编译原理实验文档

P5——C1 编译器

PB09210340 张宇鹏

Mail: ypzhange@mail.ustc.edu.cn

特点和支持：

类型系统：

基本类型：int float bool char

衍生类型：**指针** 数组(多维)[*不对插入语句对数组越界进行运行时检查*]

修饰词：**静态修饰** 常量修饰

类型定义

字符串

复合类型的**结构等价**判断

隐式类型转换

函数系统：

支持返回值，参数，**无基本类型限定**，递归调用

不支持函数体中的函数声明

语句：

表达式语句

条件语句 （条件表达式短路计算）

For/While 循环语句

复合语句

输入/输出语句

空语句

解释器：

自制解释器 EsIR(请参考其独立的说明文档)

文件：

SRC/

Type.c

ASTCodeGenerater.c

ast.c

interpreter.c

symtab.c

list.c *Reference

Config/

C1.lex

C1.y

Include/

common.h

op.h

type.h

u

Test/

CESP.c1 //短路计算测试

Originaltest.c1 //各种测试

StaticTest.c1 //静态变量测试

Recusive.c1 //递归测试

ForTest.c1 //For 语句测试

Right.c1 //综合测试(堆排序)

除去 list.c 外，所有上述文件均作了大量的修改。

使用方法：

编译器生成:

C1$ make C1

编译:

C1$ bin/C1 test/right.c1 -o test/right.o
C1 可执行文件 目标源代码 中间代码输出文件

解释运行:

C1$ bin/Interpreter test/right.o [-d]/[-d2]
解释器可执行文件 目标中间代码 调试选项(-d2 is detail debug)

程序样例:

Test 目录下任意.c1 文件.
Right.c1 测试结果：

erthanese@erthanese-VirtualBox:~/Documents/C1$ bin/Interpreter test/right.o
start C/1
Before the sort
==========
91 48 1 80 28 22 15 88 59 5
After the sort
==========
1 5 15 22 28 48 59 80 88 91
end C/1

词法系统：

```
%x COMMENT
%x LINECOMMENT

identifier [A-Za-z_][A-Za-z_0-9]*
character   \'([^'\n])+\'
bad_chr      \'([^'\n])
string       \"([^"\n])+\"
bad_string \"([^"\n])+

%%
"//"                        BEGIN(LINECOMMENT);
<LINECOMMENT>[^\n]+
<LINECOMMENT><<EOF>>      error(1);
<LINECOMMENT>\n   BEGIN(INITIAL);
"/*"                        BEGIN(COMMENT);
<COMMENT>[^*/]+             ;
<COMMENT><<EOF>>      error(1);
<COMMENT>"*/"        BEGIN(INITIAL);
[\t ]*
"{"     return(BEGINSYM);
"}"     return(ENDSYM);
"["     return(LBRACK);
"]"     return(RBRACK);
"const"     return(CONSTSYM);
"static"     return(STATICSYM);
"typedef"          return(TYPEDEFSYM);
"float"      {yylval.tval = Float; return(BASETYPE);}
"int"              {yylval.tval = Int; return(BASETYPE);}
"char"      {yylval.tval = Char; return(BASETYPE);}
"bool"      {yylval.tval = Bool; return(BASETYPE);}
"void"      {yylval.tval = Void; return(BASETYPE);}
"true"
"false"
"break" return(BREAKSYM);
"return" return(RETURNSYM);
"main"  return(MAINSYM);
"if"            return(IFSYM);
"else"   return(ELSESYM);
"while"  return(WHILESYM);
"for"           return(FORSYM);
"read"   return(READSYM);
"write"   return(WRITESYM);
```

```
"sizeof"	{yylval.ival = OP_SIZEOF; return(SIZEOF);}
{identifier}
[0-9]+[eE]["+"|"-"]?[0-9]+
[0-9]+"."[0-9]+
{character}
{bad_chr}
{string}
{bad_string}
0[0-7]+
0[xX][0-9A-Fa-f]+
[0-9]+
"<<"		{yylval.ival = OP_LSH; return(LSH);}
">>"		{yylval.ival = OP_RSH; return(RSH);}
"=="		{yylval.ival = OP_EQL; return(EQL);}
"<="		{yylval.ival = OP_LEQ; return(LEQ);}
"!="		{yylval.ival = OP_NEQ; return(NEQ);}
"<"		{yylval.ival = OP_LSS; return(LSS);}
">="		{yylval.ival = OP_GEQ; return(GEQ);}
">"		{yylval.ival = OP_GTR; return(GTR);}
"*"	{yylval.ival = OP_MULT; return(MULT);}
"/"	{yylval.ival = OP_DIV; return(DIV);}
"+"		{yylval.ival = OP_PLUS; return(PLUS);}
"-"	{yylval.ival = OP_MINUS; return(MINUS);}
"%"		{yylval.ival = OP_MOD; return(MOD);}
"="		{yylval.ival = OP_ASGN; return(ASGN);}
"&&"		{yylval.ival = OP_ANDAND; return(ANDAND);}
"||"		{yylval.ival = OP_OROR; return(OROR);}
"!"	{yylval.ival = OP_NOT; return(NOT);}
"~"		{yylval.ival = OP_BITNOT; return(BITNOT);}
"&"		{yylval.ival = OP_AND; return(AND);}
"|"	{yylval.ival = OP_OR; return(OR);}
"^"		{yylval.ival = OP_XOR; return(XOR);}
","	{return(COMMA);}
";"	{return(SEMICOLON);}
"("	{return(LPAREN);}
")"	{return(RPAREN);}
[\n]
%%
```

