International Standard
Norme internationale

2382/15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION•МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ•ORGANISATION INTERNATIONALE DE NORMALISATION

Data processing — Vocabulary — Part 15: Programming languages

First edition — 1985-11-01

Traitement de l'information — Vocabulaire — Partie 15: Langages de programmation

Première édition — 1985-11-01

UDC/CDU 681.3 : 001.4

Ref. No./Réf. nº : ISO 2382/15-1985 (E/F)

Descriptors : data processing, programming languages, vocabulary./**Descripteurs** : traitement de l'information, langages de programmation, vocabulaire.

Price based on 17 pages/Prix basé sur 17 pages

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 2382/15 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

Printed in Switzerland

Avant-propos

L'ISO (Organisation internationale de normalisation) est une fédération mondiale d'organismes nationaux de normalisation (comités membres de l'ISO). L'élaboration des Normes internationales est confiée aux comités techniques de l'ISO. Chaque comité membre intéressé par une étude a le droit de faire partie du comité technique créé à cet effet. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'ISO participent également aux travaux.

Les projets de Normes internationales adoptés par les comités techniques sont soumis aux comités membres pour approbation, avant leur acceptation comme Normes internationales par le Conseil de l'ISO. Les Normes internationales sont approuvées conformément aux procédures de l'ISO qui requièrent l'approbation de 75 % au moins des comités membres votants.

La Norme internationale ISO 2382/15 a été élaborée par le comité technique ISO/TC 97, *Systèmes de traitement de l'information*.

L'attention des utilisateurs est attirée sur le fait que toutes les Normes internationales sont de temps en temps soumises à révision et que toute référence faite à une autre Norme internationale dans le présent document implique qu'il s'agit, sauf indication contraire, de la dernière édition.

Imprimé en Suisse

Contents

Page

0 Introduction 1

Section one: General

1 Scope and field of application 2

2 Principles and rules followed

2.1 Definition of an entry 2

2.2 Organization of an entry 2

2.3 Classification of entries 3

2.4 Selection of terms and wording of definitions 3

2.5 Multiple meanings 3

2.6 Abbreviations 3

2.7 Use of parentheses 4

2.8 Use of brackets 4

2.9 Use of terms printed in italic typeface in definitions and use of asterisk 4

2.10 Spelling 4

2.11 Organization of the alphabetical index 4

Section two: Terms and definitions

15 Programming languages

15.01 General objects 5

15.02 General properties and mechanisms 6

15.03 Data objects 8

15.04 Properties and mechanisms related to data objects 9

15.05 Processing objects 11

15.06 Properties and mechanisms related to processing objects 11

Alphabetical indexes

English 14

French 16

Sommaire

Page

0 Introduction 1

Section un : Généralités

1 Objet et domaine d'application 2

2 Principes d'établissement et règles suivies

2.1 Définition de l'article 2

2.2 Constitution d'un article 2

2.3 Classification des articles 3

2.4 Choix des termes et des définitions 3

2.5 Pluralité de sens ou polysémie 3

2.6 Abréviations 3

2.7 Emploi des parenthèses 4

2.8 Emploi des crochets 4

2.9 Emploi dans les définitions de termes imprimés en caractères italiques et de l'astérisque 4

2.10 Mode d'écriture et orthographe 4

2.11 Constitution de l'index alphabétique 4

Section deux : Termes et définitions

15 Langages de l'information

15.01 Objets généraux des langages de programmation 5

15.02 Propriétés et procédés généraux 6

15.03 Objets représentant des données 8

15.04 Propriétés et procédés relatifs aux objets représentant des données ... 9

15.05 Objets représentant des traitements 11

15.06 Propriétés et procédés relatifs aux objets représentant des traitements 11

Index alphabétiques

Anglais 14

Français 16

INTERNATIONAL STANDARD
NORME INTERNATIONALE

ISO 2382/15-1985 (E/F)

Data processing — Vocabulary — Part 15: Programming languages

0 Introduction

Data processing gives rise to numerous international exchanges of both intellectual and material nature. These exchanges often become difficult, either because of the great variety of terms used in various fields or languages to express the same concept, or because of the absence or imprecision of the definitions of useful concepts.

To avoid misunderstandings and to facilitate such exchanges, it is essential to clarify the concepts, to select terms to be used in various languages or in various countries to express the same concept and to establish definitions providing satisfactory equivalents for the various terms in different languages.

ISO 2382 was initially based mainly on the usage to be found in the *Vocabulary of Information Processing*, established and published by the International Federation for Information Processing and the International Computation Centre, and in the *USA Standard Vocabulary for Information Processing* and its revised edition, established and published by the American National Standards Institute (formerly known as the American Standards Association). Published and draft International Standards relating to data processing and documentation from other international organizations (such as the International Telecommunication Union and the International Electrotechnical Commission) together with published and draft national standards have been considered.

The purpose of ISO 2382 is to provide definitions that are rigorous, uncomplicated and which can be understood by all concerned. The scope of each concept defined has been chosen to provide a definition that is suitable for general application. In those circumstances where a restricted application is concerned, the definition may need to be more specific.

However, while it is possible to maintain the self-consistency of individual parts, the reader is warned that the dynamics of language and the problems associated with the standardization and maintenance of vocabularies may introduce duplications and inconsistencies between parts.

Traitement des données — Vocabulaire — Partie 15: Langages de programmation

0 Introduction

Le traitement des données est à l'origine de multiples échanges intellectuels et matériels sur le plan international. Ceux-ci souffrent souvent des difficultés provoquées par la diversité des termes utilisés pour exprimer la même notion dans des langues ou dans des domaines différents, ou encore de l'absence ou de l'imprécision des définitions pour les notions les plus utiles.

Pour éviter des malentendus et faciliter de tels échanges, il paraît essentiel de préciser les notions, de choisir les termes à employer dans les différentes langues et dans les divers pays pour exprimer la même notion, et d'établir pour ces termes des définitions équivalentes dans chaque langue.

