SLOVENSKI STANDARD
SIST EN 27185:1997

01-december-1997

Programming languages - Pascal (ISO/IEC 7185:1990)

Programming languages - Pascal (ISO/IEC 7185:1990)

Programmiersprachen - Pascal (ISO/IEC 7185:1990)

Langages de programmation - Pascal (ISO/IEC 7185:1990)

Ta slovenski standard je istoveten z: EN 27185:1992

ICS:

35.060	Jeziki, ki se uporabljajo v informacijski tehniki in tehnologiji	Languages used in information technology

SIST EN 27185:1997 en

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 27185:1992

June 1992

UDC 681.3.04:681.3.06:800.92PASCAL

Descriptors: Data processing, programming languages, Pascal, computer programs, specifications

English version

Programming languages - Pascal (ISO/IEC 7185-1990)

Langages de programmation - Pascal (ISO/IEC 7185-1990)

Programmiersprachen - Pascal (ISO/IEC 7185-1990)

This European Standard was approved by CEN on 1992-06-18. CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CEN member.

The European Standards exist in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the Central Secretariat has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CEN

European Committee for Standardization
Comité Européen de Normalisation
Europäisches Komitee für Normung

Central Secretariat: rue de Stassart,36 B-1050 Brussels

FOREWORD

The Technical Board has decided to submit the

International Standard ISO/IEC 7185:1990 "Programming languages - Pascal (ISO/IEC 7815:1990)"

for Formal Vote. The standard was accepted.

For the time being, this document exists only in English.

National Standards identical to this European Standard shall be published at the latest by 92-12-31 and conflicting national standards shall be withdrawn at the latest by 92-12-31.

According to the CEN/CENELEC Common Rules, the following countries are bound to implement this standard: Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxemburg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, United Kingdom.

ENDORSEMENT NOTICE

The text of the ISO/IEC 7185:1990 was approved by CEN as a European Standard without any modification.

INTERNATIONAL STANDARD

ISO/IEC 7185

Second edition
1990-10-15

Information technology — Programming languages — Pascal

(Revision of ISO 7185 : 1983)

Technologies de l'information — Langages de programmation — Pascal

(Révision de l'ISO 7185 : 1983)

Reference number
ISO/IEC 7185 : 1990 (E)

Contents

Page

1 Scope 1

2 Normative reference 1

3 Definitions 1

4 Definitional conventions 2

5 Compliance 3

5.1 Processors 3

5.2 Programs 5

6 Requirements 5

6.1 Lexical tokens 5

6.2 Blocks, scopes and activations 8

6.3 Constant - definitions 11

6.4 Type - definitions 11

6.5 Declarations and denotations of variables 21

6.6 Procedure and function declarations 24

6.7 Expressions 38

6.8 Statements 42

6.9 Input and output 49

6.10 Programs 55

Annexes

A Collected syntax 58

B Cross - references 64

C Required identifiers 71

D Errors 72

E Implementation - defined features 76

F Implementation - dependent features 78

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 7185 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

This second edition cancels and replaces the first edition (ISO 7185 : 1983).

Annexes A, B, C, D, E and F are for information.

Introduction

This International Standard provides an unambiguous and machine independent definition of the programming language Pascal. Its purpose is to facilitate portability of Pascal programs for use on a wide variety of data processing systems.

Language history

The computer programming language Pascal was designed by Professor Niklaus Wirth to satisfy two principal aims

a) to make available a language suitable for teaching programming as a systematic discipline based on certain fundamental concepts clearly and naturally reflected by the language;

b) to define a language whose implementations could be both reliable and efficient on then-available computers.

However, it has become apparent that Pascal has attributes that go far beyond these original goals. It is now being increasingly used commercially in the writing of both system and application software. This International Standard is primarily a consequence of the growing commercial interest in Pascal and the need to promote the portability of Pascal programs between data processing systems.

In drafting this International Standard the continued stability of Pascal has been a prime objective. However, apart from changes to clarify the specification, two major changes have been introduced.

a) The syntax used to specify procedural and functional parameters has been changed to require the use of a procedure or function heading, as appropriate (see **6.6.3.1**); this change was introduced to overcome a language insecurity.

b) A fifth kind of parameter, the conformant-array-parameter, has been introduced (see **6.6.3.7**). With this kind of parameter, the required bounds of the index-type of an actual-parameter are not fixed, but are restricted to a specified range of values.

