的开源代码。现在npm已成为世界上最大的包管理分发平台，如图2-6所示，其包存有量已远超排名第二的Maven Central。

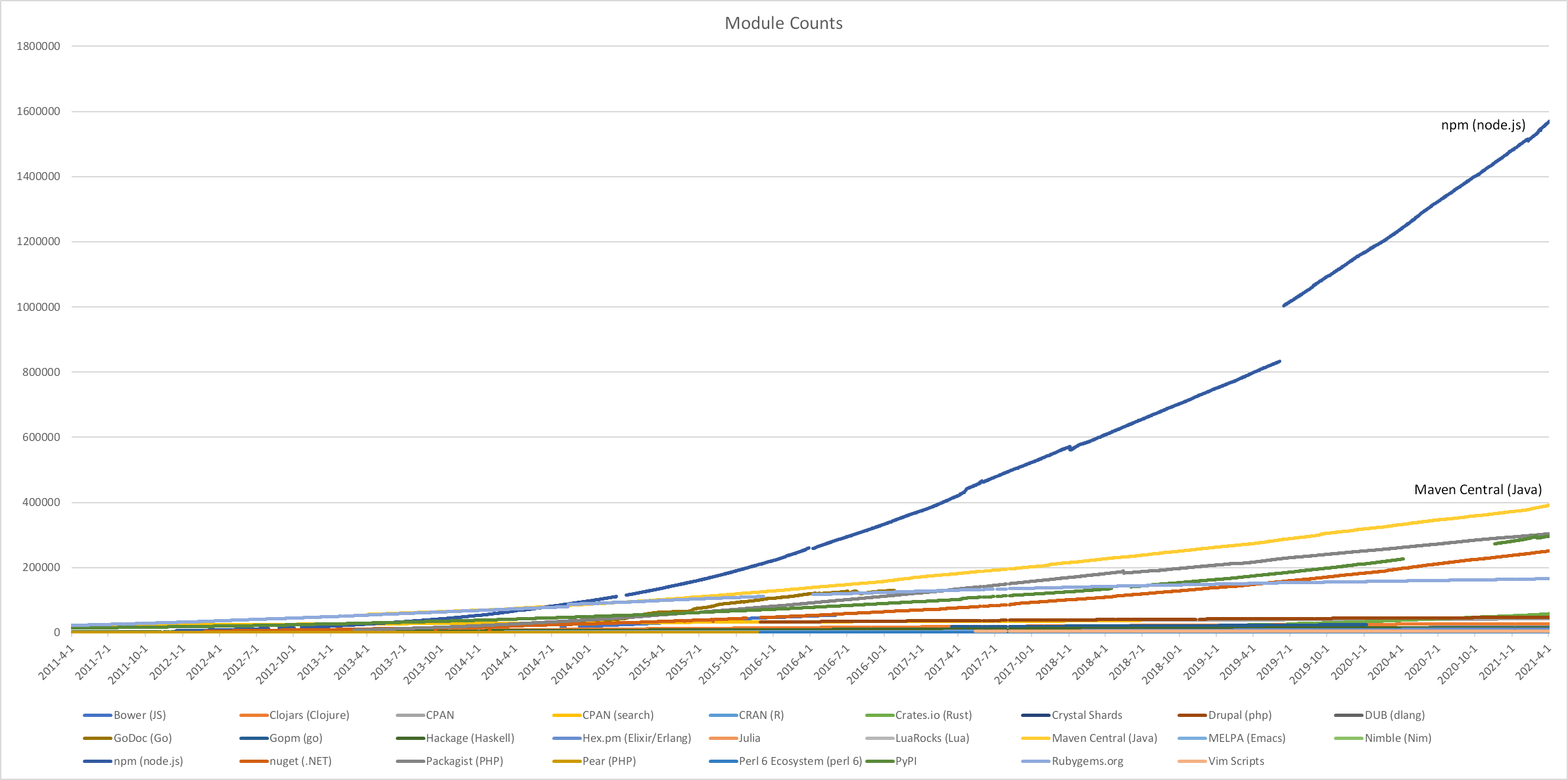


图2-6　　module counts[13]

本工程使用yarn作为包管理工具。yarn同样使用npm仓库，但yarn拥有比npm更好的性能，特别适合作为npm的代替品在大陆地区内使用。

第三章　　系统总体设计

本章首先对课题进行需求分析，通过分析结果对系统提供的主要功能进行了描述，然后从两个角度介绍该系统的整体结构设计，并简要介绍了系统整体运作流程。

3.1 需求分析

算法是编程中重要的一环，而实现优秀的算法往往需要有扎实的编程思想基础。事实上，对于编程思想的教学，在早些时候就已经提前到高中开始进行了。使用过人教版《数学》选修1-2的同学可能已经学过书中“流程图”的相关内容。这里的“流程图”，便是编程思想的基础。流程图编写的思想与其他数学问题的解题思想有明显不同：一般的数学问题需要在题内进行相关知识的发散性、双向性思考，从而得出结论；而流程图的构建则是线性的，构建的是过程，而不是结果。如 ，这个语句在解方程题中可以得出 的结论，而在流程图或编程中，这个语句仅代表将 的值更改为 的操作，即本质上是构建“操作”，而不是直接从语句解算结果。为了方便解释这样的事情，往往需要在草稿纸上记住程序运行的位置、当前的变量状态等信息。

尽管现代的IDE已经具备相当方便的调试功能，可以在程序运行的过程中方便地知道程序的运行位置、调用栈、内存，但这些IDE往往界面复杂，充斥着各种对新人不友好的名词，C语言初学者往往只能靠反复执行printf函数以及笔记画图来获知程序的运行状态，而C语言教学者往往也只能采用类似的方式进行教学。这种操作不仅存在明显的割裂感，也浪费了较多时间进行输出处理。特别是当学习具有具体意义的算法时，依靠IDE的调试功能进行讲解和理解确实会相当枯燥，对于初学者来说实在是一道门槛。

因此，大家希望可以拥有一套系统，能把C语言编程与算法图解联系起来，达到实时演示的目的，方便大家的学习。

3.2 系统功能设计

根据上述需求分析，可确定本系统的以下两个核心功能：源码即时编辑运行功能和算法数据演示功能。其中源码即时编辑运行功能类似于传统IDE提供的代码编辑器及调试功能，支持在网页中编辑源码并单步执行。算法数据演示功能即从代码中提取出变量表等运行状态，并即时反馈到图形演示区中。除此以外还有菜单区，可在此选择进入不同的教案。为方便用户使用，程序主界面将按从左到右的顺序分为三个功能块：菜单区、图形演示区、代码编辑器。

