

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)

© ISO/IEC 2000

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.

IECNORM.COM : Click to view the full PDF of ISO/IEC 2382-7:2000

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 *opérands* 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.

Exemples: 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.

Exemples: 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

récurif direct

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

07.03.18

récurif 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*.

07.04.11

code (in computer programming)

A piece of *program* text expressed in a *programming language* or in a form produced by an *assembler*, *compiler*, or other *translator*.

07.04.12

coding (in computer programming)

The process of expressing a *program* in a *programming language*.

07.04.13

absolute code

Code in which all *addresses* are *absolute addresses*.

07.04.14

assembly code

Code expressed in a form that can be recognized and processed by an *assembler*.

07.04.15

assembled origin

The *address* of the initial *storage location* assigned to all or part of a *program* by an *assembler*, a *compiler*, or a *linkage editor*.

07.04.16

cross-assembler

An *assembler* that uses one *computer* to assemble a *program* into an *object language* of a different computer.

07.04.17

relocating assembler

An *assembler* the product of which is *relocatable*.

07.04.18

assemble-and-go

An operating technique in which there are no stops between the *assembling*, * *linking*, * *loading*, and *execution* of a *program*.

07.04.19

to compile

To *translate* all or part of a *program* expressed in a *high-level language* into a program expressed in an *intermediate language*, an *assembly language*, or a *machine language*.

07.04.20

compiler

A *translator* that can *compile*.

07.04.11

code (en programmation des ordinateurs)

Morceau du texte d'un *programme* exprimé dans un *langage de programmation* ou dans une forme produite par un *assembleur*, un *compilateur* ou tout autre *traducteur*.

07.04.12

programmation

Action d'exprimer un *programme* dans un *langage de programmation*.

07.04.13

code absolu

Code dans lequel les *adresses* sont des *adresses absolues*.

07.04.14

code assembleur

Code exprimé sous une forme qui peut être reconnue et traitée par un *assembleur*.

07.04.15

origine d'assemblage

Adresse de l'*emplacement de mémoire* initial assigné à tout ou partie d'un *programme* par un *assembleur*, un *compilateur* ou un *éditeur de liens*.

07.04.16

assembleur croisé

Assembleur qui utilise un *ordinateur* pour assembler un *programme* vers un *langage objet* d'un ordinateur de type différent.

07.04.17

assembleur translatif

Assembleur dont le résultat est *translatable*.

07.04.18

assemblage-exécution

Technique d'exploitation dans laquelle il n'y a pas d'interruption entre l'*assemblage*, l'*édition des liens*, le chargement et l'*exécution* d'un *programme*.

07.04.19

compiler

Traduire tout ou partie d'un *programme* exprimé en *langage évolué* vers un programme exprimé dans un *langage intermédiaire*, un *langage d'assemblage* ou un *langage machine*.

07.04.20

compilateur

Traducteur conçu pour *compiler*.

**07.04.21
compilation**

The process or the result of *compiling*.

**07.04.22
compilation unit**

All or part of a *program* expressed in a *high-level language* and sufficiently complete to be *compiled*.

**07.04.23
compiler code**

Code expressed in a form that can be recognized and processed by a *compiler*.

**07.04.24
compiler generator
compiler compiler
metacompiler**

A *translator* or *interpreter* used to specify and construct all or part of a *compiler*.

**07.04.25
cross-compiler**

A *compiler* that uses one *computer* to *compile* a *program* into an *object language* of a different computer.

**07.04.26
compile-and-go**

An operating technique in which there are no stops between the *compiling*, * *linking*, * *loading*, and *execution* of a *program*.

**07.04.27
to disassemble**

To *translate* * *object code* to an *assembly language* representation.

**07.04.28
to decompile**

To *translate* a *compiled* * *program* from its *machine language* version into a form that may resemble the original program in *high-level language*.

NOTE - A decompiled program should recompile into its original machine language version.

**07.04.29
decompiler**

A *software tool* that *decompiles* * *programs*.

**07.04.21
compilation**

Action de *compiler* ou résultat de cette action.

**07.04.22
unité de compilation**

Tout ou partie d'un *programme* exprimé en *langage évolué* et suffisamment complet pour pouvoir être *compilé*.

**07.04.23
code compilable**

Code exprimé sous une forme qui peut être identifiée et traitée par un *compilateur*.

**07.04.24
métacompilateur
compilateur de compilateur**

Traducteur ou *interpréteur* utilisé pour spécifier et construire tout ou partie d'un *compilateur*.