Project history

In 1977, a working group was formed within the British Standards Institution (BSI) to produce a standard for the programming language Pascal. This group produced several working drafts, the first draft for public comment being widely published early in 1979. In 1978, BSI's proposal that Pascal be added to ISO's program of work was accepted, and the ISO Pascal Working Group (then designated ISO/TC97/SC5/WG4) was formed in 1979. The Pascal standard was to be published by BSI on behalf of ISO, and this British Standard referenced by the International Standard.

In the USA, in the fall of 1978, application was made to the IEEE Standards Board by the IEEE Computer Society to authorize project 770 (Pascal). After approval, the first meeting was held in January 1979.

In December of 1978, X3J9 convened as a result of a SPARC (Standards Planning and Requirements Committee) resolution to form a US TAG (Technical Advisory Group) for the ISO Pascal standardization effort initiated by the UK. These efforts were performed under X3 project 317.

In agreement with IEEE representatives, in February of 1979, an X3 resolution combined the X3J9 and P770 committees into a single committee called the Joint X3J9/IEEE-P770 Pascal Standards Committee. (Throughout, the term JPC refers to this committee.) The first meeting as JPC was held in April 1979.

The resolution to form JPC clarified the dual function of the single joint committee to produce a dpANS and a proposed IEEE Pascal standard, identical in content.

ANSI/IEEE770X3.97-1983, American National Standard Pascal Computer Programming Language, was approved by the IEEE Standards Board on September 17, 1981, and by the American National Standards

Institute on December 16, 1982. British Standard BS6192, Specification for Computer programming language Pascal, was published in 1982, and International Standard 7185 (incorporating BS6192 by reference) was approved by ISO on December 1, 1983. Differences between the ANSI and ISO standards are detailed in the Foreword of ANSI/IEEE770X3.97-1983.

In 1985, the ISO Pascal Working Group (then designated ISO/TC97/SC22/WG2, now ISO/IEC JTC1/SC22/WG2) was reconvened after a long break. An Interpretations Subgroup was formed, to interpret doubtful or ambiguous portions of the Pascal standards. As a result of the work of this subgroup, and also of the work on the Extended Pascal standard being produced by WG2 and JPC, BS6192/ISO7185 was revised and corrected during 1988/89; it is expected that ANSI/IEEE770X3.97-1983 will be replaced by the revised ISO 7185.

The major revisions to BS6192:1982 to produce the new ISO 7185 are:

a) resolution of the differences with ANSI/IEEE770X3.97-1983;

b) relaxation of the syntax of real numbers, to allow "digit sequences" rather than "unsigned integers" for the various components;

c) in the handling of "end-of-line characters" in text files;

d) in the handling of run-time errors.

This page intentionally left blank

Information technology — Programming languages — Pascal

1 Scope

1.1

This International Standard specifies the semantics and syntax of the computer programming language Pascal by specifying requirements for a processor and for a conforming program. Two levels of compliance are defined for both processors and programs.

1.2

This International Standard does not specify

a) the size or complexity of a program and its data that will exceed the capacity of any specific data processing system or the capacity of a particular processor, nor the actions to be taken when the corresponding limits are exceeded;

b) the minimal requirements of a data processing system that is capable of supporting an implementation of a processor for Pascal;

c) the method of activating the program-block or the set of commands used to control the environment in which a Pascal program is transformed and executed;

d) the mechanism by which programs written in Pascal are transformed for use by a data processing system;

e) the method for reporting errors or warnings;

f) the typographical representation of a program published for human reading.

2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the standard listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 646:1983, *Information processing—ISO 7-bit coded character set for information interchange.*

3 Definitions

For the purposes of this International Standard, the following definitions apply.

NOTE — To draw attention to language concepts, some terms are printed in italics on their first mention or at their defining occurrence(s) in this International Standard.

3.1 Error

A violation by a program of the requirements of this International Standard that a processor is permitted to leave undetected.

NOTES

1 If it is possible to construct a program in which the violation or non-violation of this International Standard requires knowledge of the data read by the program or the implementation definition of implementation-defined features, then violation of that requirement is classified as an error. Processors may report on such violations of the requirement without such knowledge, but there always remain some cases that require execution, simulated execution, or proof procedures with the required knowledge. Requirements that can be verified without such knowledge are not classified as errors.