L'ISO 2382 a été basée à l'origine principalement sur l'usage tel qu'il a été relevé d'une part dans le *Vocabulary of Information Processing* établi et publié par l'International Federation for Information Processing et le Centre International de Calcul et d'autre part dans le *USA Standard Vocabulary for Information Processing* dans son édition révisée établie et publiée par l'American National Standards Institute (connu auparavant sous l'appellation d'American Standards Association). Les Normes internationales publiées ou au stade de projets concernant le traitement de l'information et la documentation émanant d'autres organisations internationales (telles que l'Union internationale des télécommunications et la Commission électrotechnique internationale).

Le but de l'ISO 2382 est de procurer des définitions rigoureuses, simples et compréhensibles par tous les intéressés. La portée de chaque notion a été choisie de façon que sa définition puisse avoir la valeur la plus générale. Toutefois il est parfois nécessaire de restreindre une notion à un domaine plus étroit, et de lui donner alors une définition plus spécifique.

D'autre part, si l'on peut assurer la cohérence interne de chaque partie prise individuellement, la cohérence des diverses parties entre elles est plus difficile à atteindre. Le lecteur ne doit pas s'en étonner: la dynamique des langues et les problèmes de l'établissement et de la révision des normes de vocabulaires peuvent être à l'origine de quelques répétitions ou contradictions entre des parties qui ne sont pas toutes préparées et publiées simultanément.

Section one: General

1 Scope and field of application

ISO 2382 is intended to facilitate international communication in data processing. It presents, in two languages, terms and definitions of selected concepts relevant to the field of data processing and identifies relationships between the entries.

In order to facilitate their translation into other languages, the definitions are drafted so as to avoid, as far as possible, any peculiarity attached to a language.

ISO 2382 deals with the main areas of data processing, including the principal processes and types of equipment used, the organization and the representation of data, the programming and operation of computers, peripheral equipment and data communication as well as particular applications.

This part of ISO 2382 (which will comprise some twenty-five parts) deals with programming language concepts that are most commonly used in the data processing community.

2 Principles and rules followed

2.1 Definition of an entry

Section two comprises a number of entries. Each entry consists of a set of essential elements that includes an index number, one term or several synonymous terms, and a phrase defining one concept. In addition, an entry may include examples, notes or illustrations to facilitate understanding of the concept.

Occasionally, the same term may be defined in different entries, or two or more concepts may be covered by one entry, as described in 2.5 and 2.8 respectively.

Other terms such as **vocabulary, concept, term** and **definition**, are used in this International Standard with the meaning defined in ISO/R 1087, *Vocabulary of terminology.*

2.2 Organization of an entry

Each entry contains the essential elements defined in 2.1 and, if necessary, additional elements. The entry may contain the following elements in the following order:

a) an index number (common for all languages in which this International Standard is published);

Section un: Généralités

1 Objet et domaine d'application

L'ISO 2382 a pour objet de faciliter les échanges internationaux dans cette technique et ses applications. À cet effet, elle présente un ensemble bilingue de termes et de définitions ayant trait à des notions choisies dans ce domaine, et définit les relations pouvant exister entre les différentes notions.

Les définitions ont été établies de manière à éviter les particularismes propres à une langue donnée, en vue de faciliter leur transposition dans les langues autres que celles ayant servi à la rédaction initiale.

L'ISO 2382 traite des principaux domaines du traitement des données, des principaux procédés et types de machines employés, de l'organisation et de la représentation des données, de la programmation et de l'exploitation des ordinateurs, des dispositifs périphériques et du transfert des données, ainsi que de certaines applications particulières.

La présente partie de l'ISO 2382, qui en comprendra environ vingt-cinq, définit les termes des langages de programmation qui sont le plus couramment utilisés parmi les informaticiens.

2 Principes d'établissement et règles suivies

2.1 Définition de l'article

La section deux est composée d'un certain nombre d'articles. Chaque article est composé d'un ensemble d'éléments essentiels comprenant le numéro de référence, le terme ou plusieurs termes synonymes et la définition d'une notion couverte par ces termes. Cet ensemble peut être complété par des exemples, des notes, des schémas ou des tableaux destinés à faciliter la compréhension de la notion.

Parfois, le même terme peut être défini dans des articles différents, ou bien deux notions ou davantage peuvent être couvertes par un seul article: voir respectivement en 2.5 et 2.8.

D'autres termes tels que **vocabulaire, notion, terme, définition** sont employés dans la présente Norme internationale avec le sens qui leur est donné dans l'ISO/R 1087, *Vocabulaire de la terminologie.*

2.2 Constitution d'un article

Chaque article contient les éléments essentiels définis en 2.1 et, si nécessaire, des éléments supplémentaires. L'article peut donc comprendre dans l'ordre les éléments suivants:

a) un numéro de référence (le même, quelle que soit la langue de publication de la présente Norme internationale);

b) the term or the generally preferred term in the language. The absence of a generally accepted term for the concept in the language is indicated by a symbol consisting of five points (.....); a row of dots may be used to indicate, in a term, a word to be chosen in each particular case;

c) the preferred term in a particular country (identified according to the rules of ISO/R 639, *Symbols for languages, countries and authorities*);

d) the abbreviation for the term;

e) permitted synonymous term(s);

f) the text of the definition (see 2.4);

g) one or more examples with the heading "Example(s)";

h) one or more notes specifying particular cases in the field of application of the concepts, with the heading "NOTE(S)";

j) a picture, a diagram, or a table.

2.3 Classification of entries

A two-digit serial number is assigned to each part of this International Standard beginning with **01** for "**fundamental terms**".

The entries are classified in groups to each of which is assigned a four-digit serial number. The first two digits are those of the part of this International Standard.

Each entry is assigned a six-digit index number. The first four digits are those of the part of this International Standard and the group.

In order that versions of this International Standard in various languages may be related, the numbers assigned to parts, groups and entries are the same for all languages.

2.4 Selection of terms and wording of definitions

The selection of terms and the wording of definitions have, as far as possible, followed established usage. When there were contradictions, solutions agreeable to the majority have been sought.

2.5 Multiple meanings

When, in one of the working languages, a given term has several meanings, each meaning is given a separate entry in order to facilitate translation into other languages.

