华北电力大学

课程设计报告

（ 2022 - 2023 年度第 1 学期）

名　　称：编译技术课程设计

题　　目：L 语言编译器的设计与实现

院　　系：计算机系

班　　级：软件 2001 班

学　　号：220201090123

学生姓名：杨照

指导教师：鲁斌　李继荣

设计周数：1　周

成　　绩：

日期：　2022 年 11 月 25 日

《编译技术》课程设计

任　务　书

一、 目的与要求

1. 任务：实现一个简单的编译程序，能够对指定程序设计语言进行编译。

2. 目的：加深对课堂讲授知识的理解，熟练掌握编译程序设计原理及常用的技术，建立编译程序的整体概念，使得学生初步具有研究、设计、编制和调试编译程序的能力。

3. 要求：熟悉有关定义、概念和实现算法，设计出程序流程框图和数据结构，编写出完整的源程序，进行静态检查，设计出输入数据、显示输出数据；基本功能完善，方便易用，操作无误；通过课程设计学会编译程序设计与实现的常用技术，具备初步分析、设计和开发编译程序的能力，具备分析与检查软件错误、解决和处理实验结果的能力。

4. 学生要求人数：2 人，1 人负责扫描器和目标代码生成器的设计和实现，另 1 人负责语法分析器和语法制导翻译程序的设计和实现。

二、 主要内容

下面是课程设计主要内容的简介，详细内容请见《编译技术课程设计指导书》。

1. 扫描器设计

该扫描器是一个子程序，其输入是源程序字符串，每调用一次输出一个单词符号。为了避免超前搜索，提高运行效率，简化扫描器的设计，假设程序设计语言中，基本字不能用作一般标识符，如果基本字、标识符和常数之间没有确定的运算符或界符作间隔，则用空白作间隔。

2. 语法分析器设计

以算法优先分析方法为例，设计一个算符优先语法分析程序。算符优先分析属于自下而上的分析方法，该语法分析程序的输入是终结符号串（即单词符号串，以一个“#”结尾），如果输入串是句子则输出“YES”，否则输出“NO”和错误信息。当然，也可采用预测分析等方法设计语法分析器，具体方法自定。

3. 语法制导翻译程序设计

采用语法制导翻译方法，实现算术表达式、赋值语句和基本控制语句等的翻译。本语法制导翻译程序的输入是终结符号串（即单词符号串，以一个“#”结尾），如果输入符号串是句子,则按照其语义进行翻译,输出等价的四元式序列。

4. 目标代码生成器设计

将程序设计语言的中间代码程序翻译为目标代码程序，其输入是四元式序列，输出是一个汇编代码文件。

三、 进度计划

序号	设计内容	完成时间	备注
1	任务布置，资料查询，方案制定	课程设计规定周次到来前	课程设计由 2 周改为 1 周，时间紧，需提前下发任务并执行
2	算法设计，程序实现	课程设计规定周次的周三前	
3	撰写报告，软件验收	周四、五	
4			

四、设计成果要求

1. 完成规定的课程设计任务，所设计软件功能符合要求；

2. 完成课程设计报告，要求格式规范，内容具体而翔实，应体现自身所做的工作，注重对设计思路的归纳和对问题解决过程的总结。

五、 考核方式

1. 平时成绩＋验收答辩＋课程设计报告；

2. 五级分制。

学生姓名：

指导教师：鲁斌　李继荣

2022 年 8 月 29 日

一、课程设计的目的与要求

1. 任务：实现一个简单的编译程序，能够对指定程序设计语言进行编译。

2. 目的:加深对课堂讲授知识的理解,熟练掌握编译程序设计原理及常用的技术，建立编译程序的整体概念，使得学生初步具有研究、设计、编制和调试编译程序的能力。

3. 要求：熟悉有关定义、概念和实现算法，设计出程序流程框图和数据结构，编写出完整的源程序，进行静态检查，设计出输入数据、显示输出数据；基本功能完善，方便易用，操作无误；通过课程设计学会编译程序设计与实现的常用技术，具备初步分析、设计和开发编译程序的能力，具备分析与检查软件错误、解决和处理实验结果的能力。

