

INTERNATIONAL
STANDARD

ISO/IEC
2382-7

NORME
INTERNATIONALE

Third edition
Troisième édition
2000-04-01

**Information technology — Vocabulary —
Part 7:
Computer programming**

**Technologies de l'information —
Vocabulaire —
Partie 7:
Programmation des ordinateurs**



Reference number
Numéro de référence
ISO/IEC 2382-7:2000(E/F)

AVA Documents
© ISO/IEC 2000
Document Sharing Hub

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

PDF – Exonération de responsabilité

Le présent fichier PDF peut contenir des polices de caractères intégrées. Conformément aux conditions de licence d'Adobe, ce fichier peut être imprimé ou visualisé, mais ne doit pas être modifié à moins que l'ordinateur employé à cet effet ne bénéficie d'une licence autorisant l'utilisation de ces polices et que celles-ci y soient installées. Lors du téléchargement de ce fichier, les parties concernées acceptent de fait la responsabilité de ne pas enfreindre les conditions de licence d'Adobe. Le Secrétariat central de l'ISO décline toute responsabilité en la matière.

Adobe est une marque déposée d'Adobe Systems Incorporated.

Les détails relatifs aux produits logiciels utilisés pour la création du présent fichier PDF sont disponibles dans la rubrique General Info du fichier; les paramètres de création PDF ont été optimisés pour l'impression. Toutes les mesures ont été prises pour garantir l'exploitation de ce fichier par les comités membres de l'ISO. Dans le cas peu probable où surviendrait un problème d'utilisation, veuillez en informer le Secrétariat central à l'adresse donnée ci-dessous.

© ISO/IEC 2000

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester. / Droits de reproduction réservés. Sauf prescription différente, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'ISO à l'adresse ci-après ou du comité membre de l'ISO dans le pays du demandeur.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland/Imprimé en Suisse

Contents

	Page
Foreword.....	v
Introduction	vii
Section 1: General	
1.1 Scope.....	1
1.2 Normative references.....	1
1.3 Principles and rules followed	2
1.3.1 Definition of an entry.....	2
1.3.2 Organization of an entry	2
1.3.3 Classification of entries.....	3
1.3.4 Selection of terms and wording of definitions	3
1.3.5 Multiple meanings.....	3
1.3.6 Abbreviations.....	3
1.3.7 Use of parentheses	3
1.3.8 Use of brackets.....	4
1.3.9 Use of terms printed in italic typeface in definitions and the use of an asterisk	4
1.3.10 Spelling.....	4
1.3.11 Organization of the alphabetical index.....	5
Section 2: Terms and definitions	
07 Computer programming.....	6
07.01 Kinds of languages	6
07.02 Methods, techniques, and program structure	12
07.03 Iteration and recursion.....	16
07.04 Program preparation.....	18
07.05 Linking and loading.....	28
07.06 Program execution	31
07.07 Debugging and checking	39
07.08 Microprogramming.....	43
07.09 Instructions and addresses.....	44
07.10 Concurrent processes	51
07.11 Support environments	54
07.12 Goals and principles	55
Figure 1 An example of a structure chart.....	59
Figure 2 An example of a call graph	60
Figure 3 An example of a box diagram	61
Figure 4 An example of a data flow diagram	62
Figure 5 An example of a bubble chart.....	63
Figure 6 An example of an input-process-output chart.....	64
Figure 7 An example of a state transition diagram of a task	65
Alphabetical indexes	
English	66
French.....	73

Sommaire

	Page
Avant-propos	vi
Introduction	viii
Section 1: Généralités	
1.1 Domaine d'application	1
1.2 Références normatives	1
1.3 Principes d'établissement et règles suivies	2
1.3.1 Définition de l'article	2
1.3.2 Constitution d'un article	2
1.3.3 Classification des articles	3
1.3.4 Choix des termes et des définitions	3
1.3.5 Pluralité de sens ou polysémie	3
1.3.6 Abréviations	3
1.3.7 Emploi des parenthèses	3
1.3.8 Emploi des crochets	4
1.3.9 Emploi dans les définitions de termes imprimés en caractères italiques et de l'astérisque	4
1.3.10 Mode d'écriture et orthographe	4
1.3.11 Constitution de l'index alphabétique	5
Section 2: Termes et définitions	
07 Programmation des ordinateurs	6
07.01 Types de langages	6
07.02 Méthodes et techniques de programmation, structure des programmes	12
07.03 Itération et récursion	16
07.04 Élaboration des programmes	18
07.05 Lier et charger	28
07.06 Exécution des programmes	31
07.07 Débogage et vérification	39
07.08 Microprogrammation	43
07.09 Instructions et adresses	44
07.10 Processus concurrents	51
07.11 Environnements de support	54
07.12 Buts et principes	55
Figure 1 Exemple d'organigramme hiérarchique	59
Figure 2 Exemple de graphe d'appel	60
Figure 3 Exemple de diagramme à pavés	61
Figure 4 Exemple de diagramme de flux de données	62
Figure 5 Exemple de diagramme à bulles	63
Figure 6 Exemple de diagramme d'entrée-sortie	64
Figure 7 Exemple de diagramme des transitions d'état d'une tâche	65
Index alphabétiques	
Anglais	66
Français	73

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 and 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 2382 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 2382-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 1, *Vocabulary*.

This third edition cancels and replaces the second edition (ISO/IEC 2382-7:1989), which has been technically revised.

ISO/IEC 2382 will consist of some 37 parts, under the general title *Information technology – Vocabulary*.

Avant-propos

L'ISO (Organisation internationale de normalisation) et la CEI (Commission électrotechnique internationale) forment le système spécialisé de la normalisation mondiale. Les organismes nationaux membres de l'ISO ou de la CEI participent au développement de Normes internationales par l'intermédiaire des comités techniques créés par l'organisation concernée afin de s'occuper des domaines particuliers de l'activité technique. Les comités techniques de l'ISO et de la CEI collaborent dans des domaines d'intérêt commun. D'autres organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'ISO et la CEI participent également aux travaux.

Les Normes internationales sont rédigées conformément aux règles données dans les Directives ISO/CEI, Partie 3.

Dans le domaine des technologies de l'information, l'ISO et la CEI ont créé un comité technique mixte, l'ISO/CEI JTC 1. Les projets de Normes internationales adoptés par le comité technique mixte sont soumis aux organismes nationaux pour vote. Leur publication comme Normes internationales requiert l'approbation de 75 % au moins des organismes nationaux votants.