2.6 Abbreviations

As indicated in 2.2, abbreviations in current use are given for some terms. Such abbreviations are not used in the texts of the definitions, examples or notes.

b) le terme ou le terme préféré en général dans la langue. L'absence dans une langue, de terme consacré ou à conseiller pour exprimer une notion est indiquée par un symbole consistant en cinq points de suspension (.....), les points de suspension peuvent être employés pour désigner dans un terme, un mot à choisir dans chaque cas particulier;

c) le terme préféré dans un certain pays (identifié selon les règles de l'ISO/R 639, *Indicatifs de langue, de pays et d'autorité*);

d) l'abréviation pouvant être employée à la place du terme;

e) le termes ou les termes admis comme synonymes;

f) le texte de la définition (voir 2.4);

g) un ou plusieurs exemples, précédés du titre «Exemple(s)»;

h) une ou plusieurs notes précisant le domaine d'application de la notion, précédées du titre «NOTE(S)»;

j) un schéma ou un tableau, pouvant être communs à plusieurs articles.

2.3 Classification des articles

Chaque partie de la présente Norme internationale reçoit un numéro d'ordre à deux chiffres, en commençant par **01** pour le chapitre «**Termes fondamentaux**».

Les articles sont répartis en groupes qui reçoivent chacun un numéro d'ordre à quatre chiffres, les deux premiers chiffres étant ceux du numéro de partie de la présente Norme internationale.

Chaque article est repéré par un numéro de référence à six chiffres, les quatre premiers chiffres étant ceux du numéro de partie de la présente Norme internationale et de groupe.

Les numéros des parties, des groupes et des articles sont les mêmes pour toutes les langues, afin de mettre en évidence les correspondances des versions de la présente Norme internationale.

2.4 Choix des termes et des définitions

Les choix qui ont été faits pour les termes et leurs définitions sont, dans toute la mesure du possible, compatibles avec les usages établis. Lorsque certains usages apparaissent contradictoires, des solutions de compromis ont été retenues.

2.5 Pluralité de sens ou polysémie

Lorsque, dans l'une des langues de travail, un même terme peut prendre plusieurs sens, ces sens sont définis dans des articles différents pour faciliter l'adaptation du vocabulaire dans d'autres langues.

2.6 Abréviations

Comme indiqué en 2.2, des abréviations littérales d'usage courant, au moins en anglais, sont indiquées pour certains termes. De telles abréviations ne sont pas employées dans le corps des définitions, exemples ou notes.

2.7 Use of parentheses

In some terms, a word or words printed in bold typeface are placed between parentheses. These words are part of the complete term, but they may be omitted when use of the abridged term in a technical context does not introduce ambiguity. In the text of another definition, example, or note in this International Standard, such a term is used only in its complete form.

In some entries, the terms are followed by words in parentheses in normal typeface. These words are not a part of the term but indicate directives for the use of the term, its particular field of application, or its grammatical form.

2.8 Use of brackets

When several closely related terms can be defined by texts that differ only in a few words, the terms and their definitions are grouped in a single entry. The words to be substituted in order to obtain the different meanings are placed in brackets, i.e. [ ], in the same order in the term and in the definition. In order to avoid uncertainty regarding the words to be substituted, the last word that according to the above rule could be placed in front of the opening bracket, is wherever possible, placed inside the bracket and repeated for each alternative.

2.9 Use of terms printed in italic typeface in definitions and use of asterisk

A term printed in italic typeface in a definition, an example, or a note is defined in another entry in this International Standard, which may be in another part. However, the term is printed in italic typeface only the first time it occurs in each entry.

Italic typeface is also used for other grammatical forms of a term, for example, plurals of nouns and participles of verbs.

The basic forms of all terms printed in italic typeface are listed in the index at the end of the part (see 2.11).

An asterisk is used to separate terms printed in italic typeface when two such terms are defined in separate entries and directly follow each other (or are separated only by a punctuation sign).

Words or terms that are printed in normal typeface are to be understood as defined in current dictionaries or authoritative technical vocabularies.

2.10 Spelling

In the English version of this International Standard, terms, definitions, examples and notes are given in the spelling preferred in the USA. Other correct spellings may be used without violating this International Standard.

2.11 Organization of the alphabetical index

For each language used, an alphabetical index is provided at the end of each part. The index includes all terms defined in the part. Multiple-word terms appear in alphabetical order under each of their key words.

2.7 Emploi des parenthèses

Dans certains termes, un ou plusieurs mots imprimés en caractères gras sont placés entre parenthèses. Ces mots font partie intégrante du terme complet, mais peuvent être omis lorsque le terme ainsi abrégé peut être employé dans un contexte technique déterminé sans que cette omission introduise d'ambiguïté. Un tel terme n'est employé dans le texte d'une autre définition, d'un exemple ou d'une note, dans la présente Norme internationale, que sous sa forme complète.

Dans certains articles, les termes définis sont suivis par des expressions imprimées en caractères normaux et placées entre parenthèses. Ces expressions ne font pas partie du terme mais indiquent des prescriptions d'emploi, précisent un domaine d'application particulier ou indiquent une forme grammaticale.

2.8 Emploi des crochets

Lorsque plusieurs termes étroitement apparentés peuvent être définis par des textes presque identiques, à quelques mots près, les termes et leurs définitions ont été groupés en un seul article. Les mots à substituer à ceux qui les précèdent pour obtenir les différents sens sont placés entre crochets (c'est-à-dire [ ]) dans le même ordre dans le terme et dans la définition. En vue d'éviter toute incertitude sur les mots à remplacer, le dernier mot qui, suivant la règle ci-dessus pourrait être placé devant le crochet d'ouverture, est placé, si possible, à l'intérieur des crochets et répété à chaque occasion.

2.9 Emploi dans les définitions de termes imprimés en caractères italiques et de l'astérisque

Dans le texte d'une définition, d'un exemple ou d'une note, tout terme imprimé en caractères italiques a le sens défini dans un autre article de la présente Norme internationale, qui peut se trouver dans une autre partie. Cependant le terme est imprimé en caractères italiques uniquement la première fois qu'il apparaît dans chaque article.