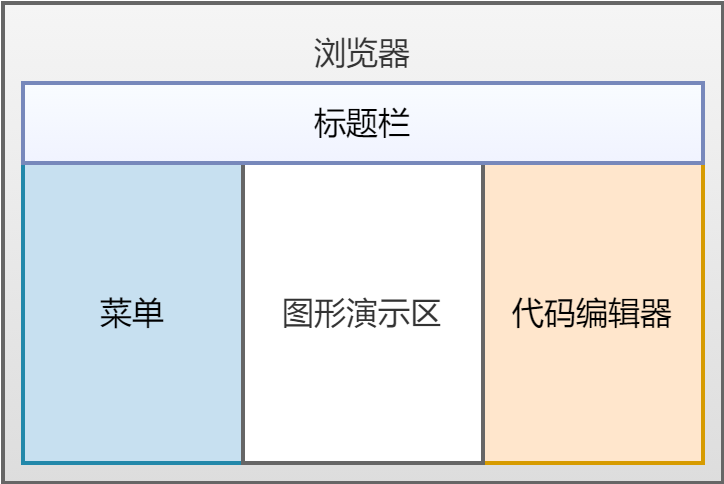


图3-1　　整体界面示意图

下面对各个功能区进行具体描述：

1. 菜单

菜单是进入程序后的第一个主界面，此处采用ExpandableListView的设计方案，将按不同的算法功能类型分章，按相同算法功能类型的不同算法细分节，给用户提供课程。

1. 图形演示区

图形演示区将按照不同的算法实现不同的演示方法。这部分将按一个特定的规则制成不同的演示器，并在选择课程后导入。因此这部分可按实际情况设计成任何样式，并没有特定的界面方案。但不同组件的输入和输出均遵守同一套约定的规则。

1. 代码编辑器

代码编辑器是让用户输入程序代码的地方，教案会根据需求向用户预先提供示例代码，用户也可在此处按教案提示的规则输入或修改C语言代码，代码编辑器也将提供相应的语法高亮功能以辅助用户进行代码编写。此代码编辑器同时具备调试功能，即允许直接运行、单步操作、重新开始等调试器常用操作，并将代码的运行结果实时反馈到图形演示区中，使用户可以直观地看到抽象的代码在演示器中对应的具体的形象。

3.3系统整体结构设计

该系统整体被设计为单个React工程，并且不设计对外调用其他库的功能，因此所有操作均在这个工程中进行。图3-2示意了该系统的模块结构和数据流。我们也可通过工作特征和系统功能两个视角来理解系统的模块分工。

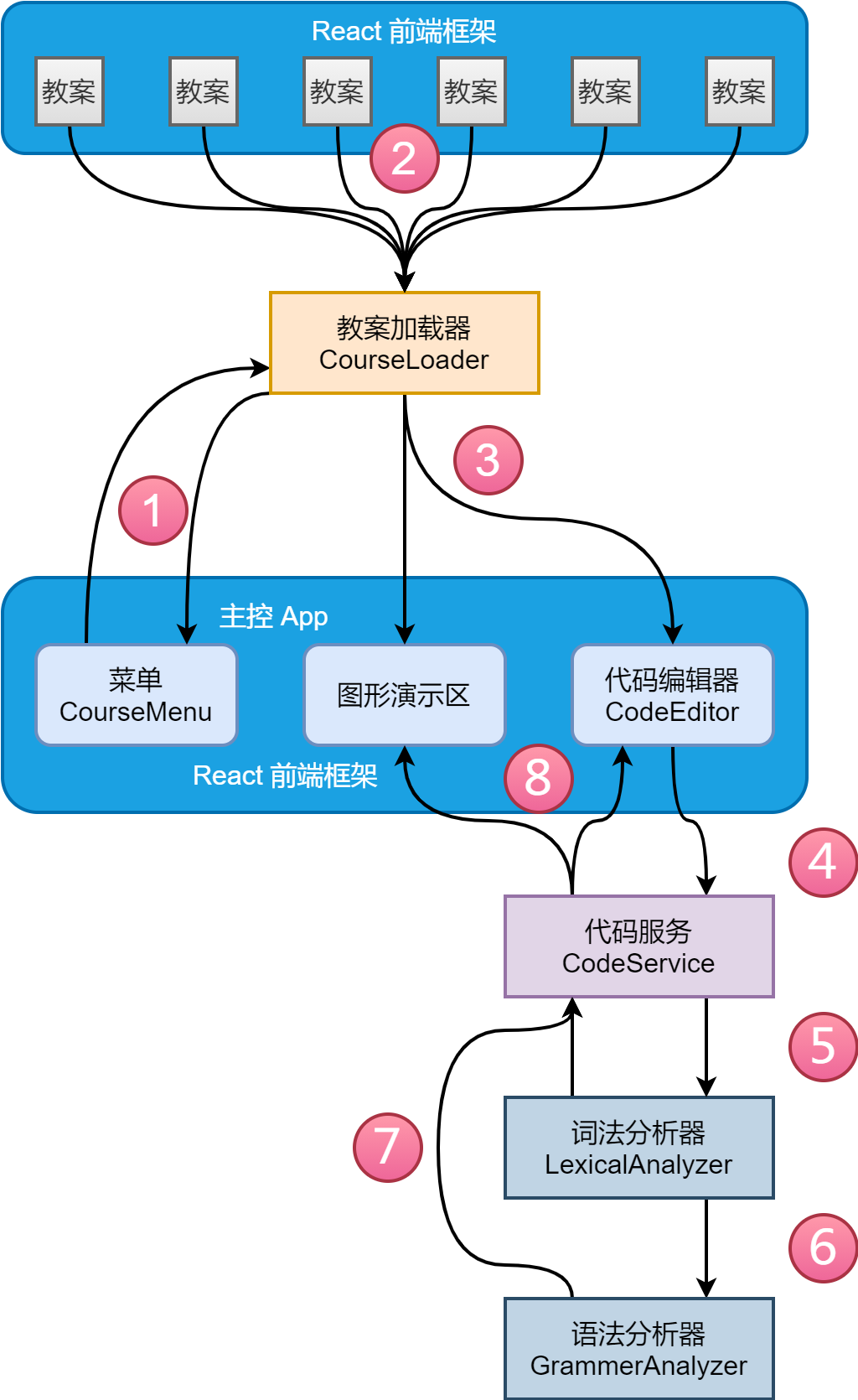


图3-2　　系统整体设计及流程图

从工作特征视角看，本系统可分为前台和后台部分。

1. 前台使用React框架构建，主要负责对用户可见的部分进行渲染，并主控、桥接系统的各个组件及其对应接口。
2. 后台部分主要负责课件的加载、代码的识别和解释执行，并将相应的处理结果返回到前台供展示。

