Приложение А. Синтаксис на езика UniPascal

Синтаксисът на езика UniPascal се задава чрез Разширените Бекус-Наурови Форми (РБНФ).

```
 1: Digit=             '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'.
 2: Letter=            '_'|'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'|
 2:                    'I'|'J'|'K'|'L'|'M'|'N'|'O'|'P'|'Q'|
 2:                    'R'|'S'|'T'|'U'|'V'|'W'|'X'|'Y'|'Z'|
 2:                        'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|
 2:                    'i'|'j'|'k'|'l'|'m'|'n'|'o'|'p'|'q'|
 2:                    'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'.
 3: ASCII_8=           Digit | Letter |
 3:                    '!'|'"'|'#'|'$'|'%'|'&'|"'"|'('|
 3:                    ')'|'*'|'+'|','|'-'|'.'|'/'|':'|
 3:                    ';'|'<'|'='|'>'|'?'|'*'|'['|'\'|
 3:                    ']'|'^'|'_'|'`'|'{'|'|'|'}'|'~'.
 4: Ident=             Letter { Letter | Digit }.
 5: QualIdent=         [Ident '.'] Ident.
 6: IdentList=         Ident { ',' Ident }.
 7: HexDigit=          Digit | 'A'| 'B'| 'C'| 'D'| 'E'| 'F'|
 7:                            'a'| 'b'| 'c'| 'd'| 'e'| 'f'.
 8: Decimal=           Digit { Digit | '_' }.
 9: IntConst=          Decimal |
 9:                    ('$' HexDigit { HexDigit | '_' }).
10: Sign=              [ '+' | '-'].
11: ScaleFactor=       ('E' | 'e') Sign Decimal.
12: RealConstant=      Decimal (('.' Decimal [ScaleFactor]) |
12:                             (['.' Decimal] ScaleFactor)).
13: SignedRealConst=   Sign RealConstant.
14: SignedIntConst=    Sign IntConst.
15: CharConst=         '"' ASCII_8 '"'|
15:                    "'" ASCII_8 "'"| '#'IntConst.
16: StringConst=       { '"' { ASCII_8 } '"' |
16:                      "'" { ASCII_8 } "'" |
16:                      CharConst } |
16:                    '""' | "''".
17: Comment=           ('{'  { ASCII_8 }  '}') |
17:                    ('(*' { ASCII_8 } '*)').
18: Program=           ProgramHeading
18:                      UsesClause
18:                    Block '.'.
19: Block=             [ Declarations ]
19:                    'begin'
19:                       Statement { ';'
19:                       Statement }
19:                    'end'.
```

```
20: Declarations=     { { LabelDeclaration } |
20:                     { ConstDeclaration } |
20:                     { TypeDeclaration }  |
20:                     { VarDeclaration }   |
20:                     { PFDeclaration } }  .
21: ProgramHeading=   'program' Ident [ '(' IdentList ')' ] ';'.
22: Label=            Ident | IntConst.
23: LabelDeclaration= 'label' Label { ',' Label } ';'.
24: ConstDeclaration= 'const' Ident '=' Constant ';' {
24:                           Ident '=' Constant ';' }.
25: Constant=         SignedRealConst | SignedIntConst |
25:                   CharConst | StringConst | ConstExpression.
26: ConstExpression=  Expression.
27: TypeDeclaration=  'type' Ident '=' Type ';' {
27:                          Ident '=' Type ';' }.
28: Type=             TypeIdent | SimpleType |
28:                   PointerType | StructuredType.
29: TypeIdent=        Ident.
30: VarDeclaration=   'var' IdentList ':' Type ';' {
30:                         IdentList ':' Type ';' }.
31: PFDeclaration=    { ProcDeclaration | FuncDeclaration }.
32: SimpleType=       OrdinalType | RealType.
33: OrdinalType=      Enumerated | SubRange | StandardType.
34: StandardType=     'integer'  | 'shortint'  | 'longint' |
34:                   'cardinal' | 'shortcard' | 'natural' |
34:                   'char'     | 'boolean'   |
34:                   'byte' | 'word' | 'longword'|.
35: RealType=         'real'.
36: Enumerated=       '(' IdentList ')'.
37: SubRange=         Constant '..' Constant.
38: StructuredType=   ['packed'] (ArrayType  |
38:                               StringType |
38:                               RecordType |
38:                               SetType    |
38:                               FileType   ).
39: ArrayType=        'array' '[' IndexType {','
39:                               IndexType } ']' 'of' Type.
40: IndexType=        OrdinalType.
41: StringType=       'string' [ '[' Constant ']' ].
42: RecordType=       'record' FieldList 'end'.
43: FieldList=        (FixedPart [';']) | (VariantPart [';']) |
43:                   (FixedPart ';' VariantPart [';']).
44: FixedPart=        IdentList ':' Type { ';'
44:                   IdentList ':' Type }.
```

```
45: VariantPart=       'case' TagField 'of'
45:                       CnstList ':' '(' [FieldList] ')' { ';'
