程序设计语言基础 课程大作业（一）

【概述】

实现简单语言Minilan的解释器。

编写一个程序，接受一个Minilan文件和一个数据文件，输出运行结果到文件中。

不考虑语法错误和运行时错误，即假设代码文件和相应数据完全正确，能正确运行且可以正常终止。

【Minilan文法】

*number* -> (0..9)+

*name* -> (a..z)+

*block* -> Begin (*function|command*)\* End

*function* -> Function *name* Paras (*name*)\* block

*command* -> Var (*name)*+ End

| Assign *name expr*

| Call *name* Argus *(expr)\** End

| Read *name*

| Print *expr*

| If *boolexpr* *block Else* *block*

| While *boolexpr* *block*

| Return *expr*

*expr* ->*number*

| *name*

| Plus *expr expr*

| Minus *expr expr*

| Mult *expr expr*

| Div *expr expr*

| Mod *expr expr*

| Apply *name* Argus *(expr)\** End

*boolexpr* ->Lt *expr expr*

| Gt *expr expr*

| Eq *expr expr*

| And *boolexpr boolexpr*

| Or *boolexpr boolexpr*

| Negb *boolexpr*

注1：首字母大写的词语为关键字

注2：出现\*和+的条目，非block时多项之间以空格分隔，block中语句之间以换行符分隔

【Minilan语义】

变量只有整形

Assign Name Expr 计算Expr的结果，并赋值给Name

Read Name 从数据文件中读取一个数，并赋值给Name

Print Name 将Name对应变量的值取出，并写入输出文件。

函数定义时 Paras后接形参表，函数调用（Apply\Run）时Argus后接实参表。允许无参函数，但Paras和Argus不可省略。

Expr为表达式，包括加减乘除（整除）和求余，以及函数调用

BoolExpr为布尔表达式，包括大于、小于、等于三种关系运算和与或非三种逻辑运算

【程序运行说明】

该程序的函数允许嵌套（从语法中可看出），参数传递使用传值法进行，寻找全局变量的申明使用静态作用域规则。

函数的返回值也为整形，使用 return 语句返回，或者函数结束时默认返回 0。使用函数有两种方法，在表达式中使用和直接作为语句使用。

程序顺序执行所有语句。同一Block内重复声明的变量，除第一次外声明语句无效，不同Block内同名局部变量的声明语句将覆盖外层Block相应声明。被声明的变量在离开该层Block后不能被使用。

【输入输出的表示方法】

输入文件 input.txt，内含用空格隔开的若干整数，即为程序的输入数据；输入文件 program.txt，为程序本身；输出文件为 output.txt。

从该语言语义可以看出仅有 Read变量、Print 表达式这两条语句用于输入输出。

程序每运行到Read *name*时，从输入文件中提取一个整数，赋值给变量。

每运行到 Print *expr*时，在 output.txt 的结尾添加一个相应表达式的值的整数，以空格分隔。

【例子】

Begin

Var x End

Assign x 1

Function fact Paras i

Begin

If Lt i 2

Begin

Return 1

End

Else

Begin

Return Mult i Apply fact Argus Minus i 1 End

End

End

Read x

Print Apply fact Argus x End

End

此程序用类 C 语言可写成:

int x;

x = 1;

int fact(int i)

{

if(i<2) return 1;

else return i \* fact(i-1);

}

read(x);

print(fact(x));

若输入为 3，则输出应为 6。

【扩展：高阶函数】

若在Minilan中允许函数的形参、实参和返回值为函数名，则称其为高阶函数扩展的Minilan。

在扩展后的Minilan中，变量有两种类型，整形\函数类型，这里暂不要求对函数类型进行更具体的划分，函数调用的正确性由测试程序代码本身来保证。

当声明变量后，对变量赋值前，变量的类型和值均不存在，对其进行Print操作无意义（不进行任何行为）。当变量被赋值为整形后，可以进行Print操作，而变量被赋值为函数后，进行Print操作无意义（不进行任何行为）。

变量被赋值为函数的语句仍为Assign语句，通常形式为

Assign name Apply *name* Argus *(expr)\** End

即将高阶函数的返回值（函数）赋值给变量。之后，被赋值的变量可以作为函数名使用，可以作为Apply语句和Call语句的参数。

被赋值为函数名的对象允许再被赋值为整形，而被赋值为整形的对象也允许再被赋值为函数名。

Example 1:

Begin

Var x End

Assign x 4

Function g Paras z

Begin

Return Mult x z

End

Function f Paras h

Begin

Var x End

Assign x 7

Return Plus x Apply h Argus 3 End

End

Print Apply f Argus g End

End

此程序用类C语言可写成:

int x;

x = 4;

int g(int z){

return (x \* z);

}

int f(fun h){

int x;

x = 7;

return h(3) + x;

}

print f(g);

该程序输入为空，输出为 19。

Example 2:

Begin

Function f Paras g h Begin

Function t Paras x Begin

Return Apply g Argus h x End

End

Return t

End

Function add Paras x y Begin

Return Plus x y

End

Var a End

Assign a Apply f Argus add 1 End

Print Apply a Argus 1 End

End

【提供部分代码】

我用C++为Minilan编写一个简单的分析器，提供一个Term类，将代码转换为Term对象的语法树，方便大家实现。

class Term{

public:

TermKind kind;

TermSubtype subtype;

Term\* father;

std::list<Term\*> sons;

int number;

std::string name;

Term(){}

Term(TermKind kind){this->kind=kind;}

void print();

};

其中TermKind表示Term结点类型，包括 Block,Function,Command,Expr,BoolExpr和Name，TermSubtype表示子类型，如Command中的子类型有Assign,Read,Print等。

Number和Name类型归属于Expr类型中，通过TermSubtype区分。

在文法中要求只能为Name类型时，将生成Name类型节点，若可以为Expr类型，则生成Expr类型节点。

同时提供parse函数，读取输入流，并生成Term语法树。

同时提供print成员函数，可以在控制台输出语法树。

具体定义和实现见Term.h与Term.cpp。如在使用中发现问题请立刻用邮件联系。