图3-2示意图中被深天蓝色包裹的部分即为前台部分，其余为后台部分。

从系统功能角度看，本系统可分为教案、教案加载器、页面视图、代码服务、编译器五大模块。

1. 教案（灰白色渐变块）被设计成了独立的React模块，可根据不同的教学需求制作不同的教案；每个教案模块向外提供初始代码、预期表量表，向内要求提供程序的实际变量表，也可以依据实际需求添加不同的需求信息，如代码正在运行的行数等，然后通过这些信息渲染图形界面。
2. 教案加载器（淡橙色块）负责提供教案菜单，以及菜单项中对应的教案标题和教案模块。
3. 页面视图（淡蓝色块）负责渲染控制程序的主体功能模块，包括菜单、图形演示区、代码编辑器三个部分。可以认为程序的中心是页面视图主控，而其它模块则为外挂服务。
4. 代码服务（淡紫色块）负责为代码编辑器和编译器提供中间支持，包括将字符串类型的代码转换成相应的内部数据类型、按指定位置或指定方向读取代码的光标服务、根据指令队列执行代码、存储变量表和运行时内存信息等。
5. 编译器（灰蓝色块）负责对C语言程序进行编译操作。它是一个非完整C语言编译器，仅提供C语言的部分核心功能，最终产出的结果仅供该平台进行执行，不输出可执行二进制文件。它包含了词法分析器和语法分析器。通过代码服务提供的功能扫描用户代码，在词法分析器中生成词法单元（token）序列，词法单元序列通过语法分析器生成语法树，并在此过程中进行语法制导翻译和中间代码生成，产出指令队列。这些信息不是互相独立的，而是包含了相关的引用，比如代码中的每个字符在经过词法分析后都会存有到对应词法单元的引用。因此编译器与代码服务息息相关。

图3-2上的数字示意了系统的主要数据流图。从选择课件到代码编写及程序执行的主要数据流如下：

1. 程序载入后，菜单组件自动通过主控向教案加载器请求菜单。
2. 用户选择菜单上的教案后，菜单组件自动加载对应的教案组件。
3. 教案组件自带的示例代码将通过主控传给代码编辑器，其自身则作为一个React组件嵌入到图形演示区中。
4. 代码编辑器在接收到来自主控或者用户发来的代码编辑请求后，将对应动作转发到代码服务中。
5. 代码更新后，对应代码将交给词法分析器进行词法分析处理。这一步将通过词法分析自动机产出代码对应的词法单元序列。
6. 词法单元序列将进一步被送往语法分析器。这一步将通过运用C语言对应CFG的LR分析器产出语法树。并在分析的过程中通过SDT（语法制导翻译方案）中的程序片段进行语法制导翻译和中间代码生成。
7. 上一步最终生成的内容将包括变量表以及每行程序对应的指令队列。这些将被交到代码服务中去。
8. 代码服务中的变量表将被送往图形演示区中的教案。指令队列将在运行程序时被使用。这些指令将在运行时对变量表中的变量产生影响，并最终反馈到图形中。

第四章　　重要模块设计——代码服务

根据需求，本系统将要设计一个平台，让用户输入代码，并在图形演示区中演示对应的数据结构。因此重点便是如何从代码中提取必要信息反馈给图形演示区。为了实现从代码中提取必要信息，需要用到编译技术；为了实现代码编辑器的各种功能，我们需要设计一个代码服务。本章将重点讲解代码服务模块的设计方案。

代码服务（类名CodeService）负责为代码编辑器和编译器提供中间支持，是这两者之间承上启下且不可或缺的角色（在本模块中，“上”被定义为代码编辑器，“下”被定义为编译器）。

4.1 数据结构

对于一般的编译器来说，需要代码服务提供的功能一般只是向前读取、向后读取等基本操作，这种代码服务可以直接使用字符串的方式进行存储。但本系统需要依托代码服务实现的功能较多，因此我们存储代码的基本单位是行。具体表现为这样的一种结构（interface）：

|  |
| --- |
| export interface CodeLine {      code: string;      tokenMap: Array<Token>;      // 待添加功能  } |

每行代码都是一个CodeLine对象。CodeService维护一个Array<CodeLine> 对象用于存放所有代码。对于上层UI来说，这种结构有利于按行显示代码，对于代码执行器来说，这种结构方便了以行为单位执行代码。

CodeLine中有n个属性：code、tokenMap。

code属性以字符串的形式直接存放该行代码。

tokenMap则以字符为单位存储了该行的所有代码的Token引用。当代码有修改时，代码服务将执行词法分析以计算此处的Token。此处Token是另一种自定义结构，将在稍后讲解。

对于编译器来说，它们往往并不关心代码的整行内容，因为换行符对于编译器来说也可以算是一种token。因此代码服务中提供了按字符读取代码的方法，其返回类型是CodeChar。具体表现为这样的结构：

|  |
| --- |
| export interface CodeChar {      char: string;   // 字符      ln: number;     // 行      col: number;    // 列      token: Token;   // 向上对应的词 token  } |

同时也有它的简化版本CodePosition，用于在处理过程中间不需要关心代码本身，只需要关心代码位置的步骤：

|  |
| --- |
| export interface CodePosition {      ln: number;      col: number;  } |

代码编译的第一步是词法分析。这一步将会把代码转换成token序列，即标识符序列。在前面提到的CodeLine和CodeChar中，都有对Token的引用。Token被表示为下面的这种形式：

|  |
| --- |
| export interface Token {      type: TokenType;      value: any;  } |

TokenType即为Token类型，value即为Token的值。这部分内容将在编译器设计中详细讲解。

4.2 方法说明

CodeService具有的公共方法如下所示。

|  |
| --- |
| export interface CodeService {      getCodeLines(): Array<CodeLine>      getAllCode(): string      getCodeLine(ln: number): CodeLine      setCodeLineByString(ln: number, code: string): void      insertCode(content: string, ln: number, col: number): CodePosition  readCharAt\_s(ln: number, col: number): CodeChar      readCharAt(ln: number, col: number): CodeChar      readNextChar(ln: number, col: number): CodeChar      readPrevChar(ln: number, col: number): CodeChar      getCodeToken(ln: number, col: number): Token  } |

getCodeLines用于获取CodeService存储的所有代码行CodeLine。

getAllCode用于获取全部代码，即把所有行都拼接起来输出。

getCodeLine用于获取单个代码行的内容。

setCodeLineByString用于替换单个代码行的内容。

insertCode用于在某个指定的位置插入代码。

readCharAt用于读取指定位置的代码。

readCharAt\_s用于读取指定位置的代码（简化版，不能处理错误位置）。

readNextChar用于读取指定位置的下一个代码。

readPrevChar用于读取指定位置的上一个代码。

getCodeToken用于获取指定位置代码的token。

第五章　　核心模块设计——编译器

编译器用于对C语言程序进行编译操作，以为代码编辑器提供语法高亮、变量表、指令队列等功能。它是本系统的低层核心功能，包含了词法分析器和语法分析器。通过代码服务提供的功能扫描用户代码，在词法分析器中生成词法单元序列，词法单元序列通过语法分析器生成语法树，并在此过程中进行语法制导翻译和中间代码生成，产出指令队列。本章将介绍编译器的相关功能。

5.1 编译原理概述

程序设计语言是向人以及计算机描述计算过程的记号。在一个程序可以运行之前，它首先需要被翻译成一种可以被计算机执行的形式。完成这项翻译工作的软件系统称为编译器（complier）[14]。

简单地说，一个编译器就是一个程序，他可以阅读以某一种语言（源语言）编写的程序，并把该程序翻译成一个等价的、用另一种语言（目标语言）编写的程序[14]。参见图5-1。