45:                       CnstList ':' '(' [FieldList] ')' }.
46: TagField=          [Ident ':'] OrdinalTypeIdent.
47: OrdinalTypeIdent=  Ident.
48: SetType=           'set' 'of' OrdinalType.
49: FileType=          'file' [ 'of' Type ].
50: PointerType=       '^' TypeIdent.
51: Expression=        (SimpleExpression [relationOp
51:                      SimpleExpression]) |
51:                    ExpTypeCast.
52: SimpleExpression=  ['+' | '-'] Term {AdditiveOp Term}.
53: Term=              Factor {MultiplicativeOp Factor}.
54: Factor=            Constant |
54:                    VariableRef |
54:                    SetConstructor |
54:                    FunctionCall |
54:                    'not' Factor |
54:                    ( '(' Expression ')' ).
55: SetConstructor=    '[' [SetElement {',' SetElement}] ']'.
56: SetElement=        Expression [ '..' Expression ].
57: FunctionCall=      QualIdent [ ActualParamList ].
58: relationOp=        '='| '<>'| '<' | '<=' | '>'| '>='| 'in'.
59: AdditiveOp=        '+'| '-' | 'or'| 'xor'| '|'.
60: MultiplicativeOp=  '*'| '/' | 'div'| 'mod'| 'and'| '&'.
61: ExpTypeCast=       TypeIdent '(' Expression ')'.
62: VarTypeCast=       TypeIdent '(' VariableRef ')'.
63: Statement=         [Label ':'] ( SimpleStatement |
63:                                  StructStatement ).
64: SimpleStatement=   EmptyStatement | Assignment |
64:                    ProcedureCall | GotoStatement.
65: EmptyStatement=    .
66: Assignment=        (VariableRef | FuncIdent) ':=' Expression.
67: VariableRef=       VarTypeCast |
67:                    (QualIdent {'.' Ident | '^' |
67:                    '[' Expression {',' Expression} ']'}).
68: FuncIdent=         Ident.
69: ProcedureCall=     QualIdent [ ActualParamList ].
70: GotoStatement=     'goto' Label.
71: StructStatement=   CompoundStatement |
71:                    IfStatement       |
71:                    CaseStatement     |
71:                    RepetativeStat    |
71:                    WithStatement     .
72: CompoundStatement= 'begin' Statement { ';' Statement } 'end'.
```

```
73: IfStatement=      'if' Expression
73:                      'then' Statement [
73:                      'else' Statement ].
74: CaseStatement=    'case' Selector 'of'
74:                      CnstList ':' Statement {';'
74:                      CnstList ':' Statement } [';'] [
74:                   'else' ':' Statement { ';'
74:                      Statement } [';'] ]
74:                   'end'.
75: Selector=         Expression.
76: CnstList=         Constant {',' Constant }.
77: RepetativeStat=   ForStatement   |
77:                   WhileStatement |
77:                   RepeatStatement.
78: WhileStatement=   'while' Expression 'do' Statement.
79: RepeatStatement=  'repeat' Statement {';'
79:                      Statement }
79:                   'until' Expression.
80: ForStatement=     'for' Ident ':=' Expression ('to' |
80:                          'downto') Expression 'do'
80:                   Statement.
81: WithStatement=    'with' VariableRef {',' VariableRef } 'do'
81:                   Statement.
82: ProcDeclaration=  ProcHeading ';' (Block | Directive) ';'.
83: ProcHeading=      ['segment']'procedure' Ident[FormalPList].
84: Directive=        'forward' | 'external' |
84:                   ('code' IntConst {',' IntConst}).
85: FuncDeclaration=  FuncHeading ';' (Block | Directive) ';'.
86: FuncHeading=      ['segment']
86:                   'function' Ident[FormalPList]':'TypeIdent.
87: FormalPList=      '(' [ Parameter { ',' Parameter } ] ')'.
88: Parameter=        (['var'|'const'] IdentList ':' TypeIdent)|
88:                    ('var' | 'const') IdentList.
89: ActualParamList=  ['(' [Expression {',' Expression} ] ')'].
90: Unit=             'unit'Ident['('IntConst')']';' 'interface'
90:                      InterfacePart (
90:                   'implementation'
90:                      ImplmntPart |
90:                   'end') '.'.
91: InterfaceUnit=    'interface' 'unit'Ident['('IntConst')']';'
91:                      InterfacePart
91:                   'end' '.'.
92: ImplmntUnit=      'implementation' 'unit' Ident ';'
92:                   ImplmntPart '.'.
93: InterfacePart=    [ UsesClause ]   {
93:                   ConstDeclaration |
93:                   TypeDeclaration  |
93:                  VarDeclaration   |
93:                  PFDeclaration    }.
94: ImplmntPart=     [ UsesClause ]
94:                  Block.
95: UsesClause=      { 'uses' IdentList';' }.
96: Compilation=     Program| Unit| InterfaceUnit| ImplmntUnit.
```