L'attention est appelée sur le fait que certains des éléments de la présente partie de l'ISO/CEI 2382 peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. L'ISO et la CEI ne sauraient être tenues pour responsables de ne pas avoir identifié de tels droits de propriété et averti de leur existence.

La Norme internationale ISO/CEI 2382-7 a été élaborée par le comité technique mixte ISO/CEI JTC 1, *Technologies de l'information*, sous-comité SC 1, *Vocabulaire*.

Cette troisième édition annule et remplace la deuxième édition (ISO/CEI 2382-7:1989), dont elle constitue une révision technique.

L'ISO/CEI 2382 comprendra environ 37 parties, présentées sous le titre général *Technologies de l'information – Vocabulaire*.

Introduction

Information technology gives rise to numerous international exchanges of both an intellectual and a 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* which was established and published by the International Federation for Information Processing and the International Computation Centre, and in the *American National Dictionary for Information Processing Systems* and its earlier editions published by the American National Standards Institute (formerly known as the American Standards Association). Published and Draft International Standards relating to information technology of other international organizations (such as the International Telecommunication Union and the International Electrotechnical Commission) as well as published and draft national standards have also been considered.

The purpose of ISO/IEC 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 among parts.

Introduction

Les technologies de l'information sont à l'origine de multiples échanges intellectuels et matériels sur le plan international. Ceux-ci souffrent souvent de difficultés provoquées par la diversité des termes utilisés pour exprimer la même notion dans des langues ou 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 l'*American National Dictionary for Information Processing Systems* y compris ses éditions précédentes publiées 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 les technologies de l'information émanant d'autres organisations internationales (telles que l'Union internationale des télécommunications et la Commission électrotechnique internationale), ainsi que les normes nationales publiées ou au stade de projets, ont également été prises en compte.

Le but de l'ISO/CEI 2382 est de procurer des définitions rigoureuses, simples et compréhensibles pour tous les intéressés. La portée de chaque notion a été choisie de façon que sa définition puisse avoir la valeur la plus générale. Cependant, 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 vocabulaire peuvent être à l'origine de quelques répétitions ou contradictions entre des parties qui ne sont pas toutes préparées et publiées simultanément.

Information technology — Vocabulary —

Part 7: Computer programming

Section 1: General

1.1 Scope

This part of ISO/IEC 2382 is intended to facilitate international communication in computer programming. It presents, in two languages, terms and definitions of selected concepts relevant to the field of information technology and identifies relationships among 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.

This part of ISO/IEC 2382 contains general and selected terms concerning computer programming, and specifically preparation, execution, debugging, and verification of programs. ITU Recommendations have been taken into account. Not included are proprietary terms and terms considered as too technical.

1.2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 2382. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 2382 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 1087:1990 ¹⁾, *Terminology — Vocabulary*.

¹⁾ Currently under revision.

²⁾ Actuellement en révision.

Technologies de l'information — Vocabulaire —

Partie 7: Programmation des ordinateurs

Section 1: Généralités

1.1 Domaine d'application

La présente partie de l'ISO/CEI 2382 a pour objet de faciliter les échanges internationaux dans le domaine de la programmation des ordinateurs. 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.

La présente partie de l'ISO/CEI 2382 contient des termes généraux et sélectionnés concernant la programmation des ordinateurs, et notamment la préparation, l'exécution, la mise au point et la vérification des programmes. Les Recommandations de l'UIT ont été prises en compte. Ne sont pas inclus les termes spécifiques d'un constructeur et les termes considérés comme trop techniques.

1.2 Références normatives

Les documents normatifs suivants contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente partie de l'ISO/CEI 2382. Pour les références datées, les amendements ultérieurs ou les révisions de ces publications ne s'appliquent pas. Toutefois, les parties prenantes aux accords fondés sur la présente partie de l'ISO/CEI 2382 sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Pour les références non datées, la dernière édition du document normatif en référence s'applique. Les membres de l'ISO et de la CEI possèdent le registre des Normes internationales en vigueur.

ISO 1087:1990 ²⁾, *Terminologie — Vocabulaire*.

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes.*

ISO 2382-6:1987, *Information processing systems — Vocabulary — Part 6: Preparation and handling of data.*

ISO 2382-10:1979, *Data processing — Vocabulary — Part 10: Operating techniques and facilities.*

ISO/IEC 2382-20:1990, *Information technology — Vocabulary — Part 20: System development.*

ISO/IEC 2382-23:1994, *Information technology — Vocabulary — Part 23: Text processing.*

1.3 Principles and rules followed

1.3.1 Definition of an entry

Section 2 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 1.3.5 and 1.3.8 respectively.

Other terms such as **vocabulary**, **concept**, **term**, and **definition** are used in this part of ISO/IEC 2382 with the meaning defined in ISO 1087.

1.3.2 Organization of an entry

Each entry contains the essential elements defined in 1.3.1 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 part of ISO/IEC 2382 is published);
- b) the term or the generally preferred term in the language. The absence of a generally preferred term for the concept in the language is indicated by a symbol consisting of five dots (.....); 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 3166);
- d) the abbreviation for the term;
- e) permitted synonymous term(s);

ISO 3166-1:1997, *Codes pour la représentation des noms de pays et de leurs subdivisions — Partie 1: Codes pays.*

ISO 2382-6:1987, *Systèmes de traitement de l'information — Vocabulaire — Partie 6: Préparation et manipulation des données.*

ISO 2382-10:1979, *Traitement de l'information — Vocabulaire — Partie 10: Techniques et moyens d'exploitation.*

ISO/CEI 2382-20:1990, *Technologies de l'information — Vocabulaire — Partie 20: Développement de système.*

ISO/CEI 2382-23:1994, *Technologies de l'information — Vocabulaire — Partie 23: Traitement de texte.*

1.3 Principes d'établissement et règles suivies

1.3.1 Définition de l'article

La section 2 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 de la 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 1.3.5 et 1.3.8.

D'autres termes tels que **vocabulaire**, **notion**, **terme**, **définition**, sont employés dans la présente partie de l'ISO/CEI 2382 avec le sens qui leur est donné dans l'ISO 1087.

1.3.2 Constitution d'un article

Chaque article contient des éléments essentiels définis en 1.3.1 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 partie de l'ISO/CEI 2382);
- 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 un cas particulier;
- c) le terme préféré dans un certain pays (identifié selon les règles de l'ISO 3166);
- d) l'abréviation pouvant être employée à la place du terme;
- e) le terme ou les termes admis comme synonymes;

- f) the text of the definition (see 1.3.4);
- 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)";
- i) a picture, a diagram, or a table which could be common to several entries.

1.3.3 Classification of entries

A two-digit serial number is assigned to each part of ISO/IEC 2382, 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 being those of the part of ISO/IEC 2382.

