HIGH POINT UNIVERSITY

Department of Computer Science

CSC4510 Programming Language Design and Translation

The Syntax of *Pascal*

Pascal Syntax

Programs and Blocks

1. program::= program_heading semicolon program_block

```
program hi(out); begin writeln('Hello, World!'); end.
```

2. program_heading::="program" identifier ("(" O(program_parameters) ")" |)

```
program merge(infile1, infile2, mergedfile);
```

3. program_block::= block "."

```
begin writeln('Hi'); writeln('there!'); end .
```

4. block::= label_declarations constant_definitions type_definitions variable_declarations procedure_function_declarations statement_part

```
const n=1; m=2; begin writeln(m+n); end
```

5. program_parameters::= List(identifier)

```
in,out,waveItAllAbout
```

Declarations and Definitions

6. label_declarations::= O("label" List(label) semicolon)
7. label::= digit_sequence

```
label 1,2,3,911;
```

8. constant_definitions::= O("const" S(constant_definition) semicolon)
9. constant_definition::= identifier "=" constant
10. type_definitions::= O("type" type_definition semicolon #(type_definition semicolon))
11. type_definition::= identifier "=" type_denoter
12. variable_declarations::= O("var" variable_declaration semicolon #(variable_declaration semicolon))
13. variable_declaration::= identifier_list ":" type_denoter
14. procedure_function_declarations::= #(procedure_declaration | function_declaration) semicolon

Subprogram Declarations

15. subprogram_declaration::=procedure_declaration | function_declaration

Procedure Declarations

16. procedure_declaration::= procedure_heading semicolon directive | procedure_identification semicolon procedure_block | procedure_heading semicolon procedure_block
17. procedure_heading::= "procedure" identifier O(formal_parameter_list)
18. procedure_identification::= "procedure" procedure_identifier
19. procedure_block::= block
20. procedure_identifier::= identifier
21. directive::= letter #(letter | digit)
22. type_denoter::= type_identifier | new_type

Function Declarations

23. function_declaration::= function_heading semicolon directive | function_identification semicolon function_block | function_heading semicolon function_block
24. function_heading::= "function" identifier (formal_parameter_list |) ":" result_type
25. function_block::= block

Formal Parameters and Arguments

26. formal_parameter_list::= formal_parameter_section #(semicolon formal_parameter_section)
27. formal_parameter_section::= value_parameter_specification | variable_parameter_specification | procedural_parameter_specification | functional_parameter_specification | conformant_array_parameter_specification
28. value_parameter_specification::= identifier_list ":" type_identifier
29. variable_parameter_specification::= "var" identifier_list ":" type_identifier
30. procedural_parameter_specification::= procedure_heading
31. functional_parameter_specification::= function_heading
32. conformant_array_parameter_specification::= value_conformant_array_specification | variable_conformant_array_specification
33. value_conformant_array_specification::= identifier_list ":" conformant_array_schema
34. variable_conformant_array_specification::= "var" identifier_list ":" conformant_array_schema
35. conformant_array_schema::= packed_conformant_array_schema | unpacked_conformant_array_schema

36. packed_conformant_array_schema::= "packed" "array" O(index_type_specification) "of" type_identifier
37. unpacked_conformant_array_schema::= "array" O(index_type_specification #(semicolon index_type_specification)) "of" type_identifier | conformant_array_schema
38. index_type_specfication::= identifier ".." identifier ":" ordinal_type_identifier

Constants

39. constant::= signed_number | constant_identifier | character_string
40. unsigned_constant::= unsigned_number | character_string | constant_identifier | "nil"
41. unsigned_number::= unsigned_integer | unsigned_real
42. constant_identifier::= identifier

Types

43. type_identifier::= identifier
44. new_type::= new_ordinal_type | new_structured_type | new_pointer_type
45. result_type::= simple_type_identifier | pointer_type_identifier
46. new_ordinal_type::= enumerated_type | subrange_type
47. new_structured_type::= O("packed") unpacked_structured_type
48. new_pointer_type::= "^" domain_type
49. simple_type_identifier::= type_identifier
50. pointer_type_identifier::= type_identifier
51. enumerated_type::= identifier_list
52. subrange_type::= constant ".." constant
53. unpacked_structured_type::= array_type | record_type | set_type | file_type
54. domain_type::= type_identifier
55. array_type::= "array" #(index_type #("," index_type)) "of" component_type
56. set_type::= "set of" base_type
57. file_type::= "file of" component_type
58. index_type::= ordinal_type
59. component_type::= type_denoter
60. base_type::= ordinal_type
61. ordinal_type::= new_ordinal_type | ordinal_type_identifier
62. ordinal_type_identifier::= type_identifier
63. record_type::= "record" field_list "end"
64. record_section::= identifier_list ":" type_denoter
65. field_list::= O(fixed_part #(semicolon variant_part)
66. fixed_part::= record_section #(semicolon record_section)
67. variant_part::= "case" variant_selector "of" variant #(semicolon variant)
68. variant_selector::= O(tag_field ":") tag_type
69. variant::= case_constant_list ":" field_list
70. tag_field::= identifier
71. tag_type::= ordinal_type_identifier
72. case_constant_list::= case_constant #("," case_constant)
73. case_constant::= constant