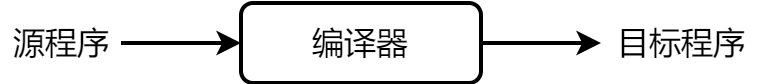


图5-1　　一个编译器

解释器（interpreter）则是另一种常见的语言处理器。它并不生成目标程序，而是利用用户输入直接执行源程序[14]，参见图5-2。

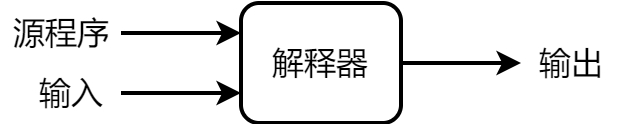


图5-2　　一个解释器

5.2 编译器的结构

通过4.2.1节的介绍，我们可以把编译器认为是一个黑盒子，它能够把源程序转换为目标程序。如果我们稍微把这个黑盒子打开一点，我们就可以看到这个过程实际是由两个部分组成：分析部分和综合部分。

分析（analysis）部分亦称之为编译器的“前端”（front end）。这个步骤将源程序分解成多种组成要素，组合成一种语法结构，然后使用这个语法结构来创建源程序的一种中间表示方法。这个步骤包含了错误检查过程，如果编译器发现源程序没有遵循正确的语法，或者语义上产生错误，它应能提供相应的信息以供用户进行改正。这个步骤往往会收集一些相关的信息，如表量表，或符号表，并将其与中间表示形式一并传送给综合步骤。

综合（synthesis）部分亦称之为编译器的“后端”（back end）。这个步骤根据分析步骤产生的中间表示形式和符号表中的信息构造相应的目标程序。

一些编译器在分析和综合两个步骤之间，还有一个与机器无关的代码优化步骤。它的目的是将中间表示形式上的程序进行一些转换，以达到更优的代码质量，如更小的内存需求、更少的代码数量、更快的执行速度等。

如果我们把一个完整的编译过程的步骤罗列出来，那么编译过程可分为词法分析、语法分析、语义分析、中间代码生成、机器无关代码优化、目标代码生成、机器相关代码优化。如图5-3所示。其中词法分析、语法分析、语义分析、中间代码生成为前端部分，目标代码生成和机器相关代码优化器为后端部分。

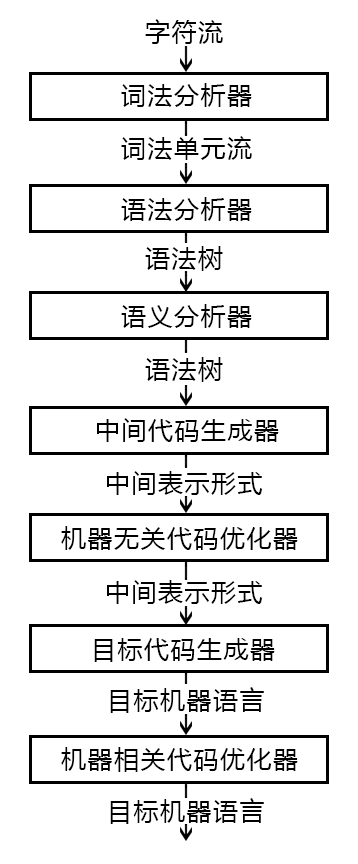


图5-3　　完整编译过程的各个步骤

在本系统中，我们使用编译器的主要的目的并不是为了生成出一个可执行程序，而是为了获取符号表和中间代码（在本系统中称为“变量表”和“指令队列”），并对其进行解释执行。再者，该系统的目的是进行编程教学，因此我们可以忽略编译过程的很多步骤，仅需实现编译器的前端，而不需要通过编译器后端产出可执行程序。

我们可以用Java语言来类比本系统的编译操作。Java语言处理器结合了编译和解释过程，如图5-4所示[14]。Java源程序首先被编译成称为“字节码”（bytecode）的中间表示形式，然后由一个虚拟机对得到的字节码进行解释执行。它明确地拆出了编译过程的两大步骤，将机器无关的步骤前置进行，机器相关的步骤后置进行，使Java程序拥有了良好的可移植性。本系统包含了编译器前端和解释执行器两部分，由两个不同的模块负责，均在同一工程中进行。



图5-4　　一个混合编译器

5.3 语言及其文法基本概念

在讲述编译器的基本编译步骤前，我们需要了解语言及其文法的基本概念。这会帮助我们理解5.4节词法分析中“自动机”的部分，以及5.5节语法分析中的内容。

5.3.1 字母表

字母表是一个有穷符号集合。其中“符号”可以被定义为字幕、数字、标点符号等。一些常见的字母表比如二进制字母表、ASCII字符集、Unicode字符集等。

字母表可以进行运算。

字母表和的乘积被定义为，如。

字母表的n次幂被定义为，即长度为n的符号串构成的集合。如。

字母表的正闭包被定义为，即长度正数的符号串构成的集合。如。

字母表的克林闭包被定义为。如。

设是一个字母表，，x称为是上的一个串。串是字母表中符号的一个有穷序列。串的长度，通常记作，是指中符号的个数。空串是长度为0的串，用表示。如：，。

串也可以进行运算。

串的连接即把两个串直接连接起来。例如，如果且，那么。对任何空串，都有。

串的幂运算被定义为。例如，如果，则，，，……

5.3.2 文法的定义

我们为文法提出的形式化定义为：

为终结符集合。终结符是文法所定义的语言的基本符号，有时也称为token。如：

为非终结符集合。非终结符是用来表示语法成分的符号，有时也称为“ 语法变量”如：

为产生式集合。产生式描述了将终结符和非终结符组合成串的方法。如

为开始符号。开始符号表示的是该文法中最大的语法成分。如。

依据上述定义，我们可以给出一些文法定义的例子，如

|  |
| --- |
| G =( { id, +, \*, (, ) }, {E}, P, E )  P ={ E → E + E ,  E → E \* E ,  E → ( E ) ,  E → id } |

为了简便起见，在不引起歧义的前提下，文法定义可以只写产生式，如：

|  |
| --- |
| G : E → E + E  E → E \* E  E → ( E )  E → id |

对一组有相同左部的产生式，可以使用竖杠进行合并操作，如：

|  |
| --- |
| E → E + E | E \* E | ( E ) | id |

一般情况下，约定下属符号是终结符：⑴字母表中排在前面的小写字母，如a、b、c。⑵运算符，如+、\*等。⑶标点符号，如括号、逗号等。⑷数字0、1、…、9。⑸粗体字符串，如**id**、**if**等。

一般情况下，约定下属符号是非终结符：⑴字母表中排在前面的大写字母，如A、B、C。⑵字母S。通常表示开始符号。⑶小写、斜体的名字，如*expr*、*stmt*等。⑷代表程序构造的大写字母。如E(表达式)、T(项)和F(因子)。

其余的约定俗成的表示方法：⑴字母表中排在后面的大写字母（如X、Y、Z）表示文法符号（即终结符或非终结符）。⑵字母表中排在后面的小写字母（主要是u、v、. . . 、z表示终结符号串（包括空串）。⑶小写希腊字母，如α、β、γ，表示文法符号串（包括空串）。⑷除非特别说明，第一个产生式的左部就是开始符号。