Each entry is assigned a six-digit index number; the first four digits being those of the part of ISO/IEC 2382 and the group.

To show the relationship between versions of ISO/IEC 2382 in various languages, the numbers assigned to parts, groups, and entries are the same for all languages.

1.3.4 Selection of terms and wording of definitions

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

1.3.5 Multiple meanings

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

1.3.6 Abbreviations

As indicated in 1.3.2, abbreviations in current use are given for some terms. Such abbreviations are not used in the texts of the definitions, examples or notes.

1.3.7 Use of parentheses

In some terms, one or more 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 of ISO/IEC 2382, such a term is used only in its complete form.

- f) le texte de la définition (voir 1.3.4);

- 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)»;

- i) une figure, un schéma ou un tableau, pouvant être communs à plusieurs articles.

1.3.3 Classification des articles

Chaque partie de l'ISO/CEI 2382 reçoit un numéro d'ordre à deux chiffres, en commençant par 01 pour la partie «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 la partie de l'ISO/CEI 2382.

Chaque article est repéré par un numéro de référence à six chiffres, les quatre premiers chiffres étant ceux du numéro de partie de l'ISO/CEI 2382 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 l'ISO/CEI 2382.

1.3.4 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.

1.3.5 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.

1.3.6 Abréviations

Comme indiqué en 1.3.2, des abréviations 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.

1.3.7 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 ne crée d'ambiguïté. Un tel terme n'est employé dans le texte d'une autre définition, d'un exemple ou d'une note de l'ISO/CEI 2382, que sous sa forme complète.

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.

1.3.8 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. To clearly identify 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.

1.3.9 Use of terms printed in italic typeface in definitions and the use of an 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 which are defined in this part of ISO/IEC 2382 are listed in the index at the end of the part (see 1.3.11).

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

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

1.3.10 Spelling

In the English language version of this part of ISO/IEC 2382, terms, definitions, examples, and notes are given in the spelling preferred in the USA. Other correct spellings may be used without violating this part of ISO/IEC 2382.

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.

1.3.8 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 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.

1.3.9 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 des termes imprimés en caractères italiques qui sont définis dans la présente partie de l'ISO/CEI 2382 est fournie dans l'index à la fin de la partie (voir 1.3.11).

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

1.3.10 Mode d'écriture et orthographe

Dans la version anglaise de la présente partie de l'ISO/CEI 2382, 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 partie de l'ISO/CEI 2382.

1.3.11 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.

1.3.11 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 clés.

... ..

Section 2: Terms and definitions

07 Computer programming

07.01 Kinds of languages

07.01.01

metalanguage

A *language* used to specify some or all aspects of another language and possibly itself.

Example: Backus-Naur form.

07.01.02

algorithmic language

An *artificial language* for expressing *algorithms*.

07.01.03 (01.05.10)

programming language

An *artificial language* for expressing *programs*.

07.01.04

machine language

An *artificial language* composed only of the *machine instructions* of a specific *computer* or class of computers.

07.01.05

machine-oriented language computer-oriented language

A *programming language*, the *simple statements* of which have the same or similar structure as the *machine instructions* of a specific *computer* or class of computers.

07.01.06

assembly language

A *machine-oriented language* that provides symbolic naming of *operations* and locations and other features such as *macroinstructions*.

07.01.07

first-generation language

1GL (abbreviation)

A *programming language* closely resembling *assembly language* and very dependent on the *machine language* of a *computer*.

Section 2: Termes et définitions

07 Programmation des ordinateurs

07.01 Types de langages

07.01.01

métalanguage

Langage utilisé pour spécifier tout ou partie des caractéristiques d'un autre langage, et éventuellement des *siennes*.

Exemple: Notation de Backus-Naur.

07.01.02

langage algorithmique

Langage artificiel permettant d'exprimer des *algorithmes*.

07.01.03 (01.05.10)

langage de programmation

Langage artificiel permettant d'exprimer des *programmes*.

07.01.04

langage machine

langage d'ordinateur

Langage artificiel qui contient uniquement des *instructions machine* d'un *ordinateur* particulier ou d'une classe d'ordinateurs.

07.01.05

langage orienté machine

Langage de programmation dont les *instructions simples* ont une structure identique ou semblable à celle des *instructions machine* d'un *ordinateur* particulier ou d'une classe d'ordinateurs.

07.01.06

langage d'assemblage

Langage orienté machine qui utilise des noms symboliques pour désigner les *opérations* et les emplacements, ainsi que d'autres caractéristiques comme les *macro-instructions*.

07.01.07

langage de première génération

Langage de programmation très similaire au *langage d'assemblage* et dépendant du *langage machine* d'un *ordinateur*.

07.01.08**high-level language
high-order language**

A *programming language* that is primarily designed for, and syntactically oriented to, particular classes of problems and that is essentially independent of the structure of a specific *computer* or class of computers.

Exemples: Ada, COBOL, Fortran, Pascal.

07.01.09**symbolic language**

A *programming language* that names *operations*, * *addresses*, * *operands*, and *results* in symbolic form.

Exemples: *Assembly language*, * *high-level language*.

07.01.10**second-generation language**

2GL (abbreviation)

A *programming language* that extends a *first-generation language* to include higher-level *language constructs* such as *macroinstructions*.

07.01.11**third-generation language**

3GL (abbreviation)

A *high-level language* that has a high ratio of *machine instructions* to each of its *simple statements* and that raises the *programmer's* level of abstraction to focusing attention on the problem to be solved instead of on an intimate knowledge about how a particular *computer* works.

Exemples: Ada, BASIC, Fortran, Modula-2, Pascal.

07.01.12**fourth-generation language**

4GL (abbreviation)

A *high-level language* that allows a user, not necessarily a *programmer*, to write *statements* in near-*natural language*, that has a ratio of *machine instructions* to *simple statements* much higher than that of a *third-generation language*, and that elevates the level of abstraction at which the user may work beyond that of previous generations of *programming languages*.

Exemples:

1. In a fourth-generation language, *sorting* a customer list could be expressed as: "Sort customer_list on customer_name in ascending order". The user need not know any sorting *algorithm*.
2. dBASE is a fourth-generation language.

07.01.08**langage évolué
langage de haut niveau**

Langage de programmation qui est notamment conçu pour une classe particulière de problèmes et est syntaxiquement adapté à cette classe, et qui est essentiellement indépendant de la structure d'un *ordinateur* particulier ou d'une classe d'ordinateurs.

Exemples: Ada, COBOL, Fortran, Pascal.

