

简单翻译器

编译原理实验

组长：软件22 王雨婷 2012013328

组员：软件22 王华清 2012013317

组员：软件22 孙 浩 2012013315



2015-6-30

编译原理

**实验环境和实验选题**

操作系统：windows 7 旗舰版 JAVA：JDK 1.8

ANTLR：antlr 3.5.2以及antlrworks-1.4.3

源语言：C++ 目标语言：javascript 测试程序：四则运算计算

**实验目的及要求**

实现一种语言的编译器，将输入的源语言的程序翻译成目标语言程序。

**实验内容**

首先根据C++和javascript的语法差异设计文法，利用antlr工具生成Lexer和Parser的java文件，编写语法制导翻译，通过antlrworks生成语法分析树文法进行检测、验证。之后再输入C++的四则运算程序，输出对应的javascript程序。

**源程序——四则运算**

相关引用

设定全局变量

int main(){

while(true){

读入输入的算式，调用play函数，输出计算结果。}

}

double play(char flag){ //操作式子,flag为式子结束标识

for 式子未结束{

if遇到数字 通过一个while对数字进行提取操作。

else if遇到括号 产生一个新的式子。

else if遇到操作符 栈为空则入栈，否则比较操作符与栈顶操作符优先级，并选择对当前优先的算式进行计算。}

while 栈内操作符非空{

根据栈内操作符完成剩下算式的计算。}

return 计算结果

}

**C++和javascript的语法对比**

首先我们仔细研究了两种语言之间的相似性和差异性。主要对比如下：

函数声明：

C++——有返回值的声明，有参数类型声明。

returnType functionID (paraType para, …, paraType para){}

Javascript——无返回值，内部有return 操作，无需传递参数类型。

functionID = function(para list){}

函数传参：

C++——需要传递参数类型，支持引用类型&。

Javascript——无需传递参数的数据类型，除对object 类型之外不支持引用的类型。

变量类型

C++——明确区分数据类型。

Javascript——除了对数组是array 类型之外基本不区分数据类型，都为var 类型。

构建数组

C++——arrayType arrayID[] = {para}。

Javascript——var arrayID = new Array(para)。

整体结构

C++需要main 函数作为程序的入口，对语法规则限制更多，更加规范。而Javascript 没有特定的main 函数作为程序的起始地址，整个程序布局由function 定义语句块作为分界，相对来说限制较少，语言更灵活。

**文法设计（部分）**

以下是我们设计的文法。在设计的过程中我们注意消除了左递归，提取了左公因子，而且文法的名称非常易懂。

program(程序) -> statements(语句群)

statements（语句群） -> statement（语句） statements(语句群) | e

statement（语句）-> type func\_or\_var(函数或者变量)

| "#include<" ID ">"

| 'using' 'namespace' ID ';'

| COMMENT（注释）

func\_or\_var -> dec\_func(函数) | dec\_var（变量）

//函数声明

dec\_fun -> dec\_func\_name '{' func\_implement（函数实现） '}' semi\_colon

dec\_func\_name -> ID '(' dec\_param(参数们) ')’

dec\_param（参数们） -> type ID dec\_param\_(参数递归) | e

dec\_param\_ （参数递归）-> , dec\_param | e

func\_implement(函数实现) -> block

//变量声明

dec\_var -> ID dec\_array ';' (数组定义)

| ID dec\_array '=' array\_value ';'(数组定义+赋值)

| ID dec\_expression dec\_var\_（正常变量）

dec\_array -> [ INT ]

array\_value -> { INT ints }

dec\_var\_ -> ';' | ',' dec\_var

dec\_expression -> expr\_value | e

**工具使用**

本次实验中用到了开源的语法分析器——anltr，由上述的文法设计编译好文法文件，通过antlr处理.g文件可生成对应的词法分析器和语法分析器的java文件。

除此之外，为了验证文法设计的合法性，我们利用antlrworks对文法产生语法分析树，从而检测、验证设计的文法。

**支持语法**

在本次编译器中我们实现了如下的语法结构：

函数定义,函数调用；全局变量，局部变量；数组定义、初始化；数组下标取值\；while循环结构；for循环结构；if-else if-else选择分支结构；switch-case选择分支结构；int，double，char，void类型；逻辑运算符；算术运算符；++、--；输入、输出；以及注释的转化。

在编译器中我们把动作内嵌到产生式中，让编译的过程更加高效。

除此之外，我们还实现了javascript的格式化。在产生式中增加tab属性（继承属性），记录每行代码需要缩进的值，最后输出时候对于js语句进行调整，使得输出的语法可读性更强。

**实验结果**

左边是C语言，右边是Javascript。