2 Processors should attempt the detection of as many errors as possible, and to as complete a degree as possible. Permission to omit detection is provided for implementations in which the detection would be an excessive burden.

3.2 Extension

A modification to clause **6** of the requirements of this International Standard that does not invalidate any program complying with this International Standard, as defined by **5.2**, except by prohibiting the use of one or more particular spellings of identifiers (see **6.1.2** and **6.1.3**).

3.3 Implementation-defined

Possibly differing between processors, but defined for any particular processor.

3.4 Implementation-dependent

Possibly differing between processors and not necessarily defined for any particular processor.

3.5 Processor

A system or mechanism that accepts a program as input, prepares it for execution, and executes the process so defined with data to produce results.

NOTE — A processor may consist of an interpreter, a compiler and run-time system, or another mechanism, together with an associated host computing machine and operating system, or another mechanism for achieving the same effect. A compiler in itself, for example, does not constitute a processor.

4 Definitional conventions

The metalanguage used in this International Standard to specify the syntax of the constructs is based on Backus-Naur Form. The notation has been modified from the original to permit greater convenience of description and to allow for iterative productions to replace recursive ones. Table 1 lists the meanings of the various metasymbols. Further specification of the constructs is given by prose and, in some cases, by equivalent program fragments. Any identifier that is defined in clause **6** as a required identifier shall denote the corresponding required entity by its occurrence in such a program fragment. In all other respects, any such program fragment is bound by any pertinent requirement of this International Standard.

A meta-identifier shall be a sequence of letters and hyphens beginning with a letter.

A sequence of terminal and nonterminal symbols in a production implies the concatenation of the text that they ultimately represent. Within **6.1** this concatenation is direct; no characters shall intervene. In all other parts of this International Standard the concatenation is in accordance with the rules set out in **6.1**.

Table 1 — Metalanguage symbols

Metasymbol	Meaning
=	Shall be defined to be
>	Shall have as an alternative definition
|	Alternatively
.	End of definition
[x]	0 or 1 instance of x
{ x }	0 or more instances of x
(x | y)	Grouping: either of x or y
‘xyz’	The terminal symbol xyz
meta-identifier	A nonterminal symbol

The characters required to form Pascal programs shall be those implicitly required to form the tokens and separators defined in **6.1**.

Use of the words *of*, *in*, *containing*, and *closest-containing*, when expressing a relationship between terminal or nonterminal symbols, shall have the following meanings

—the x *of* a y: refers to the x occurring directly in a production defining y;

—the x *in* a y: is synonymous with ‘the x of a y’;

—a y *containing* an x: refers to any y from which an x is directly or indirectly derived;

—the y *closest-containing* an x: that y containing an x and not containing another y containing that x.

These syntactic conventions are used in clause **6** to specify certain syntactic requirements and also the contexts within which certain semantic specifications apply.

In addition to the normal English rules for hyphenation, hyphenation is used in this International Standard to form compound words that represent meta-identifiers, semantic terms, or both. All meta-identifiers that contain more than one word are written as a unit with hyphens joining the parts. Semantic terms ending in “type” and “variable” are also written as one hyphenated unit. Semantic terms representing compound ideas are likewise written as hyphenated units, e.g., digit-value, activation-point, assignment-compatible, and identifying-value.

NOTES are included in this International Standard only for purposes of clarification, and aid in the use of the standard. NOTES are informative only and are not a part of the International Standard.

Examples in this International Standard are equivalent to NOTES.

5 Compliance

There are two levels of compliance, level 0 and level 1. Level 0 does not include conformant-array-parameters. Level 1 does include conformant-array-parameters.

5.1 Processors

A processor complying with the requirements of this International Standard shall

a) if it complies at level 0, accept all the features of the language specified in clause **6**, except for **6.6.3.6 e)**, **6.6.3.7**, and **6.6.3.8**, with the meanings defined in clause **6**;

b) if it complies at level 1, accept all the features of the language specified in clause **6** with the meanings defined in clause **6**;

c) not require the inclusion of substitute or additional language elements in a program in order to accomplish a feature of the language that is specified in clause **6**;

d) be accompanied by a document that provides a definition of all implementation-defined features;

e) be able to determine whether or not the program violates any requirements of this International Standard, where such a violation is not designated an error, report the result of this determination to the user of the processor before the execution of the program-block, if any, and shall prevent execution of the program-block, if any;

f) treat each violation that is designated an error in at least one of the following ways