07.01.09**langage symbolique**

Langage de programmation qui utilise une forme symbolique pour désigner les *opérations*, les *adresses*, les *opérandes* et les *résultats*.

Exemples: Les *langages d'assemblage*, les *langages évolués*.

07.01.10**langage de deuxième génération**

Langage de programmation qui prolonge un *langage de première génération* pour inclure des *éléments de langage* de niveau supérieur tels que des *macro-instructions*.

07.01.11**langage de troisième génération**

Langage de haut niveau pouvant générer de nombreuses *instructions machine* pour chacune de ses *instructions simples* et qui augmente le niveau d'abstraction du *programmeur* en concentrant son attention sur le problème à résoudre et non pas sur une connaissance approfondie de la manière dont fonctionne un *ordinateur* particulier.

Exemples: Ada, BASIC, Fortran, Modula-2, Pascal.

07.01.12**langage de quatrième génération**

Langage évolué qui permet à un utilisateur, non nécessairement *programmeur*, d'écrire des *instructions* dans un *langage* presque *naturel*; ce langage peut générer, pour chacune de ses *instructions simples*, des *instructions machine* beaucoup plus nombreuses que dans le cas des *langages de troisième génération*, et il élève le niveau d'abstraction auquel l'utilisateur peut travailler, bien au-delà des générations précédentes de *langages de programmation*.

Exemples:

1. Dans un langage de quatrième génération, l'ordre de *tri* d'une liste de clients pourrait être exprimé comme suit: «trier la liste_de_clients en fonction du nom_de_client en ordre ascendant». L'utilisateur n'a pas besoin de connaître d'*algorithme* de tri.
2. dBASE est un langage de quatrième génération.

07.01.13
extensible language

A *programming language* that can be altered or can alter itself to provide a *programmer* with additional user-specified capabilities.

Exemples: Ada, C++, FORTH, LISP, LOGO, Prolog, Smalltalk.

07.01.14
algebraic language

A *programming language* that permits the construction of *statements* resembling algebraic expressions.

Exemples: Ada, Fortran, Pascal.

07.01.15
problem-oriented language
application-oriented language

A *programming language* that reflects the concepts of a particular application area.

Exemples: SQL for *database* applications, COBOL for business applications.

07.01.16
object-oriented language

A *programming language* that supports *object-oriented* concepts.

Exemples: Eiffel, Smalltalk.

07.01.17
imperative language

A *programming language* that achieves its primary effect by changing the state of *variables* by assignment.

07.01.18
procedural language
procedure-oriented language

A *programming language* that provides the means to state what is to be achieved by the actions of a *data processing system* by giving specific *statements* or *instructions* to be *executed* in a specific sequence.

Exemples: Ada, BASIC, COBOL, Fortran, and Pascal.

07.01.19
nonprocedural language

A *programming language* that provides the means to state what is to be achieved by the actions of a *data processing system* without giving specific *statements* or *instructions* to be *executed* in a specific sequence.

07.01.13
langage extensible

Langage de programmation qui peut être modifié, ou peut se modifier lui-même, pour fournir au *programmeur* des possibilités supplémentaires spécifiées par l'utilisateur.

Exemples: Ada, C++, FORTH, LISP, LOGO, Prolog, Smalltalk.

07.01.14
langage algébrique

Langage de programmation qui permet de construire des *instructions* ressemblant à des expressions algébriques.

Exemples: Ada, Fortran, Pascal.

07.01.15
langage d'application
langage orienté problème

Langage de programmation qui convient pour représenter un domaine d'application particulier.

Exemples: SQL pour les applications utilisant des *bases de données*, COBOL pour les applications de gestion.

07.01.16
langage orienté objets

Langage de programmation adapté aux concepts *orientés objets*.

Exemples: Eiffel, Smalltalk.

07.01.17
langage impératif

Langage de programmation dont le principal mode d'action est de changer l'état des *variables* par affectation.

07.01.18
langage procédural

Langage de programmation qui permet de spécifier ce qu'on attend d'un *système informatique* en indiquant des *instructions* particulières, à *exécuter* dans un ordre particulier.

Exemples: Ada, BASIC, COBOL, Fortran, Pascal.

07.01.19
langage non procédural

Langage de programmation qui permet de spécifier ce qu'on attend d'un *système informatique* sans indiquer d'*instructions* particulières à *exécuter* dans un ordre particulier.

07.01.20**functional language**

A *programming language* that provides the means to state what is to be achieved by the actions of a *data processing system* exclusively through the use of *function calls*.

Examples: FORTH, LISP, ML, Miranda, Postscript.

07.01.21**structured programming language**

A *programming language* that provides *language constructs* for *structured programming* (2).

07.01.22**block-structured language**

A *programming language* that supports the use of *block statements*.

Examples: Ada, ALGOL, C, Pascal, PL/I.

07.01.23**general-purpose language**

A *high-level language* suitable for use in a wide variety of applications.

07.01.24**special-purpose language**

A *programming language* that focuses its capabilities on a particular kind of application.

Examples: A form-filling *language*; Postscript.

07.01.25**interactive language****conversational language**

A *programming language* that supports communication between a user and a *data processing system* in a *conversational mode*.

07.01.26**list processing language**

A *programming language* designed to manipulate *data* expressed in the form of *lists* or *character strings*.

Example: LISP.

07.01.20**langage fonctionnel**

Langage de programmation qui permet de spécifier ce qu'on attend d'un *système informatique* en employant uniquement des *appels de fonctions*.

Exemples: FORTH, LISP, ML, Miranda, Postscript.

07.01.21**langage de programmation structurée**

Langage de programmation qui fournit des *éléments de langage* pour la *programmation structurée* (2).

07.01.22**langage à structure de blocs**

Langage de programmation qui permet l'utilisation d'*instructions blocs*.

Exemples: Ada, ALGOL, C, Pascal, PL/I.

07.01.23**langage (de programmation) universel**

Langage évolué convenant à une grande variété d'applications.

07.01.24**langage spécialisé**

Langage de programmation adapté à un type particulier de problèmes.

Exemples: Un *langage* de traitement de formulaires; Postscript.

07.01.25**langage interactif****langage conversationnel**

Langage de programmation qui permet à un utilisateur de communiquer avec un *système informatique*, en *mode dialogué*.

07.01.26**langage de traitement de listes**

Langage de programmation destiné à manipuler des *données* exprimées sous la forme de *listes* ou de *chaînes de caractères*.

Exemple: LISP.

07.01.27

expression language

A *programming language* in which assignments can be made in the context of an *expression*.

Example: C.