Les caractères italiques sont également utilisés pour les autres formes grammaticales du terme, par exemple, les noms au pluriel et les verbes au participe.

La liste des formes de base de tous les termes imprimés en caractères italiques est fournie dans l'index à la fin de la partie (voir 2.11).

L'astérisque sert à séparer les termes imprimés en caractères italiques quand deux termes se rapportent à des articles séparés et se suivent directement (ou bien sont séparés simplement par un signe de ponctuation).

Les mots ou termes écrits en caractères normaux doivent être compris dans le sens qui leur est donné dans les dictionnaires courants ou vocabulaires techniques faisant autorité.

2.10 Mode d'écriture et orthographe

Dans la version anglaise de la présente Norme internationale, les termes, définitions, exemples et notes sont écrits suivant l'orthographe prévalant aux États-Unis. D'autres orthographes correctes peuvent être utilisées sans violer la présente Norme internationale.

2.11 Constitution de l'index alphabétique

Pour chaque langue de travail, un index alphabétique est fourni à la fin de chaque partie. L'index comprend tous les termes définis dans la partie. Les termes composés de plusieurs mots sont répertoriés alphabétiquement suivant chacun des mots constituants caractéristiques «(mots clés)».

Section two: Terms and definitions

15 Programming languages

15.01 General objects

15.01.01
lexical unit
(lexical) token
A language construct that, by convention, represents an elemental unit of meaning.

Examples: A *literal* such as "2G5", a *keyword* such as PRINT, a separator such as a semicolon.

15.01.02
identifier
A *lexical unit* that names a language object.

Examples: The names of *variables, arrays, records, labels, procedures* etc.

NOTE — An identifier usually consists of a letter optionally followed by letters, digits or other characters.

15.01.03
keyword
A *lexical unit* that, in certain contexts, characterizes some language construction.

Example: In some contexts, IF characterizes an if-statement.

NOTE — A keyword normally has the form of an *identifier*.

15.01.04
reserved word
A *keyword* that may not be used as an *identifier*.

NOTE — In Ada[1] all keywords are reserved words; FORTRAN has no reserved words.

15.01.05
literal
A *lexical unit* that directly represents a value.

Examples: 14 represents the integer fourteen, "APRIL" represents the string of characters APRIL, 3.0005E2 represents the number 300.05.

15.01.06
statement
A language construct that represents a step in a sequence of actions or a set of *declarations*.

1) Ada is a registered trademark of the US Department of Defense, Ada Joint Program Office.

Section deux: Termes et definitions

15 Langages de programmation

15.01 Objets généraux des langages de programmation

15.01.01
unité lexicale
entité lexicale
Élément de langage représentant, par convention, une unité signifiante élémentaire.

Exemples: Un *libellé* tel que «2G5»; un *mot-clé* tel que PRINT; un séparateur tel que le point-virgule.

15.01.02
identificateur
Unité lexicale servant à désigner un objet de langage.

Exemples: Les noms de *variables*, de *tableaux*, d'*articles*, d'*étiquettes*, de *procédures*, etc.

NOTE — Un identificateur est fréquemment constitué par une lettre suivie, facultativement, par des lettres, des chiffres ou d'autres caractères.

15.01.03
mot-clé
Unité lexicale qui, dans certains contextes, caractérise un élément de langage.

Exemple: Dans certains contextes, le mot-clé «IF» caractérise une instruction conditionnelle.

NOTE — Un mot-clé est généralement formé comme un *identificateur*.

15.01.04
mot réservé
Mot-clé qui ne peut être employé comme *identificateur*.

NOTE — En Ada[1], tous les mots-clés sont des mots réservés; en FORTRAN, il n'y a pas de mot réservé.

15.01.05
libellé
littéral
Unité lexicale qui représente explicitement une valeur.

Exemples: 14 représentant l'entier quatorze, «AVRIL» représentant la chaîne de caractères AVRIL, 3.0005E2 représentant le nombre 300,05.

15.01.06
instruction
énoncé
Élément de langage représentant un pas d'une séquence d'actions ou un ensemble de *déclarations*.

1) Ada est une marque déposée du Département US de la défense, Ada Joint Program Office.

15.01.07
compound statement
A *statement* constructed by sequencing statements.

NOTE — Most often the statements are grouped together by some syntactic device.

15.01.08
block (in programming languages)
A *compound statement* that coincides with the *scope* of at least one of the *declarations* contained within it.

NOTE — A block may also specify *storage* allocation or segment *programs* for other purposes.

15.01.09
module
program unit
A language construct that consists of *procedures* or *data* declarations* and that can interact with other such constructions.

Examples: In Ada, a package; in FORTRAN, a program unit; in PL/I, an external procedure.

15.01.10
encapsulated type
A *module* representing an abstract *data type*.

NOTE — An encapsulated type hides the representation of its values but permits operations on the values by other modules.

Example: A *stack* processing module.

15.01.11
program (in programming languages)
A logical assembly of one or more interrelated *modules*.

15.01.12
comment
A language construct for the inclusion of text in a *program* and having no impact on the *execution* of the program.

NOTE — Comments are used to explain certain aspects of the program.

15.01.13
environment description
A language construct for the description of features that are not part of a *program* but are relevant to its *execution*.

Examples: Machine characteristics, special properties of *files*, interfaces with other programs.

15.02 General properties and mechanisms

15.02.01
declaration
The mechanism for establishing a language object.

NOTE — A declaration normally involves attaching an *identifier*, and allocating attributes, to the language object concerned.

15.01.07
instruction composée
Instruction constituée d'une suite d'instructions.

NOTE — Le plus souvent, les limites de cette suite d'instructions sont déterminées par la syntaxe.

15.01.08
bloc (en langages de programmation)
Instruction composée dont les limites coïncident avec celles de la *portée* d'au moins une *déclaration* qui y est contenue.

NOTE — Un bloc peut accessoirement déterminer l'attribution de *mémoire* ou segmenter des *programmes* à d'autres fins.

15.01.09
module de programme
unité de programme
Élément de langage, composé de *procédures* ou de *déclarations* de *données*, pouvant se combiner avec d'autres éléments de même nature.

