

# 1 Grammar for extended PL/0

*program* ::= *block* .

*block* ::= [ *constdec* ] [ *vardec* ] { [ *procdec* ] [ *fundec* ] } *compstmt*

*constdec* ::= **const** *constdef* { , *constdef* } ;

*constdef* ::= *ident* = *const*

*const* ::= [ + | - ] *unsign* | *character*

*character* ::= ' *letter* ' | ' *digit* '

*string* ::= "{ASCII decimal characters from 32 to 126 exclude 34}"

*unsign* ::= *digit* { *digit* }

*ident* ::= *letter* { *letter* | *digit* }

*vardec* ::= **var** *vardef* ; { *vardef* ; }

*vardef* ::= *ident* { , *ident* } : *type*

*type* ::= *basictype* | **array** ' [ *unsign* ' ] ' **of** *basictype*

*basictype* ::= **integer** | **char**

*procdef* ::= *prothead* *block* { ; *prothead* *block* } ;

*fundef* ::= *funthead* *block* { ; *funthead* *block* } ;

*prothead* ::= **procedure** *ident* ' ( ' [ *paralist* ] ) ' ;

*funthead* ::= **function** *ident* ' ( ' [ *paralist* ] ) ' : *basictype* ;

*paralist* ::= [ **var** ] *ident* { , *ident* } : *basictype* { ; *paralist* }

*statement* ::= *assignstmt* | *ifstmt* | *repeatstmt* | *callstmt*  
| *compstmt* | *readstmt* | *writestmt* | *forstmt* | *nullstmt*

*assignstmt* ::= *ident* := *expression* | *funident* := *expression*  
| *ident* ' [ *expression* ' ] := *expression*

*funident* ::= *ident*

*expression* ::= [ + | - ] *term* { *addop* *term* }

*term* ::= *factor* { *multop* *factor* }

*factor* ::= *ident* | *ident* ' [ *expression* ' ] | *unsign* ' ( ' *expression* ' ) ' | *callstmt*

*callstmt* ::= *ident* ' ( ' *arglist* ' ) '

*arglist* ::= *argument* { , *argument* }

*argument* ::= *expression*

*addop* ::= + | -

*multop* ::= \* | /

*condition* ::= *expression* *relop* *expression*

*relop* ::= < | <= | > | >= | <>

$ifstmt ::= \text{if } condition \text{ then } statement$   
                    $|\text{if } condition \text{ then } statement \text{ else } statement$   
 $repeatstmt ::= \text{repeat } statement \text{ until } condition$   
 $forstmt ::= \text{for } ident := expression \text{ (to|downto) } expression \text{ do } statement$   
 $callstmt ::= ident'([ arglist ])'$   
 $compstmt ::= \text{begin } statement \{; statement \} \text{end}$   
 $readstmt ::= \text{read}'( ident \{, ident \} )'$   
 $writestmt ::= \text{write}'( string , expression )' | \text{write}'( string )' | \text{write}'( expression )'$   
 $letter ::= a|b|c|...|z|A|B|C|...|Z$   
 $digit ::= 0|1|2|3|...|9$