NOTE - The expression "if (x = y < 0) ..." is legal in C, but would be illegal in Ada.

07.01.28

text-formatting language

A *problem-oriented language* designed to indicate the manner in which *text* should be formatted.

Examples: HTML, nroff.

07.01.29

markup language

A *text-formatting language* designed to transform raw *text* into structured *documents*, by inserting procedural and descriptive markup into the raw text.

NOTE - This entry is a modified version of the entry 23.06.33 in ISO/IEC 2382-23:1994.

07.01.30

page description language

PDL (abbreviation)

A *text-formatting language* used to specify the printed or *display image* of a *document*, page by page.

Examples: HPGL, Postscript.

07.01.31

authoring language

A *problem-oriented language* designed to develop courseware for *computer-aided* instruction.

07.01.32

macrolanguage (1)

A *programming language* designed to define *macro-definitions* and *macroinstructions*.

07.01.33

macrolanguage (2)

A *programming language* that includes *macrodefinitions* and *macroinstructions*.

07.01.27

langage d'expression

Langage de programmation dans lequel des affectations de valeurs peuvent être faites dans le cadre d'une *expression*.

Exemple: C.

NOTE - L'expression «if (x= y < 0) ...» est autorisée en C, mais ne l'est pas en Ada.

07.01.28

langage de formatage de texte

Langage d'application conçu pour indiquer la façon dont du *texte* doit être formaté.

Exemples: HTML, nroff.

07.01.29

langage de balisage

Langage de formatage de texte destiné à transformer un *texte* brut en *document* structuré, en insérant, dans le *texte* brut, des marques procédurales et descriptives.

NOTE - Cet article est une version modifiée de l'article 23.06.33 de l'ISO/CEI 2382-23:1994.

07.01.30

langage de description de page

LDP (abréviation)

Langage de formatage de texte utilisé pour spécifier l'*image* imprimée ou *affichée* d'un *document*, page par page.

Exemples: HPGL, Postscript.

07.01.31

langage auteur

Langage d'application conçu pour écrire un cours dans le cadre d'un enseignement *assisté par ordinateur*.

07.01.32

macrolangage (1)

Langage de programmation conçu pour définir des *macrodéfinitions* et des *macro-instructions*.

07.01.33

macrolangage (2)

Langage de programmation qui inclut des *macrodéfinitions* et des *macro-instructions*.

07.01.34**specification language**

A *problem-oriented language*, often a *computer-processible* combination of *natural language* and *artificial language*, used for expressing the *requirements*, design, behavior, or other characteristics of a system or a component and that provides special *language constructs* and, sometimes, *verification * protocols* used to develop, analyze, and document the specified entities.

07.01.35**requirement specification language**

A *specification language* with special *language constructs* and, sometimes, *verification * protocols*, used to develop, analyze, and document *hardware* requirements or *software* requirements, or both.

07.01.36**design language**

A *specification language* with special *language constructs* and, sometimes, *verification * protocols*, used to develop, analyze, and document the design of *hardware* or *software*.

07.01.37**hardware design language****HDL** (abbreviation)

A *design language* with special *language constructs* and, sometimes, *verification * protocols*, used to develop, analyze, and document a *hardware* design.

07.01.38**program design language**

A *design language* with special *language constructs* and *verification * protocols*, used to develop, analyze, and document the design of a *program*.

07.01.39**pseudocode**

A combination of *language constructs* from a *programming language* with those of *natural language* that is not necessarily *computer-processible*, but intended to make the design of a *program* manifest to human readers.

Example:

```
IF the data arrive faster than expected,
    THEN reject every third input.
ELSE process all data received.
ENDIF.
```

07.01.34**langage de spécification**

Langage d'application, résultant souvent d'une combinaison, pouvant être traitée par *ordinateur*, de *langage naturel* et de *langage artificiel*; ce langage permet d'exprimer les *besoins*, la conception, le comportement ou d'autres caractéristiques d'un système ou d'un composant, et fournit des *éléments de langage* particuliers, et parfois des *protocoles de vérification*, pour développer, analyser et documenter les entités spécifiées.

07.01.35**langage de spécification de besoins**

Langage de spécification contenant des *éléments de langage* particuliers, et parfois des *protocoles de vérification*, pour développer, analyser et documenter des besoins d'un *matériel* ou d'un *logiciel*, ou des deux.

07.01.36**langage de conception**

Langage de spécification contenant des *éléments de langage* particuliers, et parfois des *protocoles de vérification*, pour développer, analyser et documenter la conception d'un *matériel* ou d'un *logiciel*.

07.01.37**langage de conception de circuits****langage de conception matérielle**

Langage de conception contenant des *éléments de langage* particuliers, et parfois des *protocoles de vérification*, pour développer, analyser et documenter la conception d'un *matériel*.

07.01.38**langage de conception de programmes****langage de conception logicielle**

Langage de spécification contenant des *éléments de langage* particuliers, et parfois des *protocoles de vérification*, pour développer, analyser et documenter la conception d'un *programme*.

07.01.39**pseudocode**

Combinaison d'éléments d'un *langage de programmation* et d'un *langage naturel*, qui n'est pas nécessairement traitable par *ordinateur*, mais a pour but d'expliquer à un lecteur humain la conception d'un *programme*.

Example:

```
SI les données arrivent plus vite que prévu,
    ALORS rejeter un article sur trois.
SINON traiter toutes les données reçues.
FINDESI.
```

07.01.40

compiler specification language

A *specification language* used to develop *compilers*.

07.01.41

test language

A *problem-oriented language* that provides the means for testing components of *hardware* or *software*.

Examples: ATLAS, ATOLL, DETOL, DMAD.

07.02 Methods, techniques, and program structure

07.02.01

structured programming (1)

A method for constructing *programs* using only hierarchically arranged constructs each having a single *entry point* and a single *exit point*.

NOTE - Three kinds of *control flow* are used in structured programming: *sequential*, conditional, and iterative.

07.02.02

structured programming (2)

Any *software* development technique that includes *structured design* and that results in the development of *structured programs*.

07.02.03

structured program

A *program* constructed according to the principles of *structured programming* (1).

07.02.04

structured design

Any disciplined approach to *software* design that adheres to specified rules based on principles such as *modularity*, *top-down* design, and *stepwise refinement* of *data*, of system structures, and of processing steps.

07.02.05

stepwise refinement

A *software* development technique in which processing steps and *data* are defined broadly at first, then further defined with increasing detail.

07.01.40

langage de spécification de compilateur

Langage de spécification utilisé pour développer des *compilateurs*.

07.01.41

langage de tests

Langage d'application qui fournit des outils pour tester les composants d'un *matériel* ou d'un *logiciel*.