Exemples: Le «package» en Ada, l'«unité de programme» en FORTRAN, la «procédure externe» en PL/I.

15.01.10
type encapsulé
Module représentant un *type de donnée* abstrait.

Exemple: Module de traitement de *pile*.

NOTE — Un type encapsulé cache la représentation de ses valeurs mais permet à d'autres modules d'opérer sur elles.

15.01.11
programme (en langages de programmation)
Module de programme, ou ensemble de modules de programme reliés entre eux.

15.01.12
commentaire
Élément de langage, permettant d'insérer dans un *programme* des textes quelconques, sans incidence sur l'*exécution* du programme.

NOTE — Les commentaires servent à expliquer certains aspects d'un programme.

15.01.13
description d'environnement
Élément de langage servant à décrire des caractéristiques qui ne font pas partie du *programme* mais concernent son *exécution*.

Exemples: Caractéristiques de la machine, propriétés particulières des *fichiers*, relations avec d'autres programmes.

15.02 Propriétés et procédés généraux

15.02.01
déclaration
Procédé permettant de créer un objet de langage.

NOTE — Une déclaration implique normalement l'affectation d'un *identificateur* et d'attributs à l'objet de langage concerné.

15.02.02
default (adjective)
Pertaining to an attribute, value, or option that is assumed when none is explicitly specified.

15.02.03
implicit declaration
A *declaration*, caused by the occurrence of an *identifier* and in which the attributes are determined by *default*.

15.02.04
built-in
predefined
Pertaining to a language object that is *declared* by the definition of the *programming language*.

Examples: The built-in function SIN in PL/I, the predefined *data type* INTEGER in FORTRAN.

15.02.05
scope (of a declaration)
That portion of a *program* within which the *declaration* applies.

NOTES

1 "The scope of an identifier" is often used as a synonym for "the scope of its declaration".

2 A language object may not be referable throughout its scope since it may be hidden by the declaration of the same *identifier* in an inner *block*.

15.02.06
local
Pertaining to the relationship between a language object and a *block* such that the language object has a *scope* contained in that block.

15.02.07
global
Pertaining to the relationship between a language object and a *block* such that the language object has a *scope* extending beyond that block but contained within an encompassing block.

15.02.08
external
Pertaining to a language object that has a *scope* that extends beyond one *module*.

Example: The *entry* names of a module are external.

15.02.09
static
Pertaining to properties that can be established before the *execution* of a *program*.

Example: The length of a fixed-length *variable* is static.

15.02.02
par défaut (qualitatif)
implicite
Qualifie un attribut, une valeur ou une option sous-entendus, en l'absence d'autres précisions.

15.02.03
déclaration implicite
Déclaration provoquée par l'apparition d'un *identificateur* et dans laquelle les attributs sont fournis *par défaut*.

15.02.04
incorporé
intrinsèque
Qualifie un objet de langage dont la *déclaration* figure dans la définition du *langage de programmation*.

Exemples: La fonction incorporée SIN en PL/I; le *type de données* incorporé INTEGER en FORTRAN.

15.02.05
portée (d'une déclaration)
champ d'application (d'une déclaration)
Partie du *programme* à laquelle s'applique la *déclaration*.

NOTES

1 L'expression «la portée d'un identificateur» est souvent employée pour désigner le champ d'application de la déclaration de cet *identificateur*.

2 La *référence* à un objet de langage n'est pas forcément possible sur toute sa portée car une déclaration du même *identificateur* dans un *bloc* interne peut cacher cet objet.

15.02.06
local (adjectif)
Par rapport à un *bloc*, qualifie un objet de langage dont la *portée* est toute entière située à l'intérieur de ce bloc.

15.02.07
global
Par rapport à un *bloc*, qualifie un objet de langage dont la *portée* s'étend au-delà de ce bloc mais reste comprise à l'intérieur d'un bloc plus vaste.

15.02.08
externe
Qualifie un objet de langage dont le *champ d'application* s'étend au-delà d'un *module de programme*.

Exemple: Les noms des *points d'entrée* d'un module de programme sont des noms externes.

15.02.09
statique (adjectif)
Qualifie des particularités qui peuvent être déterminées avant l'*exécution* d'un *programme*.

Exemple: La longueur d'une *variable* de longueur fixe est statique.

15.02.10
dynamic
Pertaining to properties that can only be established during the *execution* of a *program*.

Example: The length of a varying length *data* object is *dynamic*.

15.02.11
lifetime
Of a language object, that portion of the *execution time* during which the object exists.

15.02.12
reference
A language construct designating a *declared* language object.

Example: An *Identifier*.

15.02.13
(name) qualification
A mechanism for referencing a component of a language object by means of a *reference* to the object and an *identifier* declared* for the component.

Examples: Used for referencing *record* components (B OF A in COBOL), members of a *library*, language objects in a *module*.

15.02.14
uniform referencing
A property of a *programming language* such that two or more language constructs for referencing are of the same form.

Examples: Language constructs for *name qualification* and *indirect references*; language constructs for *subscripting* and *actual parameters*.

15.03 Data objects

15.03.01
variable
A language object that may take different values, one at a time.

NOTE — The values of a variable are usually restricted to a certain *data type*.

15.03.02
constant
A language object that takes only one specific value.

15.03.03
aggregate
A structured collection of *data* objects, forming a *data type*.

15.02.10
dynamique (qualificatif)
Qualifie des particularités qui ne peuvent être déterminées que durant l'*exécution* d'un *programme*.

Exemple: La longueur d'un objet de longueur variable est *dynamique*.

15.02.11
durée de vie
Partie du *temps d'exécution* durant laquelle un objet de langage considéré existe.

15.02.12
référence
Élément de langage désignant un objet de langage déclaré.

Exemple: Un *identificateur*.

15.02.13
qualification par nom
Procédé permettant de désigner un constituant d'un objet, au moyen de la *référence* à l'objet et d'un *identificateur* attribué à ce constituant.

Exemples: La façon de désigner un composant d'un *article* (B OF A en COBOL), un élément d'une *bibliothèque*, un objet de langage dans un *module de programme*.