**07.04.25
compilateur croisé**

Compilateur qui utilise un *ordinateur* pour *compiler* un *programme* vers un *langage objet* d'un ordinateur de type différent.

**07.04.26
compilation-exécution**

Technique d'exploitation dans laquelle il n'y a pas d'interruption entre la *compilation*, l'*édition des liens*, le chargement et l'*exécution* d'un *programme*.

**07.04.27
désassembler**

Traduire un *code objet* en une représentation de *langage* d'*assemblage*.

**07.04.28
décompiler**

Traduire un *programme* obtenu par *compilation*, depuis sa forme en *langage machine*, vers une forme qui ressemble au programme original en *langage évolué*, sans lui être nécessairement identique.

NOTE - Un programme décompilé devrait pouvoir être recompilé dans sa version d'origine en langage machine.

**07.04.29
décompilateur**

Outil logiciel qui *décompile* des *programmes*.

07.04.30
to interpret

To analyze, *translate*, and *execute* each *statement* or each *language construct* in a *source program* before handling the next statement.

07.04.31
interpreter
interpretive program
A program that can interpret.

07.04.32
interpretive code
Code expressed in a form that can be recognized and processed by an interpreter.

07.04.33
machine code
Code expressed in a form that can be recognized and executed by the processing unit of a computer.

07.04.34
source language
A programming language used in a source program.

07.04.35
machine-dependent
Pertaining to software that relies on features unique to a particular kind of computer and, therefore, may be executed only on computers of that kind.

07.04.36
machine-independent
Pertaining to software that does not rely on features unique to a particular kind of computer and, therefore, may be executed on computers of more than one kind.

07.04.37
source program
A program that a particular translator can accept.

07.04.38
source code
*Code expressed in a form suitable for input to an assembler, * compiler, or other translator.*

07.04.30
interpréter
Analyser, traduire et exécuter chaque instruction ou chaque élément de langage d'un programme source avant de traiter l'instruction suivante ou l'élément suivant.

07.04.31
interpréteur
Programme conçu pour interpréter.

07.04.32
programme interprétable
Code exprimé sous une forme qui peut être reconnue et traitée par un interpréteur.

07.04.33
code machine
Code exprimé sous une forme qui peut être reconnue et exécutée par l'unité centrale d'un ordinateur.

07.04.34
langage source
Langage de programmation utilisé dans un programme source donné.

07.04.35
spécifique d'une machine
spécifique
Qualifie un logiciel qui est basé sur des caractéristiques propres à un type particulier d'ordinateur, et ne peut, par conséquent, être exécuté que sur des ordinateurs de ce type.

07.04.36
indépendant d'une machine
indépendant
Qualifie un logiciel qui n'est pas basé sur des caractéristiques propres à un type particulier d'ordinateur, et peut, par conséquent, être exécuté sur des ordinateurs de plusieurs types.

07.04.37
programme source
Programme que l'on peut soumettre à un traducteur déterminé.

07.04.38
code source
Code exprimé sous une forme convenant comme donnée d'entrée pour un assembleur, un compilateur ou tout autre traducteur.

07.04.39**source module****compilation unit** (deprecated in this sense)

All or part of a *source program* sufficiently complete for *translation*.

NOTE - See also *compilation unit*.

07.04.40**intermediate language**

A *target language* into which all or part of a *source program* or a single *statement*, in a *source language*, is *translated* before it is further translated or *interpreted*.

NOTE - For a further *translation* an intermediate language may serve as a source language.

07.04.41**root compiler**

A *compiler* that *compiles* into an *intermediate language* only.

NOTE - A root compiler, when combined with a *code generator*, comprises a full compiler.

07.04.42**code generator**

A *subprogram*, often part of a *compiler*, that transforms all or part of a *program* from some *intermediate language* into an *object language*.

07.04.43**source code generator**

A *software tool* that accepts as *input* the requirements or design for a *program* and produces *source code* that implements the requirements or design.

07.04.44**to parse**

To determine the syntactic structure of a *language construct* by decomposing it into *lexical tokens* and establishing the relationships among them.

Exemples: To parse *blocks* into *statements*, *statements* into *expressions*, *expressions* into *operators* and *operands*.

07.04.45**parser**

A *software tool* that *parses* *programs* or other *text*, often as the first step of assembly, *compilation*, interpretation, or analysis.

07.04.39**module source**

Tout ou partie d'un *programme source* suffisamment complet pour pouvoir être *traduit*.

NOTE - Voir aussi *unité de compilation*.

07.04.40**langage intermédiaire**

Langage cible dans lequel on *traduit* tout ou partie d'un *programme source* ou une simple *instruction* d'un *langage source*, avant d'effectuer une autre *traduction* ou une interprétation.

