PL/306/2 language description

Fran Jarnjak, Vladimir Vacic

1. PL/306/2 accepts the following characters:

   Letters: a . . z A . . Z $ @ #
   Digits: 0 . . 9
   Punctuation: ( ) ~ + - * / % : ; , < > = ! & |

2. PL/306/2 is case insensitive.

3. In PL/306/2 a keyword or an identifier is a string of characters starting with a letter followed by one or more letters or digits. A keyword or an identifier cannot be longer than 16 characters, and cannot span more than one line.

4. In PL/306/2 a number is a string of digits starting with one or more digits followed by an optional dot that can be followed by one or more digits. A number ending with a dot will be interpreted as a real number with its decimal part equal to 0.


6. Data types are INTEGER and FLOAT.

7. PL/306/2 treats the following characters or combination thereof as operators:

   assignment operator: :=
   relational operators: < > =
   logical operators: ! | &
   arithmetic operators: + - * / %
   special operators: ( ) ~ : ; ,

8. Each statement in PL/306/2 is terminated with a semicolon.

9. Each statement starting with ~ is considered to be a comment. Remainder of the line after ~ is ignored.
10. **Variable declarations**

Each variable needs to be declared at a beginning of the program, outside any procedure block. General form of a variable declaration is:

```
DECLARE (variable, variable, ..., variable) DATATYPE;
```

An example variable declaration is:

```
DECLARE (length, width, result) FLOAT;
DECLARE (items) INTEGER;
```

11. **Assignment statement**

Assignment statement consists of a list of one or more variables of the same type, followed by the assignment operator `:=`, and the value being assigned. General form of an assignment statement is:

```
variable, variable, ..., variable := value;
```

An example of an assignment statement is:

```
length, width := 3.14;
items := 4;
```

12. **An identifier at the beginning of a line, followed by a colon `:` is considered to be a label.** A label has to be followed by a statement. General form of a label is:

```
label: statement;
```

An example of a label is:

```
count: items := items + 1;
```

13. **Jumps** in flow of control in PL/306/2 are performed through a combination of a keyword **GOTO** and a label name. General form of a **GOTO** statement is:

```
GOTO label;
```

An example of a **GOTO** statement is:

```
GOTO count;
```

14. **Procedure** block starts with an identifier followed by a colon and the keyword **PROCEDURE**, list of parameters enclosed in a pair of brackets and a semicolon. Procedure block ends with the keyword **END** followed by the same identifier used to start the procedure block. The aforementioned identifier is considered to be the procedure name. General form of a PL/306/2 procedure is:
identifier: **PROCEDURE** (variable, variable, ..., variable);
    statement;
    statement;
    ...
    statement;
**END** identifier;

An example procedure is:

area: **PROCEDURE** (length, width);
    result := length * width;
**END** area;

If a procedure parameter’s name coincides with a previously declared variable, the parameter will be used in the body of the procedure.

15. A procedure can be called from the body of the program and/or from another procedure using the following general form:

identifier(variable, variable, ..., variable);
identifier(value, value, ..., value);

where identifier is the name of the procedure and the variables and/or values are going to be passed as the parameters in the procedure call. Values and variables can be intermixed in no particular order.

An example procedure call is:

area(length, width);
area(2.5, width);
area(1.5, 6.7);

16. The **body of the program** starts with the keyword **START** and ends with the keyword **END**.

The body of the program may contain a list of statements and/or procedure calls.

Keyword **START** is considered to be the main entry point. General form of the program body is:

```
**START**;
    statement;
    statement;
    procedure call;
    statement;
    ...
    procedure call;
    statement;
**END**;
```
An example of the program body is:

```
START;
    length := 4.5;
    width := 5;
    area(length, width);
END;
```

17. The *structure of a program* is as follows:

Variable declaration block;
Procedure declaration;
Body of the program;

General form of a program is as follows:

```
DECLARE (variable, variable, ..., variable) DATATYPE;
...
DECLARE (variable, variable, ..., variable) DATATYPE;

identifier: PROCEDURE (variable, variable, ..., variable);
    statement;
    statement:
    ...
    statement;
END identifier;
...

identifier: PROCEDURE (variable, variable, ..., variable);
    statement;
    statement:
    ...
    statement;
END identifier;

START;
    statement;
    statement;
    procedure call;
    statement;
    ...
    procedure call;
    statement;
END;
```

An example program looks like this:

```
DECLARE (length, width, result) FLOAT;
DECLARE (items) INTEGER;

area: PROCEDURE (length, width);
    result := length * width;
end area;
```
START;
    length := 4.5;
    width := 5;
    area(length, width);
END;

18. Keyword **STOP** terminates the program at any point.

19. Keyword **GET** followed by a list of one or more variables enclosed by a pair of brackets is considered to be an *input statement*. The source of the input data is the operator’s console. General form of an input statement looks like this:

   GET(variable, variable, ... , variable);

   An example input statement is:

   GET(items);
   GET(length, width);

20. Keyword **PUT** followed by an optional keyword **SKIP** and a list of one or more variables or constants enclosed by a pair of brackets is considered to be an *output statement*. The target of the output is the operator’s console. Keyword **SKIP** is used to enforce a new line character after each outputted variable. A general form of an output statement looks like this:

   PUT [SKIP] (variable, value, ... , variable);

   An example of an input statement is:

   PUT(items);
   PUT SKIP (length, width);
   PUT SKIP (1, 2, 3, items);

21. A *conditional statement* begins with the keyword **IF** followed by a condition, a statement preceded by a keyword **THEN**, and an optional statement preceded by a keyword **ELSE**. The conditional statement ends with the keyword **ENDIF**. If the condition is satisfied, the statement preceded by the keyword **THEN** will be executed. Otherwise, it will not be executed. If the condition is not satisfied, statement preceded by the keyword **ELSE** will be executed, if present. General form of the conditional statement is:

   IF condition
       THEN statement;
       [ ELSE statement; ]
   ENDIF;

   An example conditional statement is:

   IF result = 0
       THEN PUT(0);
ELSE PUT(1);
ENDIF;