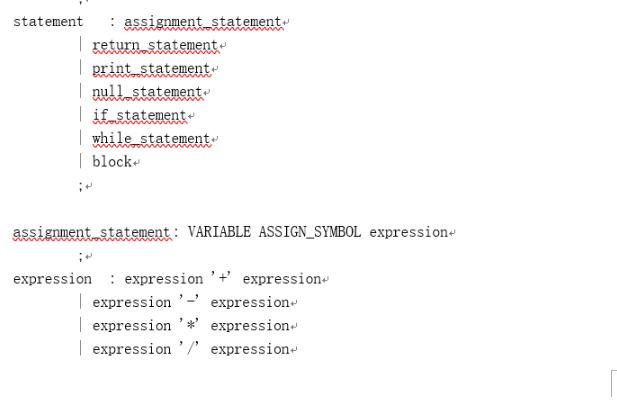
**第五章-IF语句和函数块block设计文档**

根据VSL的语法语义，它将IF等关键字归为statement类。运算符等为expression，根据这个，则建立一个新的语法树基类



//statement抽象语法树基类

class StatAST {

public:

virtual ~StatAST() = default;

virtual Value\* codegen() = 0;

};

这个语法树用于各个关键字，然而实际测试的时候，因为Number等语法树是继承与ExprAst的，再进行分析的时候容易混淆，故将部分语法树都改为继承Statement基类，并在后面将进行相应处理

一．块函数blcok

因为与万花筒不同，VSL有函数体函数块的概念，是多个statement语句的集合，可以观察到在FUNC函数{}中是函数体，因为还有WHILE语句里会嵌套函数块，所以首先建立一个新的语法树存储一个{}里的所有state并存储为一个列表。

首先是block语法树

class BlockStatAST : public StatAST {

std::vector<std::unique\_ptr<StatAST>> StatList;

public:

BlockStatAST(std::vector<std::unique\_ptr<StatAST>> StatList)

: StatList(std::move(StatList)) {}

public:

Value \*codegen() override;

};

然后全局函数初始化

static std::unique\_ptr<StatAST> ParseBlock();

接着解析函数块

//解析函数块

//block::='{' declaration\_list statement\_list '}'

static std::unique\_ptr<StatAST> ParseBlock() {

std::vector<std::unique\_ptr<StatAST>> StatList;

getNextToken(); //eat '{'

while (CurTok != '}') {

if (CurTok == '{') {

ParseBlock();

}

else if (CurTok == CONTINUE) {

getNextToken();

}

else {

auto statResult = ParseStatement();

StatList.push\_back(std::move(statResult));

}

}

getNextToken(); //eat '}'

return llvm::make\_unique<BlockStatAST>(std::move(StatList));

}

同时建立一个新的parseStatement分析statement类型的函数，之后用于IF等关键字如下并初始化

//statment类型

static std::unique\_ptr<StatAST> ParseStatement()

{

switch (CurTok) {

case'{':

return ParseBlock();

case IF:

return ParseIfStat();

break;

case RETURN:

return ParseRetStat();

case VAR:

return ParseDecStat();

break;

case WHILE:

return ParseWhileStat();

break;

default:

return ParseAssStat();

}

}

二.IF语句和相关关键词

IF语句和then，else等基本同万花筒，但是在VSL中IF语句需要有RETURN关键字。则有

1.RETURN语法树

//RETURN　语法树

class RetStatAST : public StatAST {

std::unique\_ptr<StatAST> Val;

public:

RetStatAST(std::unique\_ptr<StatAST> Val)

: Val(std::move(Val)) {}

Value \*codegen() override;

};

2.RETURN关键字解析

static std::unique\_ptr<StatAST> ParseRetStat(){

getNextToken();

auto Val = ParseExpression();

if (!Val)

return nullptr;

return llvm::make\_unique<RetStatAST>(std::move(Val));

}

3.IR生成

Value \*RetStatAST::codegen() {

Function \*TheFunction = Builder.GetInsertBlock()->getParent();

if (Value \*RetVal = Val->codegen()) {

Builder.CreateRet(RetVal);

BasicBlock \*afterRet = BasicBlock::Create(TheContext, "afterReturn", TheFunction);

Builder.SetInsertPoint(afterRet);

return RetVal;

}

}

同时IF语句末尾有一个新的关键字FI用于结束IF,但是不影响其他，则可在解析时添加如下即可

if (CurTok == ELSE) {

getNextToken();

Else = ParseExpression();

if (!Else)

return nullptr;

}

else if (CurTok != FI)

return LogError("expected FI or ELSE");

if (CurTok != FI)

return LogError("expected FI");

getNextToken();