5.3.3 文法的分类

文法一般被分为4种类型：0型文法、1型文法、2型文法、3型文法。

0型文法又称无限制文法（Unrestricted Grammar）或短语结构文法（Phrase Structure Grammar, PSG）。它的定义是。

1型文法又称上下文有关文法（Context-Sensitive Grammar, CSG）。它的定义是。使用1型文法的产生式的一般形式为：。

2型文法又称上下文无关文法（Context-Free Grammar, CFG）。它的定义是。使用2型文法的产生式的一般形式为：。

3型文法又称正则文法（Regular Grammar, RG）。正则文法分为左线型文法（）和右线性文法（）。

4个级别的文法存在逐级包含和逐级限制规则。从0型文法到3型文法，每一级别的文法限制性都更大。他们之间的关系可以用图5-5表示。

图5-5　　各级文法的包含关系

5.3.4 CFG的分析树

正则文法可以用来描述程序设计语言中的大多数单词，但是，它的生成能力有限，它难以用于描述程序设计语言中的句子构造；而上下文无关文法可以描述大部分程序设计语言中的语法构造，也是被研究得最多的一种文法。对于CFG文法，我们往往使用分析树来表示对语言的应用。

分析树是句子结构的一种图形化表示，也可以用来描述推导或归约的过程。

在分析树中：

根节点的标号为文法开始符号。

内部节点表示对一个产生式的应用。

叶节点的标号既可以是非终结符，也可以是终结符。从左到右排列叶节点得到的符号串称为是这棵树的产出或边缘。

给定一个句型，其分析树中的每一棵子树的边缘称为该句型的一个短语。如果子树只有父子两代结点，那么这棵子树的边缘称为该句型的一个直接短语。

比如，对于以下文法：

|  |
| --- |
| ① E → E + E  ② E → E \* E  ③ E → ( E )  ④ E → id |

句子“id+(id\*id)”的分析树如图5-6所示：

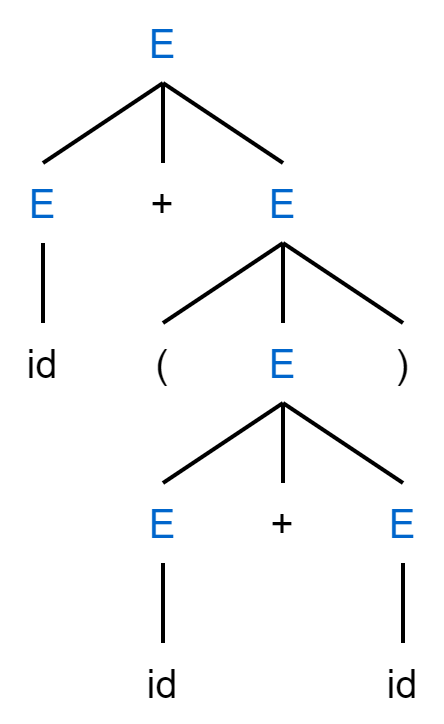


图5-6　　一颗分析树

5.4 词法分析

编译的第一个步骤称为词法分析（lexical analysis）。词法分析器的主要任务是读入源程序的字符流，并将其组织成词法单元序列作为输出。词法分析器通常还要和符号表进行交互，当它发现了一个标识符的词素时，需要将这个词素添加到符号表中。

词法单元可以使用如下写法来表示：

<type, value>

比如我们要处理这样一个语句：

position = position + increment \* 16

经过词法分析器后，将会输出如表5-1的词法单元序列：

表5-1　　position = position + increment \* 16 对应的词法单元序列

|  |  |
| --- | --- |
| **Type** | **value** |
| Identifier | “position” |
| Assign |  |
| Identifier | “position” |
| add |  |
| Identifier | “increment” |
| Multiply |  |
| number\_int | 60 |

为了将程序语句转换成词法单元序列，我们需要使用有穷自动机。

5.4.1 有穷自动机

有穷自动机（Finite Automata, FA）是由两位神经物理学家MeCuloch和Pitts于1948年提出的，对一类处理系统建立的数学模型。这类系统具有一系列离散的输入输出信息和有穷数目的内部状态。系统只需要根据当前所处的状态和当前面临的输入信息就可以决定系统的后继行为。每当系统处理了当前的输入后，系统的内部状态也将发生改变。

有穷自动机的定义为。其中表示有穷状态集，表示输入符号集合，即输入字母表。表示转换函数。表示开始状态，表示接收状态。

有穷自动机通常用转换图来表示。每个结点表示一个状态，状态之间的转换使用带标记的有向边表示。开始状态只有一个，一般由“start”箭头指向。接收状态可以有多个，一般用双圈表示。

例如，一个只接收“a”和“b”两种字符，并且以“abb”结尾的语言，其对应的一种自动机可使用如图5-6的转换图表示。

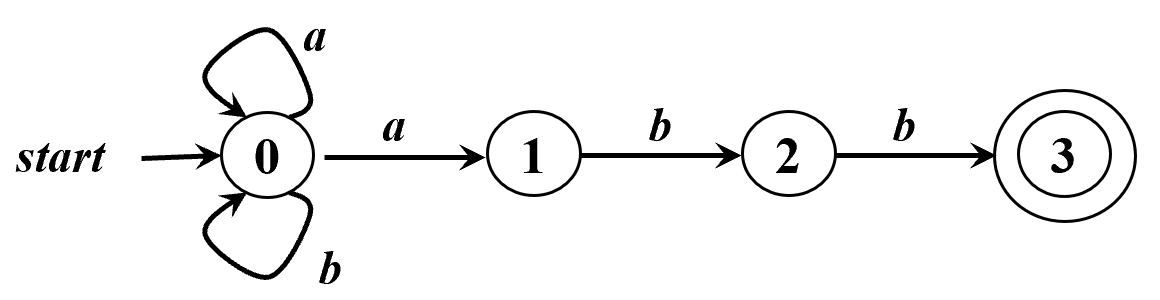


图5-6　　一个状态转换图

图5-6所示的是其中一种可接收上述语言的自动机的状态转换图。事实上，可以接收这个语言的自动机不止一种。一个自动机对应的是一种语言，但一种语言可以对应多个自动机。

自动机一般分为确定的自动机（Deterministic finite automata, DFA）和非确定的自动机（Nondeterministic finite automata, NFA）。我们可以简单地认为，DFA和NFA之间的区别主要在于转换函数。DFA的转换函数目标是一个状态，而NFA的转换函数目标是一个状态集合。并且，NFA的转换边可以包含。