15.02.14
référence uniforme
Propriété d'un *langage de programmation* telle que plusieurs éléments de ce langage servant à référencer soient de même forme.

Exemples: Un langage utilisant les mêmes éléments pour la *qualification par nom* et la *référence indirecte*, ou encore pour les indices et les *paramètres effectifs* est dit à référence uniforme.

15.03 Objets relatifs aux données

15.03.01
variable
Objet de langage susceptible de prendre des valeurs différentes, mais une seule à la fois.

NOTE — Les valeurs que peut prendre une variable se limitent généralement à celles d'un seul *type de données*.

15.03.02
constante
Objet de langage qui ne prend qu'une seule valeur spécifiée.

15.03.03
agrégat
Ensemble structuré d'objets relatifs aux *données* défini comme un seul *type de données*.

15.03.04
array (in programming languages)
An *aggregate* that consists of *data* objects, with identical attributes, each of which may be uniquely referenced by *subscripting*.

15.03.05
record (in programming languages)
An *aggregate* that consists of *data* objects, with possibly different attributes, which usually have *identifiers* attached to them.

NOTE — In some programming languages, records are called structures.

15.03.06
variant part (of a record)
A part of a *record* the *data* objects of which are defined in alternative ways.

NOTE — Both the number and composition of data objects can vary.

15.03.07
area (in programming languages)
A space together with a mechanism for inserting *data* objects into it, and for accessing and for deleting data objects from it.

15.03.08
(formal) parameter
dummy argument
A language object, the *identifier* of which appears in an *entry* of a *procedure*, that is associated with the corresponding *actual parameter* specified by the *procedure call* for use in the *execution* of the procedure.

15.03.09
actual parameter
(actual) argument
A language object that appears in a *procedure call* and that is associated with the corresponding *formal parameter* for use in the *execution* of the *procedure*.

15.04 Properties and mechanisms related to data objects

15.04.01
(data) type
A set of values together with a set of permitted operations.

15.04.02
picture (in programming languages)
A language construct that describes a *data type* by means of a model *character string* literal*.

15.04.03
format
A language construct that specifies the *representation*, in *character* form, of *data* objects in a *file*.

15.03.04
tableau (en langages de programmation)
Agrégat dont chaque constituant possède des *attributs* identiques, et peut être désigné sans ambiguïté par *indiçage*.

15.03.05
article (en langages de programmation)
Agrégat dont les constituants peuvent avoir des attributs différents et comporter un *identificateur*.

NOTE — Dans certains langages de programmation les articles sont appelés structures.

15.03.06
partie variable (d'un article)
Partie d'un *article*, dont les objets sont définis de plusieurs façons.

NOTE — Le nombre et la composition des objets peuvent différer.

15.03.07
zone (en langages de programmation)
Espace auquel est associé une règle permettant d'y introduire des *données*, d'avoir accès à celles-ci et de les supprimer.

15.03.08
paramètre formel
paramètre fictif
Objet de langage dont *l'identificateur* apparaît dans le *point d'entrée* d'une *procédure* et qui, lors de chaque exécution de celle-ci, est associé au *paramètre réel* correspondant fourni par l'*appel de procédure*.

15.03.09
paramètre réel
paramètre effectif
Objet de langage présent dans un *appel de procédure*, et qui est associé au *paramètre formel* correspondant pour l'*exécution* de la *procédure*.

15.04 Propriétés et procédés relatifs aux objets relatifs aux données

15.04.01
type de données
Ensemble de valeurs associé à l'ensemble des opérations permises sur ces valeurs.

15.04.02
image (en langages de programmation)
Élément de langage décrivant un *type de données* au moyen d'un *libellé* modèle composé d'une *chaîne de caractères*.

15.04.03
(description de) format
Élément de langage spécifiant la représentation, sous forme de *caractères*, des objets désignant les *données* dans un *fichier*.

15.04.04
subscripting
A mechanism for referencing an *array* element by means of an array *reference* and one or more *expressions* that, when evaluated, denote the position of the element.

NOTE — This term also applies to the use of the mechanism.

15.04.05
indirect referencing
A mechanism for referencing via a *data* object the value of which points to the referenced language object.

NOTES

1 This term also applies to the use of the mechanism.

2 The referencing may be done along a chain of data objects in which each data object, except the last, points to the next, the last data object pointing to the referenced language object.

15.04.06
data flow
The transfer of *data* between *constants, variables* and *files* accomplished by the *execution* of *statements, procedures, modules,* or *programs.*

15.04.07
assignment
A mechanism to give a value to a *variable*.

NOTE — This term also applies to the use of the mechanism.

15.04.08
assignment by name
An *assignment* of a *record* value to a record *variable* pertaining only to those components with matching *identifiers.*

15.04.09
(to) initialize
To give a value to a *data* object at the beginning of its *lifetime.*

15.04.10
automatic storage allocation
A mechanism for allocating space to *data* objects only for the duration of the *execution* of their *scope*.

NOTE — Automatic storage allocation is one form of dynamic *storage* allocation; another form is *program* controlled storage allocation.

15.04.11
assumed-size aggregate
An *aggregate* formal parameter* that takes some or all of its *subscript* ranges from a corresponding *actual parameter.*

15.04.04
indiçage
Procédé permettant d'obtenir la *référence* d'un élément d'un *tableau* au moyen de la référence du tableau et d'*expressions* dont l'évaluation fournit la position de cet élément dans le tableau.

NOTE — Ce terme désigne également l'utilisation de ce procédé.

15.04.05
référence indirecte
Procédé permettant de désigner un objet de langage au moyen de la valeur d'une *donnée*.

NOTES

1 Ce terme désigne également l'utilisation de ce procédé.

2 La référence peut être obtenue au moyen d'une chaîne d'objets dans laquelle chaque objet est un pointeur vers l'objet suivant, sauf le dernier qui pointe vers l'objet référencé.

15.04.06
flux de données
Transfert de *données* entre *constantes, variables* ou *fichiers*, résultant de l'*exécution* d'*instructions*, de *procédures*, de *modules de programmes* ou de *programmes.*

15.04.07
assignation
affectation
Procédé permettant d'attribuer une valeur à une *variable* ou à un *agrégat.*

NOTE — Ce terme désigne également l'utilisation de ce procédé.