Exemples: ATLAS, ATOLL, DETOL, DMAD.

07.02 Méthodes et techniques de programmation, structure des programmes

07.02.01

programmation structurée (1)

Méthode de construction de *programmes* n'utilisant que des éléments disposés hiérarchiquement et ayant chacun un *point d'entrée* unique et un *point de sortie* unique.

NOTE - Trois types de *flux de commande* sont utilisés en programmation structurée: *séquentiel*, conditionnel et itératif.

07.02.02

programmation structurée (2)

Toute technique de développement de *logiciel* qui inclut la *conception structurée* et produit des *programmes structurés*.

07.02.03

programme structuré

Programme construit selon les principes de la *programmation structurée* (1).

07.02.04

conception structurée

Toute méthode disciplinée de conception de *logiciel*, conforme à des règles spécifiées fondées sur des principes tels que la *modularité*, la conception *descendante* et l'*affinement progressif* des *données*, des structures du système et des étapes de traitement.

07.02.05

affinement progressif

Technique de développement de *logiciel* dans laquelle les étapes de traitement et les *données* sont d'abord définies approximativement, puis sont redéfinies avec plus de détails.

07.02.06**to nest**

To incorporate one or more structures of one kind into a structure of the same kind.

Examples: To nest one *loop* (the nested or inner loop) within another loop (the nesting or outer loop); to nest one *subprogram* within another subprogram.

07.02.07**functional programming**

A method for structuring *programs* mainly as sequences of possibly *nested* * *function calls*.

07.02.08**modular programming**

A *software* development technique in which software is developed as a collection of *modules*.

07.02.09**logic programming**

A method for structuring *programs* as sets of logical rules with predefined *algorithms* for the processing of *input data* to a program according to the rules of that program.

07.02.10**jump**

A departure from the sequential *execution* of *instructions* or *statements*.

NOTE - A jump is caused by an appropriate instruction or statement, in contrast to *asynchronous* interruption or interruption due to an *exception* where control is transferred to an *exception handler*.

07.02.11**to jump**

To depart from the implicit or declared order in which *instructions* or *statements* are being *executed*.

07.02.12**indicator**

A device or a *variable* that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition.

Examples: A *flag*, a *semaphore*.

07.02.13**flag**

A *variable* indicating the status of a certain condition.

07.02.06**emboîter**

Incorporer une ou plusieurs structures d'un certain type à l'intérieur d'une structure du même type.

Exemples: Emboîter une *boucle* (boucle emboîtée ou interne) à l'intérieur d'une autre boucle (boucle emboîtante ou externe); emboîter un *sous-programme* à l'intérieur d'un autre sous-programme.

07.02.07**programmation fonctionnelle**

Méthode utilisée pour structurer les *programmes* en les écrivant, pour la plus grande part, sous la forme de suites d'*appels de fonctions*, éventuellement *emboîtés*.

07.02.08**programmation modulaire**

Technique de développement de *logiciel* dans laquelle le logiciel est écrit sous la forme d'une collection de *modules*.

07.02.09**programmation logique**

Méthode utilisée pour structurer les *programmes* sous la forme d'ensembles de règles logiques munis d'*algorithmes* prédéfinis permettant de traiter, selon ces règles, les *données d'entrée* d'un programme.

07.02.10**saut**

Déviation par rapport à l'*exécution* séquentielle des *instructions*.

NOTE - Un saut est provoqué par une instruction appropriée, par opposition à l'interruption *asynchrone* ou à une interruption due à une *exception*, dans laquelle la commande est transférée à un *gestionnaire d'exception*.

07.02.11**sauter**

S'écarter de l'ordre implicite ou explicite selon lequel les *instructions* doivent être *exécutées*.

07.02.12**indicateur**

Dispositif ou *variable* qui peut être mis dans un état prescrit basé sur les résultats d'un processus ou l'apparition d'une condition spécifiée.

Exemple: Un *drapeau*, un *sémaphore*.

07.02.13**drapeau****fanion**

Variable indiquant l'état d'une certaine condition.

07.02.14

switch

A choice of one *jump* from a selection of jumps, controlled by a *flag*.

07.02.15

working space

work space

working area

work area

A portion of a *storage device* used by a *program* to hold *data* temporarily.

07.02.16

mutual exclusion

A principle requiring that, at a given time, only one *asynchronous procedure* may access the same *shared variable* or *execute* members of a group of *critical sections*.

07.02.17

synchronization

The action of maintaining common timing and coordination of the *execution* of two or more *asynchronous procedures*.

07.02.18

hashing

hash addressing

A method of transforming a *search key* into an *address* for the purpose of *storing* and retrieving *data*.

NOTE - The method is often designed to minimize the *search time*.

07.02.19

hash function (in hashing)

A function used to determine the position of a given item in a set of items.

NOTE - The hash function operates on a selected *field*, the *key*, in each item and is used to map the set of keys to a usually much smaller set of storage positions; therefore this mapping is usually a many-to-one mapping.

07.02.20

hash value

The number generated by a *hash function* to indicate the position of a given item in a *storage device*.

07.02.21

collision (in hashing)

hash clash

The occurrence of the same *hash value* for two or more different keys.

07.02.14

aiguillage

Choix, commandé par un *drapeau*, d'un *saut* parmi plusieurs sauts possibles.

07.02.15

zone de travail

espace de travail

Partie de *mémoire* utilisée par un *programme* pour y ranger temporairement des *données*.

07.02.16

exclusion mutuelle

Principe selon lequel une seule *procédure asynchrone* à la fois accède à une *variable partagée* ou *exécute* des membres d'un groupe de *sections critiques*.

07.02.17

synchronisation

Action de maintenir un instant commun et une coordination dans l'*exécution* de plusieurs *procédures asynchrones*.

07.02.18

hachage

adressage calculé

Méthode utilisée pour transformer une *clé de recherche* en une *adresse*, en vue de ranger ou de retrouver des *données*.

NOTE - Cette méthode vise à minimiser le *temps d'exploration*.

07.02.19

fonction de hachage (en hachage)

Fonction utilisée pour déterminer la position d'un élément déterminé dans un ensemble d'éléments.

NOTE - La fonction de hachage utilise une zone déterminée de chaque élément, la *clé*, et elle permet de faire correspondre l'ensemble des clés à un ensemble, généralement beaucoup plus petit, de positions de mémoire; cette correspondance est donc généralement du type «plusieurs à un».

07.02.20

valeur de hachage

valeur d'adressage calculé

Nombre résultant d'une *fonction de hachage* et indiquant la position en *mémoire* d'un élément déterminé.