4. 学生要求人数：2 人，1 人负责扫描器和目标代码生成器的设计和实现，另 1 人负责语法分析器和语法制导翻译程序的设计和实现。

二、课程设计正文

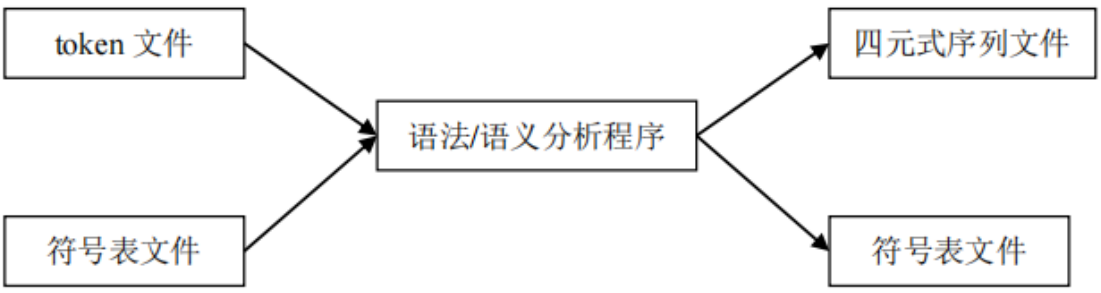
语法/语义分析

1. 目 的

通过设计、编制、调试一个典型的语法分析程序，实现对词法分析程序所提供的单词序列进行语法检查和结构分析，进一步掌握常用的语法分析方法。

2. 任 务

在词法分析程序产生的 token 文件、符号表文件基础上，完成语法和语义分析，产生相应的中间代码——四元式序列。在此，可把语法/语义分析作为独立的一遍



进行处理。

采用如下四元式：

操作码助记符	四元式	意义
:=	(51, a, 0, r)	r←a
+	(43, a, b, r)	r←a+b

-	(45, a, b, r)	r←a-b
*	(41, a, b, r)	r←a*b
/	(48, a, b, r)	r←b/a
j<	(53, a, b, n)	若 a<b 转至第 n 个四元式
j<=	(54, a, b, n)	若 a<=b 转至第 n 个四元式
j>	(57, a, b, n)	若 a>b 转至第 n 个四元式
j>=	(58, a, b, n)	若 a>=b 转至第 n 个四元式
j=	(56, a, b, n)	若 a=b 转至第 n 个四元式
j	(52, 0, 0, n)	转至第 n 个四元式
j<>	(55, a, b, n)	若 a<>b 转至第 n 个四元式

3. 设计过程

3.1 设计思路

①设计一个 Token 类和一个 Symbol 类，再用 Token 列表和 Symbol 列表来接受读入的 token.txt 和 symbol.txt 文件；

②依次读取 Token 列表内容，用 next 指向当前 Token 对象；

③根据 Token 对象的 Type 值，判断语句是否合法以及应该进入哪些处理阶段；

④依次处理 Token 列表内容

3.2 数据结构

Token 类

```
public class Token {
    private int type;     // 单词种别编码
    private int address;  // 单词在符号表中登记的指针（其序号）
    public Token(int string, int string2) {
        setType(string);
        setAddress(string2);
    }
    public Token() {}
    public int getType() {
        return type;
    }
    public void setType(int s1) {
```

```
        this.type = s1;
    }
    public int getAddress() {
        return address;
    }
    public void setAddress(int s2) {
        this.address = s2;
    }
    public String toString() {
        return type + "    " + address;
    }
    public Object clone() throws CloneNotSupportedException {
        Token other = new Token(address, address);
        return other;
    }}
```

Symbol 类，用来存放 L 语言源程序中出现的标识符和常数

```
public class Symbol{
    private int number;        // 序号
    private String name;       // 名字
    private int type;      // 类型
    public Symbol(int number, String string2) {
        // TODO Auto-generated constructor stub
        setNumber(number);
        setName(string2);
    }
    public Symbol() {
        // TODO Auto-generated constructor stub
    }
    public int getNumber() {
        return number;
    }
    public void setNumber(int string) {
        this.number = string;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getType() {
        return type;
    }
    public void setType(int type) {
        this.type = type;
    }
    public String toString() {
        return number + "    " + name ;
    }
```