DFA和NFA是可等价的，即这两种状态机可以识别相同的语言，这两者也可通过一定的方法进行转换。

5.4.2 使用DFA进行C语言词法分析

本系统在进行词法分析时，使用了DFA。所使用的DFA对应的状态转换图如图5-7所示。

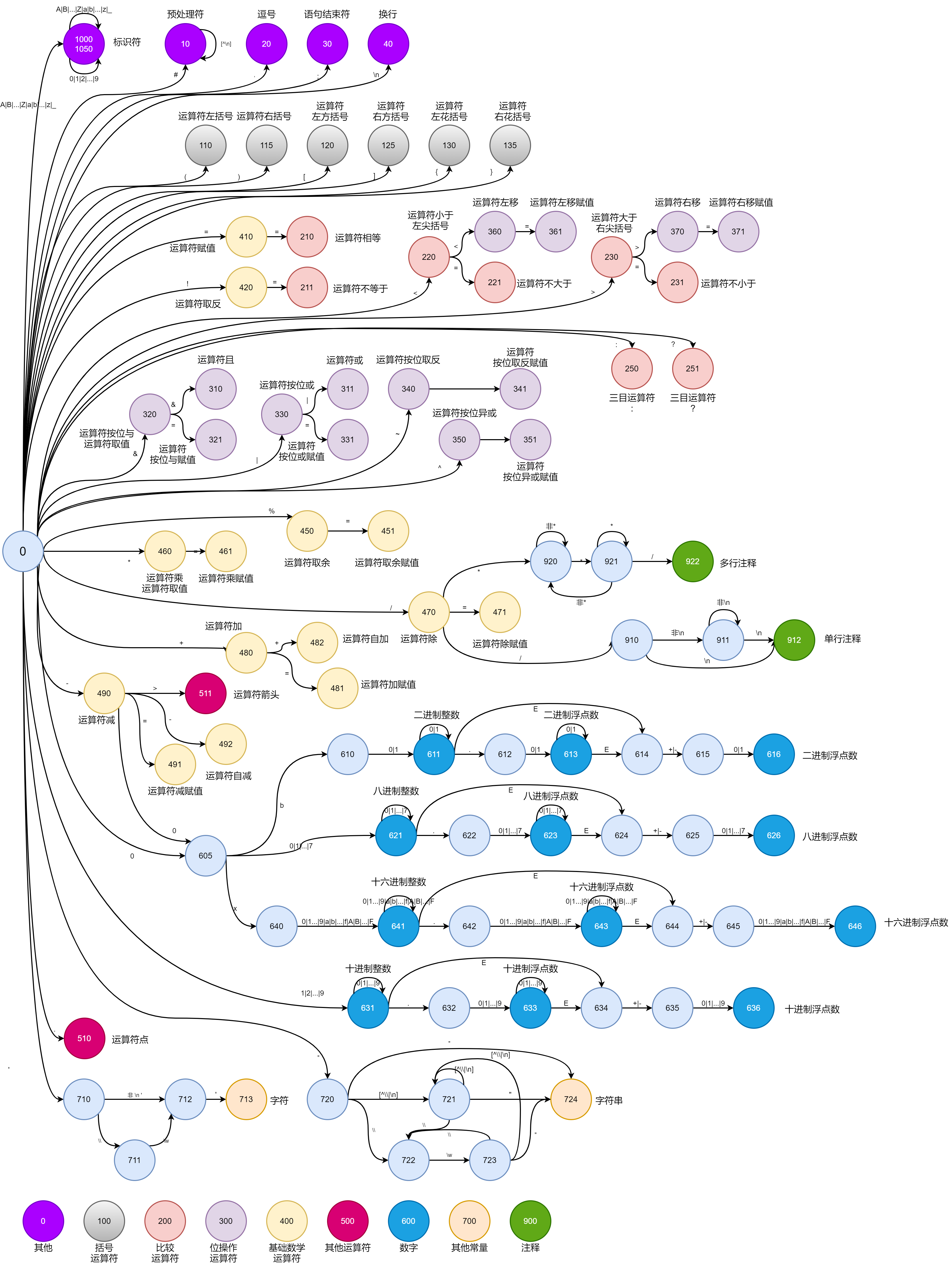


图5-7　　C语言DFA

为了方便绘图，该状态转换图对标准定义的绘图方法进行了一定程度上的修改，见下述说明。

图中省略了“start”标识，规定0为DFA的开始状态。

所有非终结符均用淡天蓝色圆来表示。

所有终结符均用非淡天蓝色圆来表示，并且按照一定的分类模式对其进行了颜色标记，详见状态转换图底部的图例说明。

转移边的转移规则大体上使用正则表达式表示（正则表达式是一种使用正则文法的语言匹配方法，受篇幅限制，此处不进行其他描述，读者可自行查阅相关资料）。为了表述方便，进行了一些变形，如使用“非\n”表示可接收除了换行符以外的所有字符。转义字符使用JavaScript的转义符规则表示。

圆上的数字除了表示状态值以外，在终结符上还同时表示词法单元“type”枚举上的一个值。

当DFA进入“1000”状态时，表示接收到的单词可能是标识符，也可能是关键字。此时会进行一些额外分析，查找关键字是否包含该接收到的单词。如果有，则将其更改为对应的“type”枚举值。

DFA使用一个循环来读入程序中的每个字符。在一轮识别开始前，程序会先记录开始位置，然后开始走自动机。当DFA在某个状态接收到字符，但并没有与该字符在该状态下对应的转移边，则表示遇到了异常。此时DFA会检查遇到异常前的状态，如果是终结符，那么表示识别成功，并将前面识别的内容（从刚才标记的开始位置）全部标记为该终结符对应的词法单元类型枚举值，然后开始新一轮识别。如果不是终结符，则表示识别失败，出现了语法错误，将其类型标记为“错误”。

值得注意的是，本系统并没有完全实现C语言的所有语法特性。事实上，考虑到初学者往往不会使用太多复杂的语法，此DFA进行了较多的省略。同时，该词法分析器在获取词法单元值的时候，并不会按C语言的规则记录该变量的细节信息（如内存宽度等），而是按照JavaScript的类型系统进行记录。

词法分析服务由代码服务启动，并从代码服务中取得向后读取、向前读取等相关功能。一般情况下，在代码发生变化之后，代码服务会将此操作经过节流[15]处理后，交由词法分析服务进行分析。

5.5 语法分析

编译的第二个步骤称为语法分析（grammar analysis）。语法分析器的主要任务是从词法分析器获取一个由词法单元组成的串（词法单元序列），并验证这个串可以由源语言的文法生成，并同时构造出一颗语法分析树，并把它传递到编译器的其余部分进行进一步处理。事实上，语法分析器可能并不需要显式地构造出语法树，因为语法制导翻译和中间代码生成步骤可以在语法分析的过程中交错完成，因此，语法分析器和编译器前端的其他部分可以使用一个模块来实现。

编译器中常用的语法分析器分析方法可以分为自底向上和自顶向下的。顾名思义，自底向上的语法分析方法从分析树的底部（叶节点）向顶部（根节点）方向构造分析树，可以看成是将输入串w归约为文法开始符号S的过程；而自顶向下的语法分析方法从分析树的顶部（根节点）向底部（叶节点）方向构造分析树，可以看成是从文法开始符号S推导出词串w的过程。而这两种分析方法，语法分析器总是按从左到右的顺序扫描输入程序，每次扫描一个符号（词法单元）。