15.04.08
assignation par nom
affectation par nom
Assignation d'une valeur aux *variables* d'un *article* à partir des valeurs des variables d'un autre article ayant respectivement les mêmes *identificateurs.*

15.04.09
initialiser
Donner une valeur à un objet relatif à une *donnée*, au début de sa *durée de vie.*

15.04.10
attribution automatique de mémoire
Action d'affecter temporairement de la place en mémoire à des objets relatifs à des *données*, seulement pour la durée d'*exécution* correspondant à leur *portée.*

NOTE — L'attribution automatique est une des formes d'attribution dynamique de *mémoire*, par opposition à l'attribution de mémoire à la demande du *programme* utilisateur.

15.04.11
agrégat de taille implicite
Agrégat employé comme *paramètre formel* et dont la gamme d'indices est déterminée par le *paramètre réel* correspondant.

15.04.12
adjustable-size aggregate
An *aggregate* formal parameter* with some or all of its *subscript* ranges *dynamic.*

15.04.13
editing (in programming languages)
Transforming values to the representations specified by a given *format.*

15.05 Processing objects

15.05.01
expression
A language construct for *computing* a value from one or more *operands.*

NOTE — Operands may be *literals, identifiers, array references, function calls* etc.

15.05.02
procedure
A *block*, with or without *formal parameters*, the *execution* of which is invoked by means of a *procedure call.*

15.05.03
function (procedure)
A *procedure* that, when *executed*, yields a value and the *procedure call* of which may be used as an *operand* in an *expression.*

Example: The function SIN yields the value sin X when called with SIN(X).

15.05.04
asynchronous procedure
A *procedure* that can be *executed* concurrently with the calling part of the *program.*

15.05.05
critical section
In an *asynchronous procedure*, a part that cannot be *executed* simultaneously with a certain part of the same or another asynchronous procedure.

NOTE — That part of the same or other asynchronous procedure is also a critical section.

15.05.06
label (in programming languages)
A language construct naming a *statement* and including an *identifier.*

15.06 Properties and mechanisms related to processing objects

15.06.01
execution sequence
The order of the *execution* of *statements* and parts of statements of a *program.*

15.04.12
agrégat de taille ajustable
Agrégat employé comme *paramètre formel* et dont certaines des limites de la gamme d'indices sont *dynamiques.*

15.04.13
mise en forme (en langages de programmation)
Transformation de valeurs en leur représentation selon une *description de format* spécifiée.

15.05 Objets représentant des traitements

15.05.01
expression
Élément de langage permettant le *calcul* d'une valeur à partir d'un ou de plusieurs *opérandes.*

NOTE — Les opérandes peuvent être des *libellés*, des *identificateurs*, des *références* à des *tableaux*, des *appels de fonction*, etc.

15.05.02
procédure
Bloc, avec ou sans *paramètres formels*, dont l'exécution peut être provoquée par un *appel de procédure.*

15.05.03
fonction (en langages de programmation)
Procédure dont l'*exécution* produit une valeur et dont l'*appel de procédure* peut être utilisé comme *opérande* dans une *expression.*

Exemple: La fonction SIN donne la valeur sin X quand elle est appelée par SIN(X).

15.05.04
procédure asynchrone
Procédure qui peut être *exécutée* en concurrence avec la partie du *programme* qui l'appelle.

15.05.05
section critique
Dans une *procédure asynchrone*, partie qui ne peut pas être *exécutée* en même temps qu'une partie déterminée d'une autre procédure asynchrone.

NOTE — Cette partie déterminée de l'autre procédure asynchrone est également une section critique.

15.05.06
étiquette (en langages de programmation)
Élément de langage désignant une *instruction* et comportant un *identificateur.*

15.06 Propriétés et procédés relatifs aux objets relatifs aux traitements

15.06.01
séquence d'exécution
Ordre dans lequel les *instructions* et parties d'instructions d'un *programme* sont exécutées.

15.06.02
control flow
An abstraction of all possible paths the *execution sequence* may take through a *program*.

NOTE — The control flow can be represented by a control flow graph.

15.06.03
unconditional statement
A *statement* that specifies only one possible *execution sequence*.

15.06.04
conditional construct
A language construct that specifies several different *execution sequences*.

Examples: A CASE *statement*; an IF statement; a conditional expression in ALGOL.

15.06.05
loop construct
A language construct that specifies an iteration in the *execution sequence*.

Examples: DO loops in FORTRAN; FOR loops in ALGOL; PERFORM loops in COBOL; DO WHILE loops in PL/I.

15.06.06
(procedure) call
A language construct for invoking the *execution* of a *procedure*.

NOTE — A procedure call usually includes an *entry* name and possible *actual parameters*.

15.06.07
entry (of a procedure)
A language construct within a *procedure*, designating the start of an *execution sequence* of the procedure.

NOTE — A procedure may have more than one entry; each entry usually includes an *identifier*, called the entry name, and possibly *formal parameters*.

15.06.08
parameter association
The association of *formal parameters* with the corresponding *actual parameters* that are specified by a *procedure call*.

15.06.09
return (from a procedure)
A language construct within a *procedure* designating the end of an *execution sequence* in the procedure.

NOTE — Usually the execution sequence continues from the point of the *procedure call*.

15.06.02
flux de commande
Ensemble des cheminements que peut suivre la *séquence d'exécution* d'un *programme*.

NOTE — Le flux de commande peut être représenté graphiquement.

15.06.03
instruction inconditionnelle
Instruction qui détermine un seul chemin possible pour la *séquence d'exécution*.

15.06.04
élément conditionnel
Élément de langage spécifiant plusieurs *séquences d'exécution* différentes.

Exemples: Une *instruction* CASE, une instruction IF; une expression conditionnelle en ALGOL.

15.06.05
élément de boucle
Élément de langage spécifiant une itération dans la *séquence d'exécution*.

Exemples: Boucles DO en FORTRAN, FOR en ALGOL, PERFORM en COBOL, DO WHILE en PL/I.