Statements

74. procedure_statement::= procedure_identifier O(actual_parameter_list) | IO_procedure_statement
75. IO_procedure_statement::= "read" read_parameter_list | "readln" readln_parameter_list | "write" write_parameter_list | "writeln" writeln_parameter_list
76. actual_parameter_list::= actual_parameter #("," actual_parameter)
77. optional_file::=O(file_variable ",")
78. read_parameter_list::= optional_file variable_access #("," variable_access)
79. readln_parameter_list::= O(optional_file variable_access #("," variable_access))
80. write_parameter_list::= optional_file write_parameter #("," write_parameter)
81. writeln_parameter_list::= O(optional_file write_parameter #("," write_parameter))
82. actual_parameter::= expression | variable_access | procedure_identifier | function_identifier
83. file_variable::= variable_access
84. variable_access::= entire_variable | component_variable | identified_variable | buffer_varible
85. write_parameter::= expression O(":" O(":" expression))
86. statement_part::= compound_statement
87. compound_statement::= "begin" statement_sequence "end"
88. statement_sequence::= statement #(semicolon statement)
89. statement::= O(label ":") (simple_statement | structured_statement)
90. simple_statement::= empty_statement | assignment_statement | procedure_statement | goto_statement
91. structured_statement::= compound_statement | conditional_statement | repetitive_statement | with_statement
92. empty_statement::=
93. assignment_statement::= variable_access | function_identifier ":=" expression
94. goto_statement::= "goto" label
95. conditional_statement::= if_statement | case_statement
96. repetitive_statement::= repeat_statement | while_statement | for_statement
97. loop::= repeat_statement | while_statement | for_statement
98. with_statement::= "with" record_variable_list "do" statement
99. if_statement::= "if" boolean_expression "then" statement O(else_part)
100. case_statement::= "case" case_index "of" case_list_element #(semicolon case_list_element) O(semicolon |) "end"
101. repeat_statement::= "repeat" statement_sequence "until" boolean_expression
102. while_statement::= "while" boolean_expression "do" statement
103. for_statement::= "for" control_variable ":=" initial_value ("to" | "downto") final_value "do" statement
104. record_variable_list::= record_variable #("," record_variable)
105. boolean_expression::= expression
106. else_part::= "else" statement
107. case_index::= expression
108. case_list_element::= case_constant_list ":" statement
109. control_variable::= entire_variable
110. initial_value::= expression
111. final_value::= expression

Expressions and Variables

112. expression::= simple_expression #(relational_operator simple_expression)
113. function_identifier::= identifier

114. entire_variable::= variable_identifier
115. component_variable::= indexed_variable | field_designator
116. identified_variable::= pointer_variable "^"
117. buffer_variable::= file_variable "^"
118. simple_expression::= O(sign) term #(adding_operator term)
119. variable_identifier::= identifier
120. indexed_variable::= array_variable "[" O(index_expression #("," index_expression)) "]"
121. field_designator::= record_variable "." field_specifier | field_designator_identifier
122. pointer_variable::= variable_access
123. term::= factor #(multiplying_operator factor)
124. array_variable::= variable_access
125. index_expression::= expression
126. record_variable::= variable_access
127. field_specifier::= field_identifier
128. field_designator_identifier::= identifier
129. factor::= variable_access | unsigned_constant | function_designator | set_constructor | expression |
"not" factor, | bound_identifier
130. field_identifier::= identifier
131. set_constructor::= O(#(member_designator #("," member_designator)))
132. bound_identifier::= identifier
133. member_designator::= expression #(".." expression)
134. function_identification::= "function" function_identifier
135. function_designator::= function_identifier O(actual_parameter_list)

Notation

1. **O**(X)::= empty | X.
2. **#**(X)::= any number of X.
3. **S**(X)::= X #(";" X).
4. **List**(X)::= X #("," X).

Michael Oudshoorn
August 13, 2023