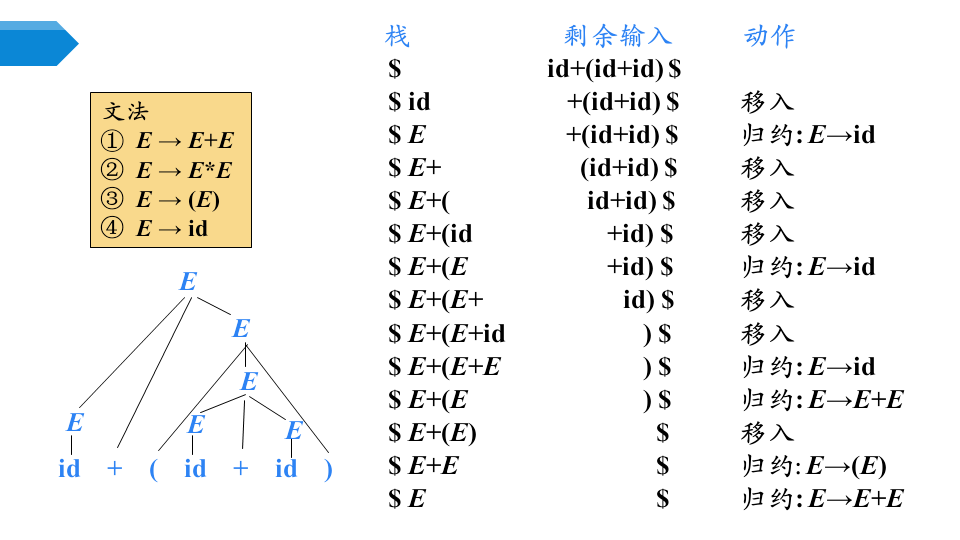
一般认为，自底向上的方法往往用于从句子识别语言，而自顶向下的方法往往用于从语言生成句子。因此本系统使用自底向上方法进行编程语言的识别，自顶向下方法此处不做讲解。

5.5.1 移入-归约语法分析技术

移入-归约语法分析是自底向上语法分析的一种形式。它使用一个栈来保存文法符号，并使用一个输入缓冲区来存放即将进行分析的其余符号。

在一次分析过程中，语法分析器将零个到多个输入符号移到栈顶，如果栈顶的符号可以进行归约，则进行归约。语法分析器将不断重复这个循环，直到它检查到了语法错误，或者栈中包含了开始符号，并且输入缓冲区为空为止。

5.3.4节提到的文法及使用该文法的一个句子的例子可使用移入-归约技术进行语法分析，过程如图5-8所示。



5-8　　一个移入-归约分析的例子

5.5.2 LR分析法

目前常用的自底向上语法分析器都基于LR(k)语法分析法的概念。其中“L”表示对输入进行从左到右的扫描，“R”表示反向构造出一个最右推导序列，“k”表示需要向前查看k个符号输入，一般情况下k≤1。

LR分析器实质上也是一种自动机。它具有两个栈：状态栈和符号栈；具有三种状态：移进、待约、归约；并具有动作表（ACTION）和转移表（GOTO），用于提供自动机的状态转移函数。

LR分析器在从左到右分析的过程中，每次过程都从剩余输入中读取一个符号，并且在ACTION表中寻找当前状态与当前输入匹配的动作。如“s1”表示使自动机跳转到状态4。然后，检查当前是否可以归约，即在ACTION表或GOTO表中将当前状态与当前输入匹配的操作，如“r1”表示使用文法规则的第1条进行归约操作。当最终操作为“acc”时，表示归约操作完成，如果此时剩余输入为空，那么分析成功。

比如，有这样一个文法：

|  |
| --- |
| **① *S*→*BB***  **② *B*→*aB***  **③ *B*→*b*** |

它的一个LR分析表如下表所示：

表5-1　　一个LR分析表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **状态** | **ACTION** | | | **GOTO** | |
| **a** | **b** | **$** | **S** | **B** |
| **0** | S3 | S4 |  | 1 | 2 |
| **1** |  |  | acc |  |  |
| **2** | S3 | S4 |  |  | 5 |
| **3** | S3 | S4 |  |  | 6 |
| **4** | R3 | R3 | R3 |  |  |
| **5** | R1 | R1 | R1 |  |  |
| **6** | R2 | R2 | R2 |  |  |

为了构造LR分析表，我们需要使用LR(1)分析技术。

5.5.3 C语言识别文法

本系统实现了一个简化版C语言的自底向上分析器。使用了如下语言来描述C语言[16]：

表5-2　　本系统的简化版C语言LR文法

|  |  |  |
| --- | --- | --- |
|  | **产生式左部** | **产生式右部** |
| 1 | <程序> | <函数定义闭包> |
| 2 | <函数定义闭包> | <函数定义> <函数定义闭包> | $ |
| 3 | <函数定义> | <类型> id ( <参数声明> ) { <函数块> } |
| 4 | <类型> | <基本类型> | <类型> [ number ] |
| 5 | <基本类型> | int | long | float | double | char | void |
| 6 | <表达式> | <项> <项闭包> | ( <项> ) <项闭包> |
| 7 | <项闭包> | <运算符> <表达式> |
| 8 | <项> | id | number | string |
| 9 | <运算符> | + | - | \* | / | % |
| 10 | <参数声明> | <声明语句> <参数声明闭包> | $ |
| 11 | <声明语句> | <类型> id <赋初值> | id <赋初值> |
| 12 | <赋初值> | = <表达式> | $ |
| 13 | <参数声明闭包> | , <声明语句> <参数声明闭包> | $ |
| 14 | <函数块> | <声明语句闭包> <函数块闭包> |
| 15 | <声明语句闭包> | <声明语句> <声明语句闭包> | $ |
| 16 | <函数块闭包> | <赋值语句> <函数块闭包> | <while循环> <函数块闭包> | <条件语句> <函数块闭包> | <函数返回> <函数块闭包> | $ |
| 17 | <赋值语句> | id <赋值或函数调用> |
| 18 | <赋值或函数调用> | = <表达式> ; | ( <参数列表> ) ; |
| 19 | <参数列表> | <参数> <参数闭包> |
| 20 | <参数闭包> | , <参数> <参数闭包> | $ |
| 21 | <while循环> | while ( <逻辑表达式> ) { <函数块> } |
| 22 | <条件语句> | if ( <逻辑表达式> ) { <函数块> } <否则语句> |
| 23 | <否则语句> | else { <函数块> } | $ |
| 24 | <函数返回> | return <因式> ; |
| 25 | <逻辑表达式> | <表达式> <比较运算符> <表达式> |
| 26 | <比较运算符> | < | > | == | >= | <= |

待补充。

第六章　　其余模块设计

从系统整体设计章节，该系统被大致分成五个模块：教案、教案加载器、页面视图、代码服务、编译器。本章将讲述教案、教案加载器、页面视图三个模块的设计。

5.1 教案

教案，本系统中亦称呼“课程”，被设计成了独立的React模块，可根据不同的教学需求制作不同的教案；每个教案模块向外提供初始代码、预期变量表，向内要求提供程序的实际变量表，也可以依据实际需求添加不同的需求信息，如代码正在运行的行数等，然后通过这些信息渲染图形界面。