四元式类

```
public class Equality4 {
    private int label;     // 四元式序号
    private int operator;       // 操作码
    private int leftAddress;   // 左操作数在 symbol 表的入口地址
    private int rightAddress;              // 右操作数在 symbol 表的入口地址
    private int resultAddress;       // 结果在 symbol 表的入口地址
    public int getLabel() {
        return label;
    }
    public void setLabel(int label) {
        this.label = label;
    }
    public int getOperator() {
        return operator;
    }
    public void setOperator(int operator) {
```

```
        this.operator = operator;
    }
    public int getLeftAddress() {
        return leftAddress;
    }
    public void setLeftAddress(int leftAddress) {
        this.leftAddress = leftAddress;
    }
    public int getRightAddress() {
        return rightAddress;
    }
    public void setRightAddress(int rightAddress) {
        this.rightAddress = rightAddress;
    }
    public int getResultAddress() {
        return resultAddress;
    }
    public void setResultAddress(int resultAddress) {
        this.resultAddress = resultAddress;
    }
    public boolean isEntrance() {
        return isEntrance;
    }
    public void setEntrance(boolean isEntrance) {
        this.isEntrance = isEntrance;
    }
    @Override
    public String toString() {
        return label + " " + operator + " " + leftAddress + " " + rightAddress + " " +
        resultAddress;
    }
    @Override
    public Object clone() throws CloneNotSupportedException {
```

```
            Equality4 other = new Equality4();
            other.setLabel(label);
            other.setOperator(operator);
            other.setLeftAddress(leftAddress);
            other.setRightAddress(rightAddress);
            other.setResultAddress(resultAddress);
            other.setEntrance(isEntrance);
            return other;
        }
    }
```

出口列表类

```
public class BoolExit {
        // 该语句块的 true 出口链表
        private ArrayList<Integer> trueExits = new ArrayList<>();
        // 该该语句块的 false 出口链表
        private ArrayList<Integer> falseExits = new ArrayList<>();
        public ArrayList<Integer> getTrueExits() {
            return trueExits;
        }
        public void setTrueExits(ArrayList<Integer> trueExits) {
            this.trueExits = trueExits;
        }
        public ArrayList<Integer> getFalseExits() {
            return falseExits;
        }
        public void setFalseExits(ArrayList<Integer> falseExits) {
            this.falseExits = falseExits;
        }
}
```

3.3 执行过程

①初始化，根据读入的 token.txt 和 symbol.txt 文件，初始化 tokenList 和 symbolList

```
public void initial(String token, String symbol) throws Exception {
    BufferedReader br = new BufferedReader(new FileReader(new File(token)));
```

```
        String line;
        tokenList.add(new Token());
        while ((line = br.readLine()) != null) {
            String[] parts = line.split(" ");
            if (parts.length == 2) {
                Token t=new Token(Integer.parseInt(parts[0]), Integer.parseInt(parts[1]));
                tokenList.add(t);
            }
        }
        br.close();
        BufferedReader br2 = new BufferedReader(new FileReader(symbol));
        symbolList.add(new Symbol());
        while ((line = br2.readLine()) != null) {
        String[] parts = line.split(" ");
            if (parts.length == 2) {
                Symbol s=new Symbol(Integer.parseInt(parts[0]), parts[1]);
                symbolList.add(s);
            }
        }
        br2.close();
        Equality4 equality4 = new Equality4();
        equality4List.add(equality4);
}
```

②开始处理代码

```
public void implement() {
        next();
        if (tokenList.get(i).getType() == 12) {   // 含有 program
            next();
            if(tokenList.get(i).getType()==21) {     // 是标识符，作为程序名字
                next();
                programBody();
            }
        }}
```

三、课程设计总结或结论

本次试验在做的过程中遇到了很多困难，经过多次的尝试，最终完成了程序的编写与运行。 由于我在刚开始做实验时，对编译的整体过程还不能很好掌握，在初期对如何去分析需求，如何去实现需求无从下手，通过向同学请教、网上搜索相关教学等才有了一个大概的思路，并尝试着去做。在实际编程中也遇到了多种问题，有些是自己粗心原因，比如在代码编写时犯了不少低级错误。有些是由于自己还是没能很好掌握语法语义分析的原理。还有些是在实际编程中可能用到课本上未学过的编程知识。由于实验较为复杂，用到的数据结构较多，内部方法也很繁杂，导致我在编写时也是时常晕头转向。对此，自己不断逐层分析，改正错误。

本次实验综合考察了我对编译原理的掌握、对编程的熟练程度以及对各种数据结构的把握能力，既让我对本学期所学的编译技术课程的全部内容有了一个很全面的复习，加深了我对编译原理的理解与认识，也很大程度的提升了我的编程水平，让我对 Java 有了更深层次的认识。最后，感谢老师们的悉心教导与指正。

四、参考文献

[1] 鲁斌,，李继荣，黄建才. 编译技术基础教程. 清华大学出版社, 第一版. 2011,10

附录（设计流程图、程序、表格、数据等）