Design for MyLang

组员：
石嘉昊 5130379072
黄志强 5120809022
陈嘉南 5130379006

目录

一、简介.....3
1、类型系统.....3
2、运行模型.....3
3、操作符.....3
4、内建函数.....4
5、控制流.....4
6、其他功能.....5
二、词法.....6
三、语法.....7
四、样例程序.....8

一、简介

1、类型系统

命令式语言，静态语言；

2、运行模型

直接按照顺序执行，如果当前是函数的声明或者函数的定义则不执行，直到调用才执行，对于其他的就直接执行。内存的管理自动进行，离开作用域就删除，调用使用 stack 实现，采用本地代码。

3、操作符

+（加） -（减） *（乘）/（除）%(取模）()(括号）[](数组下标）->(Class 成员) :=(赋值) =(等于) !=(不等于) <(小于) <= > >= &(逻辑操作与) |(逻辑操作或) ~(逻辑操作非)。

注意：基本数据类型赋值：复制相应的值。

扩展数据类型的赋值：分别取每一项进行赋值。

优先级表格：

优先级	操作符	结合方式
1	()(括号)；[](数组下标)；->(Class 成员)	左结合
2	~（逻辑操作非）	右结合
3	*（乘）；/（除）； %(取	左结合

	模）	
4	+（加）；-（减）	左结合
5	<(小于）；<=（小于等于）；>（大于）；>=（大于等于）	左结合
6	=（等于）；!=（不等于）	左结合
7	&(逻辑操作与)	左结合
8	|(逻辑操作或)	左结合
9	:=（赋值）	右结合

4、内建函数

read（identifier，identifier，….）按照顺序读入用户输入的值，并赋值给相应的 identifier。（多了取对应的，少了给默认值）

read(Array|Class,N)　读入 N 个用户的值，并复制给 Array 或者 Class 的每一项。（多了取对应的，少了给默认值）

write（identifier, identifier，….）按照顺序输出相应的值。

write(Array|Class,N)　输出 Array 或者 Class 中的前 N 项。（N 大了全输出，少个取相应的）。

5、控制流

选择结构：

if(statement) then

……..

(else if)

……..

(else)

endif

循环结构：

while(statement)

….

endwhile

for(<identifier> from <identifier|Integer> to <identifier|Integer> step <identifier|Integer>)

…

endfor

foreach(<identifier|Integer> in Array)

…

Endforeach

6、其他功能

支持数据结构：Integer，Float，Char（基本类型）

Array，Class（扩展类型）

注意：Array 的访问采用 name[index]的方式；

Class 的访问采用 name->varname 的方式

1-5 的功能描述

函数的声明：function identifier（identifier，….）；

函数的定义：function identifier（identifier，…）

…….

(return)

endfunction

注意：函数对于基本函数类型采用的是传值的方式，对于扩展类型采用传引用的方式。

函数必须先声明才能使用

二、词法

- Interger::= [+|-] <0|1|2…|9> {0|1|2…|9};
- Float ::= [+|-] <0|1|2…|9> {0|1|2…|9} "." {0|1|2...|9};
- Char::= ASCII (0-127);
- Const ::= Integer | Float | Char;
- Identifier::=<_|a|b|...|z|A|B|...|Z>{_|a|b|...|z|A|B|...|Z|0|1|...|9} (not Keyword);
- Keyword::= "Interger" | "Float" | "Char" | "Array" | "Class" | "if"| "then" | "else" | "endif" | "while" | "endwhile" | "for" | "endfor" | "step" | "from" | "to" | "foreach" | "in" | "endforeach" | "var" | "as" | "of" | "function" | "endfunction" | "return" | "endclass" ;
- Op::= +|-|*|/|%|(|):=|=|!=|<|<=|>|>=|&|||~;
- BaseType::= [Integer|Float|Char];
- ArrayType ::= [Integer|Float|Char|Class|Array];