07.02.21

collision (en hachage)

Apparition de la même *valeur de hachage* pour plusieurs *clés* différentes.

07.02.22**collision resolution** (in hashing)

The process of applying further calculations or other means to resolve a *collision*.

07.02.23**shared variable**

A *variable* that can be accessed by two or more *asynchronous procedures* or *concurrently * executed * programs*.

07.02.24**to bind**

To relate an *identifier* to another object in a *program*.

Examples: To relate an identifier to a value, an *address* or another identifier, or to associate *formal parameters* and *actual parameters*.

07.02.25**binding**

The process of relating an *identifier* to another object in a *program*.

07.02.26**binding time**

The instant at which *binding* takes place.

NOTE - *Programming languages* designed for both efficient *execution* and flexibility, such as Ada, PL/I and C++, provide for multiple options that allow choices of binding times.

07.02.27**static binding**

Binding performed prior to the *execution* of a *program* and not subject to change during execution.

07.02.28**dynamic binding**

Binding performed during the *execution* of a *program*.

07.02.29**early binding**

A characteristic of *programming languages* that perform most *bindings* during *translating*, usually to achieve *execution* efficiency.

Examples: COBOL, Fortran, Pascal.

07.02.22**résolution de collision** (en hachage)

Processus consistant à appliquer d'autres calculs ou d'autres méthodes pour résoudre une *collision*.

07.02.23**variable partagée**

Variable à laquelle peuvent accéder plusieurs *procédures asynchrones* ou bien plusieurs *programmes * exécutés de façon concurrente*.

07.02.24**relier**

Rattacher un *identificateur* à un autre objet de *programme*.

Exemple: Relier un identificateur à une valeur, à une *adresse* ou à un autre identificateur ou bien relier des *paramètres formels* à des *paramètres effectifs*.

07.02.25**liaison**

Processus consistant à rattacher un *identificateur* à un autre objet de *programme*.

07.02.26**instant de liaison**

Instant auquel une *liaison* prend place.

NOTE - Les *langages de programmation* conçus pour être à la fois efficaces à l'*exécution* et souples d'emploi, tels que Ada, PL/I et C++, contiennent plusieurs options qui permettent de choisir l'instant de liaison.

07.02.27**liaison statique**

Liaison effectuée avant l'*exécution* d'un *programme*, et non susceptible d'être modifiée durant l'exécution.

07.02.28**liaison dynamique**

Liaison effectuée durant l'*exécution* d'un *programme*.

07.02.29**liaison précoce**

Caractéristique des *langages de programmation* qui effectuent la plupart des *liaisons* durant la phase de *traduction*, habituellement pour obtenir une meilleure efficacité d'*exécution*.

Exemples: COBOL, Fortran, Pascal.

07.02.30

late binding

A characteristic of *programming languages* that perform most *bindings* during *execution*, usually to achieve flexibility.

Examples: dBASE, Smalltalk.

07.02.31

heap

A part of *internal storage* used for dynamically building or deleting *data objects*, where the order of using the data objects is undefined.

07.02.32

data flow

The movement of *data* through the active parts of a *data processing system* in the course of the performance of specific work.

07.03 Iteration and recursion

07.03.01

iteration

The process of performing a sequence of steps repeatedly.

07.03.02

iteration step

A single *execution* of the sequence of steps of an *iteration*.

07.03.03

loop

A *sequence* of *statements* or of *instructions* that may be *executed* iteratively while a certain condition prevails.

NOTE - In some implementations, no test is made to discover whether the condition prevails until the loop has been executed once.

07.03.04

infinite loop

closed loop

A *loop* whose *execution* can be terminated only by external intervention.

07.03.05

loop assertion

A *logical expression* specifying one or more conditions that must be met each time a particular part of a *loop* is *executed*.

07.02.30

liaison tardive

Caractéristique des *langages de programmation* qui effectuent la plupart des *liaisons* durant l'*exécution*, habituellement pour obtenir une meilleure souplesse.

Exemples: dBase, Smalltalk.

07.02.31

tas

Partie de la *mémoire interne* utilisée pour construire ou supprimer dynamiquement des *objets de données* et dans laquelle l'ordre d'utilisation des données n'est pas défini.

07.02.32

flux de données

Mouvement des *données* à travers les parties actives d'un *système informatique* pendant l'accomplissement d'un travail particulier.

07.03 Itération et récursion

07.03.01

itération

répétition

Action de répéter une succession d'étapes.

07.03.02

pas d'itération

Action d'*exécuter* une seule fois la succession d'étapes d'une *itération*.

07.03.03

boucle

Suite d'*instructions* qui peut être *exécutée* de façon répétée tant qu'une certaine condition est vérifiée.

NOTE - Dans certaines réalisations, on exécute la boucle une fois avant de tester si la condition est vérifiée.

07.03.04

boucle infinie

boucle fermée

Boucle dont l'*exécution* ne peut être arrêtée que par une intervention extérieure.

07.03.05

condition de boucle

Expression logique spécifiant une ou plusieurs conditions qui doivent être satisfaites chaque fois qu'on veut *exécuter* une partie déterminée d'une *boucle*.

07.03.06**loop body**

The part of a *loop* that accomplishes the loop's primary purpose.

07.03.07**loop control**

A *language construct* that includes a test to determine whether an *iteration* of a *loop* is to be *executed*.

07.03.08**loop-control variable****loop parameter**

A *data object* used to determine whether to *exit* from a *loop*.

07.03.09**iteration scheme**

The method used in the *loop control* to determine whether to *exit* from a *loop*.

Example: A "do ... while" clause.

07.03.10**fixed-count iteration**

An *iteration scheme* that terminates *execution* of a *loop* after a specific number of *iterations* rather than until a specific condition occurs.

07.03.11**termination test**

In a *loop control*, the test in which a TRUE condition indicates that the *iteration* should halt.

Example: In Pascal, the *loop-control variable* for a termination test is preceded by an "until" clause.

07.03.12**continuation test**

In a *loop control*, the test in which a TRUE condition indicates that the *iteration* should continue; the FALSE condition indicates that the iteration should terminate.

Example: In Pascal, the *loop-control variable* for a continuation test is in the "while" clause.

07.03.13**pretest loop**

A *loop control* that performs a test before entry into the *loop body*.

Example: A "for" loop in Ada.

NOTE - Usually, a pretest loop is preferred, because a *posttest loop* permits one *execution* of the *loop* before the test is first performed.

07.03.06**corps de la boucle**

Partie d'une *boucle* qui accomplit la tâche principale de la boucle.