1) there shall be a statement in an accompanying document that the error is not reported, and a note referencing each such statement shall appear in a separate section of the accompanying document;

2) the processor shall report the error or the possibility of the error during preparation of the program for execution and in the event of such a report shall be able to continue further processing and shall be able to refuse execution of the program-block;

3) the processor shall report the error during execution of the program;

and if an error is reported during execution of the program, the processor shall terminate execution; if an error occurs within a statement, the execution of that statement shall not be completed;

NOTE — 1 This means that processing will continue up to or beyond execution of the program at the option of the user.

g) be accompanied by a document that separately describes any features accepted by the processor that are prohibited or not specified in clause **6**: such extensions shall be described as being 'extensions to Pascal as specified by ISO/IEC 7185';

h) be able to process, in a manner similar to that specified for errors, any use of any such extension;

i) be able to process, in a manner similar to that specified for errors, any use of an implementation-dependent feature.

NOTE — 2 The phrase 'be able to' is used in **5.1** to permit the implementation of a switch with which the user may control the reporting.

A processor that purports to comply, wholly or partially, with the requirements of this International Standard shall do so only in the following terms. A *compliance statement* shall be produced by the processor as a consequence of using the processor or shall be included in accompanying documentation. If the processor complies in all respects with the requirements of this International Standard, the compliance statement shall be

<*This processor*> complies with the requirements of level <*number*> of ISO/IEC 7185.

If the processor complies with some but not all of the requirements of this International Standard then it shall not use the above statement, but shall instead use the following compliance statement

<*This processor*> complies with the requirements of level <*number*> of ISO/IEC 7185, with the following exceptions: <*followed by a reference to, or a complete list of, the requirements of the International Standard with which the processor does not comply*>.

In both cases the text <*This processor*> shall be replaced by an unambiguous name identifying the processor, and the text <*number*> shall be replaced by the appropriate level number.

NOTE — 3 Processors that do not comply fully with the requirements of the International Standard are not required to give full details of their failures to comply in the compliance statement; a brief reference to accompanying documentation that contains a complete list in sufficient detail to identify the defects is sufficient.

5.2 Programs

A program conforming with the requirements of this International Standard shall

a) if it conforms at level 0, use only those features of the language specified in clause **6**, except for **6.6.3.6 e)**, **6.6.3.7**, and **6.6.3.8**;

b) if it conforms at level 1, use only those features of the language specified in clause **6**; and

c) not rely on any particular interpretation of implementation-dependent features.

NOTES

1 A program that complies with the requirements of this International Standard may rely on particular implementation-defined values or features.

2 The requirements for conforming programs and compliant processors do not require that the results produced by a conforming program are always the same when processed by a compliant processor. They may be the same, or they may differ, depending on the program. A simple program to illustrate this is

```
program x(output); begin writeln(maxint) end.
```

6 Requirements

6.1 Lexical tokens

NOTE — The syntax given in this subclause describes the formation of lexical tokens from characters and the separation of these tokens and therefore does not adhere to the same rules as the syntax in the rest of this International Standard.

6.1.1 General

The lexical tokens used to construct Pascal programs are classified into special-symbols, identifiers, directives, unsigned-numbers, labels, and character-strings. The representation of any letter (upper case or lower case, differences of font, etc.) occurring anywhere outside of a character-string (see **6.1.7**) shall be insignificant in that occurrence to the meaning of the program.

```
letter = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j'
       | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't'
       | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' .

digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .
```

6.1.2 Special-symbols

The special-symbols are tokens having special meanings and are used to delimit the syntactic units of the language.

```
special-symbol = '+' | '−' | '*' | '/' | '=' | '<' | '>' | '[' | ']'
               | '.' | ',' | ':' | ';' | '↑' | '(' | ')'
               | '<>' | '<=' | '>=' | ':=' | '..' | word-symbol .

word-symbol = 'and' | 'array' | 'begin' | 'case' | 'const' | 'div'
            | 'do' | 'downto' | 'else' | 'end' | 'file' | 'for'
            | 'function' | 'goto' | 'if' | 'in' | 'label' | 'mod'
            | 'nil' | 'not' | 'of' | 'or' | 'packed' | 'procedure'
            | 'program' | 'record' | 'repeat' | 'set' | 'then'
            | 'to' | 'type' | 'until' | 'var' | 'while' | 'with' .
```