15.06.06
appel (de procédure)
Élément de langage destiné à provoquer l'*exécution* d'une *procédure*.

NOTE — Un appel de procédure comporte généralement un nom de *point d'entrée* et, éventuellement, des *paramètres effectifs*.

15.06.07
point d'entrée (d'une procédure)
entrée (d'une procédure)
Élément de langage, dans une *procédure*, désignant le début d'une *séquence d'exécution* dans la procédure.

NOTE — Une procédure peut comporter plusieurs points d'entrée; chaque point d'entrée comporte généralement un *identificateur*, appelé nom du point d'entrée, et éventuellement *des paramètres formels*.

15.06.08
association de paramètres
Association entre *paramètres formels* et *paramètres réels* correspondants spécifiés par un *appel de procédure*.

15.06.09
point de sortie (d'une procédure)
retour
Élément de langage, dans une *procédure* et marquant la fin d'une *séquence d'exécution* dans cette procédure.

NOTE — Généralement la séquence d'exécution se poursuit à partir du point d'*appel de procédure*.

15.06.10
side effect
Any external effect caused by the *execution* of a *function procedure* other than that of yielding the result value.

15.06.11
branch construct
A language construct specifying a choice between different *execution sequences* by means of *label* references.*

15.06.12
exception (in programming languages)
A special situation which may arise during *execution*, which is considered abnormal, which may cause a deviation from the normal *execution sequence,* and for which facilities exist in the programming language to define, raise, recognize, ignore and handle it.

Examples: (ON-) condition in PL/I, exception in Ada.

15.06.13
(operator) precedence
An order relation defining the sequence of the application of *operators* within an *expression.*

15.06.14
conversion (in programming languages)
The transformation between values which *represent* the same *data* item but belong to different *data types*.

NOTE — Information may be lost due to conversion since accuracy of data representation varies among different data types.

15.06.15
activation
The representation of a *procedure* created by the invocation of that procedure.

15.06.16
connection
A mechanism that enables interaction among *modules*, particularly *procedure calls* to *asynchronous procedures*.

Examples: In COBOL, an ENABLE *statement* establishes a communication connection, an OPEN statement establishes an *input-output* connection.

15.06.10
effet oblique
effet secondaire
effet de bord
Tout effet, autre que la production d'une valeur résultante, provoqué par l'*exécution* d'une *fonction.*

15.06.11
élément de branchement
Élément de langage permettant de choisir entre différentes *séquences d'exécution* grâce à des *références* à des *étiquettes.*

15.06.12
exception (en langages de programmation)
Situation particulière considérée comme anormale pouvant se produire pendant l'*exécution* d'un *programme*, qui peut provoquer un changement dans la séquence normale d'exécution et pour laquelle le langage fournit des moyens permettant de la définir, la faire apparaître, la reconnaître, la traiter ou ne pas en tenir compte.

Exemples: (ON-) condition en PL/I, exception en Ada.

15.06.13
priorité (des opérateurs)
Ordre défini par une règle pour l'exécution des opérations spécifiées par les *opérateurs* dans une *expression.*

15.06.14
conversion (en langages de programmation)
Transformation d'une valeur pour *représenter* la même *donnée* selon des *types de données* différents.

NOTE — La conversion peut provoquer une perte d'information, du fait que la précision de la représentation varie selon le type de données.

15.06.15
activation
Représentation d'une *procédure*, créée par un appel particulier à cette procédure.

15.06.16
connexion
Procédé qui permet des interactions entre *modules de programme*, particulièrement des *appels* de *procédures asynchrones.*

Exemples: En COBOL, une *instruction* ENABLE établit une connexion de communication, une instruction OPEN établit une connexion d'*entrée-sortie*.

English alphabetical index

A

activation activation 15.06.15
adjustable-size adjustable-size aggregate 15.04.12
aggregate aggregate 15.03.03
adjustable-size aggregate 15.04.12
assumed-size aggregate 15.04.11
allocation automatic storage allocation 15.04.10
area area (in a programming language) ... 15.03.07
argument (actual) argument 15.03.09
dummy argument 15.03.08
array array 15.03.04
assignment assignment 15.04.07
assignment by name 15.04.08
association parameter association 15.06.08
assumed-size assumed-size aggregate 15.04.11
asynchronous asynchronous procedure 15.05.04
automatic automatic storage allocation 15.04.10

B

block block (in a programming language) .. 15.01.08
branch branch construction 15.06.11
built-in built-in 15.02.04

C

call (procedure) call 15.06.06
comment comment 15.01.12
compound compound statement 15.01.07
conditional conditional construction 15.06.04
connection connection 15.06.16
constant constant 15.03.02
construction branch construction 15.06.11
conditional construction 15.06.04
loop construction 15.06.05
control control flow 15.06.02
conversion conversion 15.06.14
critical critical section 15.05.05

D

data data flow 15.04.06
(data) type 15.04.01
declaration declaration 15.02.01
implicit declaration 15.02.03
default default 15.02.02
description environment description 15.01.13
dynamic dynamic 15.02.10

E

editing editing 15.04.13
effect side effect (of a function procedure) . 15.06.10
encapsulated encapsulated type 15.01.10
entry entry (of a procedure) 15.06.07
environment environment description 15.01.13
exception exception (in a programming language) 15.06.12
execution execution sequence 15.06.01
expression expression 15.05.01
external external 15.02.08

F

flow control flow 15.06.02
data flow 15.04.06
format format 15.04.03
function function (procedure) 15.05.03

G

global global 15.02.07

I

identifier identifier 15.01.02
implicit implicit declaration 15.02.03
indirect indirect referencing 15.04.05
initialize (to) initialize 15.04.09

K

keyword keyword 15.01.03

L

label label (in a programming language) ... 15.05.06
lexical lexical unit 15.01.01
lexical token 15.01.01
lifetime lifetime 15.02.11
literal literal 15.01.05
local local 15.02.06
loop loop construction 15.06.05

M

module module 15.01.09

N

name assignment by name 15.04.08
(name) qualification 15.02.13

O

operator (operator) precedence 15.06.13

P

parameter actual parameter 15.03.09
(formal) parameter 15.03.08
parameter association 15.06.08