07.03.07**contrôle de boucle**

Élément de langage qui comprend un test pour déterminer s'il faut, ou non, effectuer une *itération* d'une *boucle*.

07.03.08**variable de boucle**

Objet de donnée utilisé pour déterminer s'il faut, ou non, sortir d'une *boucle*.

07.03.09**type de boucle**

Méthode utilisée dans le *contrôle de boucle* pour déterminer s'il faut, ou non, sortir d'une *boucle*.

Exemple: Une clause «do ... while».

07.03.10**boucle à compteur**

Type de boucle qui termine l'*exécution* d'une *boucle* après un nombre spécifié d'*itérations* plutôt que par l'apparition d'un événement particulier.

07.03.11**test de fin**

Dans un *contrôle de boucle*, test selon lequel un état VRAI indique que l'*itération* doit s'arrêter.

Exemple: En Pascal, la *variable de boucle* d'un test de fin est précédée d'une clause «until».

07.03.12**test de continuation**

Dans un *contrôle de boucle*, test selon lequel un état VRAI indique que l'*itération* doit continuer; un état FAUX indique que l'*itération* doit s'arrêter.

Exemple: En Pascal, la *variable de boucle* d'un test de continuation est précédée d'une clause «while».

07.03.13**boucle à test préliminaire**

Contrôle de boucle qui effectue un test avant l'entrée dans le *corps de la boucle*.

Exemple: Une boucle «for» en Ada.

NOTE - Habituellement, on préfère une boucle à test préliminaire, parce qu'une *boucle à test final* provoque une *exécution* de la *boucle* avant même d'effectuer une fois le test.

07.03.14
posttest loop

A *loop control* that performs the test after the *loop body*.

Example: In Pascal the "repeat ... until" construct.

07.03.15
in-test loop

A *loop control* that performs the test somewhere in the middle of the *loop body*.

Example: The *exit statement* in Ada.

07.03.16
recursion

A *process* in which a *subprogram* either contains a *subprogram call* on itself, or *calls* another subprogram that calls the original subprogram or that initiates a further chain of subprogram calls that eventually leads back to a subprogram call of the original subprogram.

07.03.17
directly recursive

Pertaining to a *subprogram* that contains a *call* on itself.

07.03.18
indirectly recursive

Pertaining to a *subprogram* that *calls* another subprogram which calls the original subprogram or that initiates a further chain of *subprogram calls* that eventually leads back to a subprogram call of the original subprogram.

07.03.19
mutual recursion

A situation in which two *subprograms* * *call* each other.

07.03.20
reentrant

Pertaining to a *program* or a part of a program in its executable version, that may be entered repeatedly, or may be entered before previous *executions* have been completed, and each execution of such a program is independent of all other executions.

07.04 Program preparation

07.04.01
programmer

A person who designs, writes, or tests *programs*.

07.03.14
boucle à test final

Contrôle de boucle qui effectue le test après le *corps de la boucle*.

Exemple: En Pascal, la construction «repeat ... until».

07.03.15
boucle à test interne

Contrôle de boucle qui effectue le test quelque part dans le milieu du *corps de la boucle*.

Exemple: L'instruction «exit» en Ada.

07.03.16
réursion

Processus dans lequel un *sous-programme* soit contient un *appel de sous-programme* sur lui-même, soit *appelle* un autre sous-programme qui à son tour appelle le sous-programme initial ou bien déclenche une autre chaîne d'appels de sous-programmes qui finalement ramène à un appel de sous-programme sur le sous-programme initial.

07.03.17
récuratif direct

Qualifie un *sous-programme* qui contient un *appel* sur lui-même.

07.03.18
récuratif indirect

Qualifie un *sous-programme* qui *appelle* un autre sous-programme qui à son tour appelle le sous-programme initial ou bien déclenche une autre chaîne d'appels de *sous-programmes* qui finalement ramène à un appel de sous-programme sur le sous-programme initial.

07.03.19
réursion mutuelle

Situation dans laquelle deux *sous-programmes* s'appellent l'un l'autre.

07.03.20
réentrant

Qualifie un *programme* ou une partie de programme sous forme exécutable, que l'on peut exécuter de façon répétée, dont on peut déclencher une nouvelle *exécution* avant l'achèvement d'une exécution antérieure, et dont chaque exécution est indépendante de toutes les autres exécutions.

07.04 Élaboration des programmes

07.04.01
programmeur

Personne qui conçoit, écrit ou teste des *programmes*.

**07.04.02
environment**

A collection of *hardware* and *software tools* to support one or more phases of software development.

**07.04.03
programming environment
programming support environment**

A collection of *hardware* and *software tools* to support the preparation of *programs*.

**07.04.04
integrated programming environment
IPE (abbreviation)**

An integrated collection of *hardware* and *software tools* under a common user interface, frequently graphical, to support the development of *programs*.

**07.04.05
to translate**

To transform, without any modification of the original meaning, all or part of a *program* expressed in one *programming language* into another programming language.

NOTE - This entry is a modified version of the entry 06.03.05 in ISO 2382-6:1987.

**07.04.06
translation**

The process or the result of *translating*.

**07.04.07
translator
translation program**

One or more *programs* that can *translate*.

**07.04.08
to assemble**

To *translate* from an *assembly language* into an *object language*.

**07.04.09
assembler**

A *translator* that can *assemble*.

**07.04.10
absolute assembler**

An *assembler* that produces *absolute code*.

**07.04.02
environnement**

Collection d'outils matériels et logiciels permettant de faciliter une ou plusieurs phases du développement du *logiciel*.

**07.04.03
environnement de programmation
contexte de programmation**

Collection de *matériels* et d'*outils logiciels* permettant de préparer des *programmes*.

**07.04.04
environnement intégré de programmation**

Collection intégrée de *matériels* et d'*outils logiciels* placés sous une interface d'utilisateur commune, généralement graphique, pour faciliter le développement de *programmes*.

**07.04.05
traduire**

Transformer tout ou partie d'un *programme* écrit dans un *langage de programmation*, pour l'exprimer dans un autre langage de programmation, sans modifier la signification originale.

NOTE - Cet article est une version modifiée de l'article 06.03.05 de la norme ISO 2382-6:1987.

**07.04.06
traduction**

Action de *traduire* ou résultat de cette action.

**07.04.07
traducteur
programme de traduction**

Programme conçu pour *traduire*.

**07.04.08
assembler**

Traduire d'un *langage d'assemblage* vers un *langage objet*.

**07.04.09
assembleur**

Traducteur conçu pour *assembler*.

**07.04.10
assembleur absolu**

Assembleur qui produit du *code absolu*.