三、语法

- BaseDeclaration::= "var"<Identifier> "as"<BaseType> “;” ;
- ArrayDeclaration::="var"<Identifier> "as""Array["<Integer>{","Integer} "]" "of"<ArrayType> “;” ;
- ClassDeclaration ::= "var"<Identifier>"as" <Identifier>” ;” ;
- ClassDefination::=” Class” <Identifier> <BaseDeclaration|ArrayDeclaration|FunctionDefination|ClassDeclaration>{BaseDeclaration|ArrayDeclaration|FunctionDefination| ClassDeclaration} “endclass” ;
- ClassSucceed::=” Class” <Identifier>” extends” ” Class” <Identifier><BaseDeclaration|ArrayDeclaration|FunctionDefination|ClassDeclaration>{BaseDeclaration|ArrayDeclaration|FunctionDefination|ClassDeclaration} “endClass” ;
- AssignStatement::=Identifier":="<Expression> “;” ;
- IfStatement::= "if""("<Expression>")" "then" <Statement> ["else"<IfStatement>] ["else"<Statement>] "endif";
- ForStatement::="for""("<Identifier> "from"<Integer|Identifier> "to" <Integer|Identifier> "step" <Integer|Identifier> ")" <Statement> "end for";
- ForeachStatement::= "foreach" "("<identifier> "in" <Array> ")" <Statement> "endforeach";
- WhileStatement::= "while""("<Expression>")"<Statement> "endWhile";
- FuctionDeclaration::= "function" <identifier> "(" [identifier] {"," <identifier> } ");";
- FunctionDefination::= "function" <identifier> "(" [identifier] {"," <identifier> } ")" Statement ["return"<Expression>]"endFunction";
- FunctionCall::= <identifier "("> [Identifier | Const]{"," Identifier |Const} <");">
- Exression ::= Term0 | Identifier ":=" Expression;
- Term0 ::= Term1 | Term0 "|" Term1;
- Term1 ::= Term2 | Term1 "&" Term2;
- Term2 ::= Term3 | Term2 "=" Term3 | Term2 "!=" Term3;
- Term3 ::= Term4 | Term3 "<" Term4 | Term3 "<=" Term4 | Term3 ">" Term4 | Term3 ">=" Term4;
- Term4 ::= Term5 | Term4 "+" Term5 | Term4 "-" Term5;
- Term5 ::= Term6 | Term5 "*" Term6 | Term5 "/" Term6 | Term5 "%" Term6;
- Term6 ::= Term7 | ~Term7;
- Term7 ::= <Identifier | Integer | Float | Char> | "(" <

Expression>")" | Array "[" < Expression> "]" | Class "->" < Identifier>;

- Statement::= {BaseDeclaration | ArrayDeclaration | ClassDeclaration | AssignStatement | IfStatement | ForStatement | WhileStatement | ForeachStatement | FunctionCall};

四、样例程序

```
QuickSort:
    function _QuickSort(A, iLo, iHi)
    var Lo as integer;
    var Hi as integer;
    var Mid as integer;
    var T as integer;
    var temp as integer;
    Lo := iLo;
    Hi := iHi;
    Mid := A[(Lo + Hi) div 2];
    while (Lo > Hi)
    while (A[Lo] < Mid) do Lo := Lo + 1; endwhile;
    while (A[Hi] > Mid) do Hi := Hi - 1; endwhile;
    if (Lo <= Hi) then
    temp := A[Lo];
    A[Lo] := A[Hi];
    A[Hi] := temp;
    T := A[Lo];
    A[Lo] := A[Hi];
    A[Hi] := T;
    Lo := Lo + 1;
    Hi := Hi - 1;
    endif;
    endwhile;
    if (Hi > iLo) then _QuickSort(A, iLo, Hi); endif;
    if (Lo < iHi) then _QuickSort(A, Lo, iHi); endif;
    endfunction;

    function QuickSort(A, size)
    _QuickSort(A, 0, size-1);
    endfunction;
```

主函数调用 QuickSort 就可以了。例如，

```
var A as Array[10] of Integer;
```

```
QuickSort(A,10);
```