Списък на нетерминалните символи и използването им в синтактическите правила (Non Terminal Symbols Cross Reference)

С минус е отбелязан номерът на РБНФ правилото, чрез което е определен съответният нетерминален символ. Останалите числа са номерата на РБНФ правилата, в които този нетерминален символ се използва. Вляво е дадено къде в текста на документацията е описан съответният нетерминален символ. Само два нетерминални символа са дефинирани, но не са използвани никъде в другите синтактични правила. Единият от тях - Compilation съответства на стартовия символ на граматиката. А другият - Comment се отнася за коментар.

1.6.	ASCII_8	-3 15 16 17
7.4.	ActualParamList	-89 57 69
5.2.	AdditiveOp	-59 52
3.2.1.	ArrayType	-39 38
6.1.2.	Assignment	-66 64
2.	Block	-19 18 82 85 94
6.2.3.	CaseStatement	-74 71
1.6.	CharConst	-15 16 25
6.2.3.	CnstList	-76 45 74
1.7.	Comment	-17 <unused>
9.	Compilation	-96 <unused>
6.2.1.	CompoundStatement	-72 71
2.3.	Constant	-25 24 37 41 54 76
2.3.	ConstDeclaration	-24 20 93
2.3.	ConstExpression	-26 25
1.5.	Decimal	-8 9 11 12
2.	Declarations	-20 19
1.2.	Digit	-1 3 4 7 8
7.1.	Directive	-84 82 85
6.1.1.	EmptyStatement	-65 64
3.1.1.	Enumerated	-36 33
5.3.	ExpTypeCast	-61 51
5.	Expression	-51 26 54 56 61 66 67 73 75 78 79 80 89
5.	Factor	-54 53 54
3.2.3.	FieldList	-43 42 45
3.2.5.	FileType	-49 38
3.2.3.	FixedPart	-44 43
6.2.4.3.	ForStatement	-80 77
7.3.	FormalPList	-87 83 86
7.2.	FuncDeclaration	-85 31
7.2.	FuncHeading	-86 85

