**PLATYPUS LANGUAGE SPECIFICATION**

# 2. The PLATYPUS Lexical Specification

## 2.1 Input Elements and Tokens

< input character *> ->* *one of* ASCII characters *but not* SEOF

<input> ->

<input elements> SEOF

<input elements> ->

<input element> | <input elements> <input element>

<input element > ->

<white space > | <comment> | <token>

<token> ->

<variable identifier> | <keyword> | <floating-point literal >

| <integer literal > | <string literal> | <separator> | <operator>

## 2.2 White Space

<white space> ->

ASCII SP character (space)

| the ASCII HT character (horizontal tab)

| the ASCII VT character (vertical tab)

| the ASCII FF character (form feed)

| <line terminator>

<line terminator> ->

CR | LF| CR LF

## 2.3 Comments

<comment> ->

!! <opt\_characters in line> <line terminator>

<characters in line> ->

<comment character> | <characters in line> <comment character>

which actually means (see 1.4):

<comment> ->

!!<opt\_characters in line> <line terminator>

<opt\_characters in line>

<characters in line> | ϵ

<characters in line> ->

<comment character> | <characters in line> <comment character>

<comment character> ->

<input character> but not <line terminator>

## 2.4 Variable Identifiers

<variable identifier> ->

<arithmetic variable identifier> | <string variable identifier>

<arithmetic variable identifier> ->

<letter> <opt\_letters or digits>

<letters or digits> ->

<letter or digit> | <letters or digits> <letter or digit>

<letter> -> *one of*

a … z A ... Z (uppercase and lowercase ASCII Latin letters A–Z)

<letter or digit> -> *one of*

a … z A...Z 0…9 (ASCII digits 0-9)

<string variable identifier> ->

<arithmetic variable identifier>$

## 2.5 Keywords

*<*keyword> ->

PLATYPUS | IF | THEN | ELSE | WHILE | REPEAT | READ | WRITE | TRUE | FALSE

## 2.6 Integer Literals

<integer literal> ->

<decimal integer literal> | < hexadecimal integer literal>

< decimal integer literal> ->

0 | <non zero digit> <opt\_digits>

<digits> ->

<digit> | <digits> <digit>

<digit> ->

1. | <non zero digit>

<non zero digit> ->

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<hexadecimal integer literal> ->

0x < hexadecimal digits>

< hexadecimal digit> ->

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

< hexadecimal digits> ->

< hexadecimal digit> | < hexadecimal digits> < hexadecimal digit>

## 2.7 Floating-point Literals

<floating-point literal> ->

<decimal integer literal> **.** <opt\_digits>

## 2.8 String Literals

<string literal> ->

“<opt\_string characters>”

<string characters> ->

<input character> | <string characters> <input character>

## 2.9 Separators

<separator> *->*

( | ) | { | } | , | ; | “ | **.**

## 2.10 Operators

<operator> ->

< arithmetic operator > | <string concatenation operator>

| < relational operator> | < logical operator >

*| <* assignment operator *>*

<arithmetic operator> ->

+ | - | \* | /

<string concatenation operator> ->

#

<relational operator> ->

> | < | == | <>

<logical operator> ->

.AND. | .OR.

<assignment operator> ->

=

# 3 The PLATYPUS Syntactic Specification

## 3.1 PLATYPUS Program

<program> ->

PLATYPUS {<opt\_statements>} SEOF

<statements> ->

<statement> | <statements> <statement>

## 3.2 Statements

<statement> ->

<assignment statement>

| <selection statement>

| <iteration statement>

| <input statement>

| <output statement>

### 3.2.1 Assignment Statement

<assignment statement> ->

<assignment expression> EOS

< assignment expression> ->

AVID = <arithmetic expression>

| SVID = <string expression>

### 3.2.2 Selection Statement (the if statement)

<selection statement> ->

IF TRUE (<conditional expression>) THEN {<opt\_statements>}

ELSE {<opt\_statements>};

### 3.2.3 Iteration Statement (the loop statement)

<iteration statement> ->

WHILE <pre-condition> (<conditional expression>)

REPEAT {<statements>};

<pre-condition> ->

TRUE | FALSE

### 3.2.4 Input Statement

<input statement> ->

READ (<variable list>);

<variable list> ->

<variable identifier> | <variable list>, <variable identifier>

### 3.2.5 Output Statement

<output statement> ->

WRITE (<opt\_variable list> | STR\_T);

**3.3 Expressions**

### 3.3.1 Arithmetic Expression

<arithmetic expression> - >

<unary arithmetic expression>

| <additive arithmetic expression>

<unary arithmetic expression> ->

- <primary arithmetic expression>

| + <primary arithmetic expression>

<additive arithmetic expression> ->

<additive arithmetic expression> + <multiplicative arithmetic expression> | <additive arithmetic expression> - <multiplicative arithmetic expression> | <multiplicative arithmetic expression>

<multiplicative arithmetic expression> ->

<multiplicative arithmetic expression> \* <primary arithmetic expression>

| <multiplicative arithmetic expression> / <primary arithmetic expression> | <primary arithmetic expression>

<primary arithmetic expression> ->

AVID\_T

| FPL\_T

| INL\_T

| (<arithmetic expression>)

### 3.3.2 String Expression

<string expression> ->

<primary string expression>

| <primary string expression> # <string expression>

<primary string expression> ->

SVID\_T

| STR\_T

### 3.3.3 Conditional Expression

<conditional expression> ->

<logical OR expression>

<logical OR expression> ->

<logical AND expression>

| <logical OR expression>.OR. <logical AND expression>

<logical AND expression> ->

<relational expression>

| <logical AND expression> .AND. <relational expression>

### 3.3.4 Relational Expression

<relational expression> ->

<primary a\_relational expression> == <primary a\_relational expression>

| <primary a\_relational expression> <> <primary a\_relational expression>

| <primary a\_relational expression> > <primary a\_relational expression>

| <primary a\_relational expression> < <primary a\_relational expression>

| <primary s\_relational expression> == <primary s\_relational expression>

| <primary s\_relational expression> <> <primary s\_relational expression>

| <primary s\_relational expression> > <primary s\_relational expression>

| <primary s\_relational expression> < <primary s\_relational expression>

<primary a\_relational expression> ->

FPL\_T

| AVID\_T

| INL\_T

<primary s\_relational expression> ->

<primary string expression>