所有教案均继承自BaseCourse抽象对象。该抽象对象自身继承自React.Component，并在此基础上实现了非抽象方法componentDidMount，要求课程对象实现getBaseCode、getListeningVariable、onVariableChanged、render四个方法，相当于一个增加了功能和限制的React组件。只要使用React框架的编程方法即可轻松地为本系统添加课程。

|  |
| --- |
| export abstract class BaseCourse<Props = {}, State = {}> extends React.Component<Props, State> {      // 调用父组件传入的 onRef（要求父组件必须有此 prop）      componentDidMount(): void {          (this.props as any).onRef(this);      }      abstract getBaseCode(): string;  // 示例代码      abstract getListeningVariable(): Array<ListeningVariable>;  // 获取需要在代码编辑器监听的变量      abstract onVariableChanged<T>(changedVariable: ChangedVariable<T>): void;   // 变量发生变化时由父组件调用      abstract render(): JSX.Element;  // render 函数  } |

5.1.1 各算法教案待补充。

5.2 教案加载器

教案加载器（淡橙色块）负责提供教案菜单，以及菜单项中对应的教案标题和教案模块。

每个“教案”都是一个CourseInfo对象。该对象的定义如下，其中title表示教案（节）标题，course表示BaseCourse教案模块。

|  |
| --- |
| export interface CourseInfo {      title: String;      course: React.ComponentClass<any, any>; // | typeof BaseCourse  } |

教案加载器自身硬编码了一个CourseMenu对象。该对象是一个ChapterInfo列表，存储了教案菜单以及CoueseInfo引用。ChapterInfo定义如下，其中title表示章标题，sections表示该章包含的节（教案）。

|  |
| --- |
| export interface ChapterInfo {      title: String;      sections: Array<CourseInfo>;  }; |

外界通过传入课程序号的方式获取相应的教案。教案不需要一个id进行区分，仅需要序号即可，因为主控在初始化程序时就会要求教案加载器返回课程列表。

课程序号CourseIndex被定义如下，其中chapter为章序号，section为节序号。

|  |
| --- |
| export interface CourseIndex {      chapter: number;      section: number;  } |

主控在接收到用户点击课程的事件后，就会调用教案加载器的相应函数，获取一个React组件元素，可直接应用到JSX中。

5.3 页面视图

页面视图（淡蓝色块）负责渲染控制程序的主体功能模块，包括菜单、图形演示区、代码编辑器三个部分。可以认为程序的中心是页面视图主控，而其它模块则为外挂组件。

程序启动后，首先会从index.tsx启动。该tsx引入了./containers/App.tsx，并以App作为ReactDOM的渲染入口，并将其挂载到HTML root元素上。

App组件初始化完成后，就会由React框架调用render函数，并在render函数中初始化菜单。图形演示区和代码编辑器则在选中教案之后才会开始渲染，并且由于选中教案后只能随时更换教案，而不能取消选择，因此无需考虑组件卸载。

待补充。

第负一章　　模板

□□☆☆☆☆☆☆☆☆☆正文*（小四号、宋体） 宋体五号，居中，位于表上*

表1-3□□☆☆☆

|  |  |  |
| --- | --- | --- |
| ×××  ××× | ××× | ××× |
| ××× | ××× （*宋体五号，垂直居中）* | ××× |
| ××× | ××× | ××× |

*（表与正文空一行）*

□□☆☆☆☆☆☆☆☆☆☆☆☆☆

*（下一章另起一页）*

第二章□□☆☆☆☆☆*（居中、小三号、黑体）*

2.1☆☆☆*（四号、黑体、顶格）*

2.1.1☆☆☆*（四号、黑体、顶格）*

□□☆☆☆☆☆☆☆☆☆正文*（小四号、宋体、空2格、用1.25倍行间距）*

总经理

经营科

后勤中心

销售科

质量检测科

设备科

副总经理

秘书室

办公室

图2-5□□**×××**组织结构图

*宋体五号居中，位于图下*

*图与下文空一行*

*正文开始标注页码;位置：页面底端（页脚）;*

1

*对齐方式：居中*

参考文献（四号、黑体、顶格）

[1]　　TIOBE Index [网]．https://www.tiobe.com/tiobe-index/

[2]　　TypeScript中文网 [网]．https://www.tslang.cn/index.html

[3]　　网景 - 维基百科 [网]．https://zh.wikipedia.org/wiki/%E7%B6%B2%E6%99%AF

[4]　　TC39 – Ecma International [网]．https://www.ecma-international.org/technical-committees/tc39/

[5]　　JavaScript - 维基百科 [网]．https://zh.wikipedia.org/wiki/JavaScript

[6]　　React官方中文文档 [网]．https://zh-hans.reactjs.org/

[7]　　Babel中文网 [网]．https://www.babeljs.cn/

[8]　　webpack [网]．https://webpack.docschina.org/

[9]　　Vue CLI [网]．https://cli.vuejs.org/zh/

[10]　　facebook/create-react-app - github [网]．https://github.com/facebook/create-react-app

[11]　　Node.js中文网 [网]．http://nodejs.cn/

[12]　　About npm | npm docs [网]．https://docs.npmjs.com/about-npm

[13]　　Module counts [网]．http://www.modulecounts.com/

[14]　　Alfred V. Aho *et al* [美]，赵建华、郑滔、戴新宇译．编译原理 第2版[M] ．机械工业出版社，2009.1

[15]　　浅谈 JS 防抖和节流，安歌 [网]．https://segmentfault.com/a/1190000018428170

[16]　　- [网]．https://github.com/phenomena9/-

[13]　　Module counts [网]．http://www.modulecounts.com/

[13]　　Module counts [网]．http://www.modulecounts.com/

[13]　　Module counts [网]．http://www.modulecounts.com/

[13]　　Module counts [网]．http://www.modulecounts.com/

[1]　　TIOBE Index [J]．https://www.tiobe.com/tiobe-index/

*作者姓名*　　*论文题目　　　　　杂志名称*　*出版年份\页码（阿拉伯数码）*

[2]□□王连芬.层次分析法引论[M].北京:中国人民大学出版社,1990.

*作者姓名 书名 出版地 出版者 出版年*

[3]□□Joe Tidd[×]，陈劲译.创新管理----技术、市场与组织变革的集成[M].清华大学出版社,2002.

*原著者姓名 国别 翻译者姓名 书名 出版者 出版年*

[4]□□姚光起．一种氧化锆材料的制备方法[P]．中国专利，891056088，1980-07-03.

*申请者 专利名 国家 专利号 授权日期*

[5]□□Pamela M.Franklin,Catherine P.Koshland,Donald Lucas,et al,Evaluation of combustion by-products of MTBE as a component of reformulated gasoline [J], Chemosphere, 2001, 42:861-872.

*（以上，如果需要两行的，第二行文字要位于序号的后边，与第一行文字对齐。中文的用五号宋体，外文的用五号Times New Roman字体。）*