OEXPL GRAMMAR OUTLINE

➤ OExpL Grammar outline

Annotated OExpL Grammar outline

An outline for the OExpL grammar is given here. Calls to functions that update the symbol table, type table, class table and the abstract syntax tree data structures are indicated as semantic actions at certain places.

- 1. Program :
- ClassDefBlock
- TypeDefBlock
- GDeclBlock
- FDefBlock
- MainBlock

```
Program : TypeDefBlock ClassDefBlock GlobalDeclBlock FuncDefBlock MainBl
;
ClassDefBlock : CLASS ClassDefList ENDCLASS
                |
;
ClassDefList : ClassDefList ClassDef
               | ClassDef
;
Classdef      : Cname '{'DECL Fieldlists MethodDecl ENDDECL MethodDefns
;
Cname         : ID              {Cptr = Cinstall($1->Name,NULL);}
              | ID Extends ID   {Cptr = Cinstall($1->Name,$3->Name);}
;
Fieldlists      : Fieldlists Fld
                |
;
Fld         : ID ID ';'         {Class_Finstall(Cptr,$1->Name,$2->Name);
;
MethodDecl : MethodDecl MDecl
```

```
            | MDecl
;
MDecl      : ID ID '(' Paramlist ')' ';' {Class_Minstall(Cptr,$2->Name,T

;
MethodDefns : MethodDefns FDef
            | FDef
;
stmt :    ...
          | Field ASSIGN Expr ';'
          ...
          | ID ASGN NEW '(' ID ')' ';'
          | Field ASSIGN NEW ’(’ ID ’)’ ';'
          | DELETE ’(’ Field ’)’ ';'
;

Expr:     ...
          | Field
          | FieldFunction
;
Field:    SELF '.' ID
          |ID '.' ID            //This will not occur inside a class.
          |Field '.' ID
;
FieldFunction : SELF '.' ID '(' Arglist ')'
                |ID '.' ID '(' Arglist ')'      //This will not occur in
                |Field '.' ID '(' Arglist ')'
;
Arglist: Arglist ',' Expr
          |Expr
          |
;
```

oexpl.txt hosted with ❤ by **GitHub** **view raw**

```
TypeDefBlock  : TYPE TypeDefList ENDTYPE
              |
              ;

TypeDefList   : TypeDefList TypeDef
              | TypeDef
              ;

TypeDef       : ID '{' FieldDeclList '}'   { TInstall(tname,size,$3); }
              ;

FieldDeclList : FieldDeclList FieldDecl
```

```
13              | FieldDecl
14              ;
15
16  FieldDecl   : TypeName ID ';'
17
18  TypeName    : INT
19              | STR
20              | ID       //TypeName for user-defined types
21              ;
```

Grammar_outline_TypeDef hosted with ❤ by **GitHub** **view raw**

```
1   GDeclBlock : DECL GDeclList ENDDECL
2              |
3              ;
4
5   GDeclList  : GDecList GDecl
6              | GDecl
7              ;
8
9   GDecl      : TypeName Gidlist ';'
10             ;
11
12  Gidlist    : Gidlist ',' Gid
13             |   Gid
14             ;
15
16  Gid        :   ID                      { GInstall(varname,ttableptr,ctab
17             |   ID '(' ParamList ')'      { GInstall(varname,ttableptr,NU
18             |   ID '[' NUM ']'          { GInstall(varname,ttableptr,NULL
19             ;
20
21  ParamList    :  ParamList ',' Param  { AppendParamlist($1,$2);}
22             |  Param
23             |  //There can be functions with no parameters
24             ;
25
26  Param        : TypeName ID { CreateParamlist($1,$2); }
27             ;
28  //NOTE 1: The second argument to the function Ginstall() must be a point
29         2: The third argument to the function Ginstall() must be a pointe
30         3: The functions CreateParamlist() and AppendParamlist() help to
31            of parameters specified in an ExpL function declaration. Desig
```

Grammar_outline_GDeclBlock hosted with ❤ by **GitHub** **view raw**

```
1   FDefList  : FDefBlock
```

```
          | FDefList FDefBlock
          ;
FDefBlock : TypeName ID '(' ParamList ')' '{' LdeclBlock Body '}'  { GUp
          ;

Body      : BEGIN Slist Retstmt END
          ;

Slist     : Slist Stmt
          |
          ;

Stmt      : ID ASGN Expr ';'
          | ....
          | IF '(' Expr ')' THEN Slist ELSE Slist ENDIF ';'
          | ...
          | ID ASGN ALLOC'(' ')' ';'
          | FIELD ASGN ALLOC'(' ')' ';'
          | FREE '(' ID ')' ';'
          | FREE '(' FIELD ')' ';'
          | READ '(' ID ')' ';'
          | READ '(' FIELD ')' ';'
          | WRITE '(' Expr ')' ';'
          ;

FIELD     : ID '.' ID
          | FIELD '.' ID
          ;

Expr      : Expr PLUS Expr   { $$ = TreeCreate(TLookup("int"),NODETYPE_P
          | ....
          | '(' Expr ')'
          | NUM
          | ID
          | ID '[' Expr ']'
          | FIELD
          | ID '(' ArgList ')'  {
                                  gtemp = GLookup($1->name);
                                  if(gtemp == NULL){
                                          yyerror("Yacc : Undefined functi
                                  }
                                  $$ = TreeCreate(gtemp->type,NODETYPE_FUN
                                  $$->Gentry = gtemp;
                                }
          ;
```

Grammar_outline_FDefBlock hosted with ❤ by **GitHub** **view raw**

```
MainBlock : INT MAIN '(' ')' '{' LdeclBlock Body '}'
                                {
                                    type = TLookup("int");
                                    gtemp = GInstall("MAIN",type,0,NULL)
                                    //...Some more work to be done
                                }
          ;
```

Grammar_outline_MainBlock hosted with ❤ by **GitHub** **view raw**

Github

Contributed By : J.Ritesh, J. Phani Koushik, M. Jayaprakash

Home | About