语法系统：

```
 0 $accept→ Program $end

 1 Program→ DeclRegon Main

 2 DeclRegon→ /* empty */
 3          | DeclRegon DeclStatement

 4 DeclStatement→ VarDecl SEMICOLON
 5              | FuncDecl
 6              | TypeDecl

 7 Type→ BASETYPE
 8       | TYPEID
 9       | CONSTSYM Type
10       | STATICSYM Type

11 VarDecl→ Type DelicateID InitExp
12          | Type error
13          | VarDecl COMMA DelicateID InitExp

14 DelicateID→ UNREGID
15           | VARID
16           | FUNCTIONID
17           | MULT DelicateID
18           | DelicateID LBRACK NUMBER RBRACK
19           | LPAREN DelicateID RPAREN

20 ArrayValue→ BEGINSYM NUMBER
21           | ArrayValue COMMA NUMBER

22 InitExp→ /* empty */
23          | ASGN Exp
24          | ASGN ArrayValue ENDSYM

25 TypeDecl→ TYPEDEFSYM Type DelicateID SEMICOLON
26           | TYPEDEFSYM error SEMICOLON

29 FuncDecl→ Type DelicateID LPAREN ParamList RPAREN Statement

30 ParamList→ /* empty */
31           | Type DelicateID
32           | ParamList COMMA Type DelicateID
```

```
34 Main→ BASETYPE MAINSYM LPAREN RPAREN Statement

35 Statement→ SEMICOLON
36              | Exp SEMICOLON

38 Statement→ FORSYM LPAREN Statement Exp SEMICOLON Exp RPAREN Statement
39              | WHILESYM LPAREN Exp RPAREN Statement
40              | BREAKSYM SEMICOLON
41              | RETURNSYM SEMICOLON
42              | RETURNSYM Exp SEMICOLON
43              | IFSYM LPAREN Exp RPAREN Statement ELSESYM Statement
44              | IFSYM LPAREN Exp RPAREN Statement

46 Statement→ BEGINSYM MulStatement ENDSYM
47              | VarDecl SEMICOLON
48              | READSYM LPAREN Exp RPAREN SEMICOLON
49              | WRITESYM LPAREN Exp RPAREN SEMICOLON
50              | error

51 MulStatement→ /* empty */
52                  | MulStatement Statement

53 ArgList→ /* empty */
54          | Exp
55          | ArgList COMMA Exp

56 Exp→ NUMBER
57      | AdrExp
58      | AND AdrExp


60 Exp→ FUNCTIONID PAREN ArgList RPAREN
61      | BITNOT Exp
62      | NOT Exp
63      | SIZEOF Exp
64      | MINUS Exp
65      | AdrExp ASGN Exp
66      | Exp EQL Exp
67      | Exp NEQ Exp
68      | Exp LEQ Exp
69      | Exp GTR Exp
70      | Exp LSS Exp
71      | Exp GEQ Exp
```

```
72      | Exp PLUS Exp
73      | Exp MINUS Exp
74      | Exp MULT Exp
75      | Exp DIV Exp
76      | Exp MOD Exp
77      | Exp ANDAND Exp
78      | Exp OROR Exp
79      | Exp AND Exp
80      | Exp OR Exp
81      | Exp XOR Exp
82      | Exp LSH Exp
83      | Exp RSH Exp
84      | LPAREN Exp RPAREN

85 AdrExp→ VARID
86          | UNREGID
87          | MULT AdrExp    //This is a useless rule ,but write it will help bison deal a conflict
88          | MULT Exp
89          | Exp LBRACK Exp RBRACK
90          | LPAREN AdrExp RPAREN
```