6.1.2.	FuncIdent	-68 66
5.1.	FunctionCall	-57 54
6.1.4.	GotoStatement	-70 64
1.5.	HexDigit	-7 9
1.3.	Ident	-4 5 6 21 22 24 27 29 46 47 67 68 80 83 86 90 91 92
1.3.	IdentList	-6 21 30 36 44 88 95
6.2.2.	IfStatement	-73 71
8.2.	ImplmntPart	-94 90 92
8.	ImplmntUnit	-92 96
3.2.1.	IndexType	-40 39
1.5.	IntConst	-9 14 15 22 84 90 91
8.1.	InterfacePart	-93 90 91
8.	InterfaceUnit	-91 96
2.2.	Label	-22 23 63 70
2.2.	LabelDeclaration	-23 20
1.2.	Letter	-2 3 4
5.2.	MultiplicativeOp	-60 53
3.1.	OrdinalType	-33 32 40 48
3.2.3.	OrdinalTypeIdent	-47 46
2.6.	PFDeclaration	-31 20 93
7.3.	Parameter	-88 87
3.3.	PointerType	-50 28
7.1.	ProcDeclaration	-82 31
7.1.	ProcHeading	-83 82
6.1.3.	ProcedureCall	-69 64
2.	Program	-18 96
2.1.	ProgramHeading	-21 18
1.3.	QualIdent	-5 57 67 69
1.5.	RealConstant	-12 13
3.1.	RealType	-35 32
3.2.3.	RecordType	-42 38
5.2.	relationOp	-58 51
6.2.4.2.	RepeatStatement	-79 77
6.2.4.	RepetativeStat	-77 71
1.5.	ScaleFactor	-11 12
6.2.3.	Selector	-75 74
3.2.4.	SetConstructor	-55 54
3.2.4.	SetElement	-56 55
3.2.4.	SetType	-48 38
1.5.	Sign	-10 11 13 14
1.5.	SignedIntConst	-14 25
1.5.	SignedRealConst	-13 25
5.	SimpleExpression	-52 51
6.1.	SimpleStatement	-64 63
3.1.	SimpleType	-32 28
3.1.	StandardType	-34 33
6.	Statement	-63 19 72 73 74 78 79 80 81
1.6.	StringConst	-16 25

3.2.2.	StringType	-41 38
6.2.	StructStatement	-71 63
3.2.	StructuredType	-38 28
3.1.4.	SubRange	-37 33
3.2.3.	TagField	-46 45
5.	Term	-53 52
2.4.	Type	-28 27 30 39 44 49
2.4.	TypeDeclaration	-27 20 93
2.4.	TypeIdent	-29 28 50 61 62 86 88
8.	Unit	-90 96
8.4.	UsesClause	-95 18 93 94
2.5.	VarDeclaration	-30 20 93
5.3.	VarTypeCast	-62 67
6.1.2.	VariableRef	-67 54 62 66 81
3.2.3.	VariantPart	-45 43
6.2.4.1.	WhileStatement	-78 77
6.2.5.	WithStatement	-81 71

Списък на терминалните символи и използването им в синтактическите правила (Terminal Symbols Cross Reference)

Следва списък на всички терминални символи с изключение на буквите (A..Z, a..z) и цифрите (0..9). Не са включени тези терминални символи, които не са ключови думи, а именно: code, forward, string, external, както и имената на всички стандартни типове.

"	3 15 16
#	3 15
$	3 9
&	3 60
'	3 15 16
(	3 21 36 45 54 61 62 87 89
(*	17
)	3 21 36 45 54 61 62 87 89
*	3 60
*)	17
+	3 10 52 59
,	3 6 23 39 55 67 76 81 84 87 89
-	3 10 52 59
.	3 5 12 18 67 90 91 92
..	37 56
/	3 60
:	30 44 45 46 63 74 86 88
:=	66 80
;	3 19 21 23 24 27 30 30 43 44 45 72 74 79 82 85 90 91 92 95
<	3 58
<=	58
	58
=	3 24 27 58