NOTE - Un langage intermédiaire peut servir de langage source pour une traduction ultérieure.

07.04.41**compilateur pivot**

Compilateur qui *compile* seulement vers un *langage intermédiaire*.

NOTE - Un compilateur pivot, lorsqu'il est combiné à un *générateur de code*, forme un compilateur complet.

07.04.42**générateur de code**

Sous-programme, souvent inclus dans un *compilateur*, qui transforme tout ou partie d'un *programme* depuis un *langage intermédiaire* vers un *langage objet*.

07.04.43**générateur de code source**

Outil logiciel auquel on indique, comme *données d'entrée*, les besoins ou la conception d'un *programme*, et qui fournit le *code source* qui met en œuvre ces besoins ou cette conception.

07.04.44**analyser**

Déterminer la structure syntaxique d'un *élément de langage* en le décomposant en ses *unités lexicales* et en établissant les relations entre elles.

Exemples: Décomposer les *blocs* en *instructions*, les *instructions* en *expressions*, les *expressions* en *opérateurs* et *opérandes*.

07.04.45**analyseur syntaxique**

Outil logiciel qui *analyse* les *programmes* ou autres *textes*, souvent comme première étape d'un assemblage, d'une *compilation*, d'une interprétation ou d'une autre forme d'analyse.

07.04.46

application generator

A *source code generator* that produces *programs* to solve one or more problems in a particular application area.

07.04.47

software tool

Software used in the development, testing, analysis, or maintenance of a *program* or its documentation.

Examples: Cross-reference generator, *decompiler*, driver, editor, flowcharter, *monitor*, test case generator, timing analyzer.

07.04.48

target language

A *language* in which a *translator* expresses its results.

07.04.49

target machine (1)

The *computer* on which a *program* is intended to be executed.

NOTE - See *host machine* (1).

07.04.50

target machine (2)

A *computer* that is being *emulated* by another computer.

NOTE - See *host machine* (2).

07.04.51

target program

The *translated* version of a *source program*.

07.04.52

host language

A *programming language* in which *statements* of a *data manipulation language* are embedded.

07.04.53

host machine (1)

A *computer* used to develop *software* intended for another computer.

NOTE - See *target machine* (1).

07.04.54

host machine (2)

A *computer* used to *emulate* another computer.

NOTE - See *target machine* (2).

07.04.46

générateur d'application

Générateur de code source qui produit des *programmes* pour résoudre un ou plusieurs problèmes dans un domaine d'application particulier.

07.04.47

outil logiciel

Logiciel utilisé pour développer, tester, analyser ou maintenir un *programme* ou sa documentation.

Exemples: Générateur de références croisées, *décompilateur*, pilote, éditeur, traceur d'organigramme, *moniteur*, générateur de tests, analyseur temporel.

07.04.48

langage cible

Langage dans lequel un *traducteur* exprime ses résultats.

07.04.49

machine cible (1)

Ordinateur sur lequel un *programme* doit s'exécuter.

NOTE - Voir *machine hôte* (1).

07.04.50

machine cible (2)

Ordinateur qui est *émulé* par un autre ordinateur.

NOTE - Voir *machine hôte* (2).

07.04.51

programme cible

Version résultant de la traduction d'un *programme source*.

07.04.52

langage hôte

Langage de programmation dans lequel sont emboîtées les *instructions* d'un *langage de manipulation de données*.

07.04.53

machine hôte (1)

Ordinateur utilisé pour développer du *logiciel* destiné à un autre ordinateur.

NOTE - Voir *machine cible* (1).

07.04.54

machine hôte (2)

Ordinateur utilisé pour *émuler* un autre ordinateur.

NOTE - Voir *machine cible* (2).

07.04.55

host machine (3)

The *computer* on which a *program* or *file* is installed.

07.04.56

object language

A *target language* for expressing *object programs*.

07.04.57

object code

A final version of *code* prior to *execution*.

NOTE - An *object program* is made up of object code.

07.04.58

object module

All or part of an *object program* sufficiently complete for *linking*.

NOTES

1. *Assemblers* and *compilers* usually produce object modules.
2. This entry is a modified version of the entry 10.02.10 in ISO 2382-10:1979.

07.04.59

object program

A *target program* that, if necessary, must be *linked*, before it can be *executed* by a particular *computer*.

07.04.60

translation time (1)

Any instant at which *translation* takes place.

07.04.61

compilation time (1)

Any instant at which *compilation* takes place.

07.04.62

assembly time (1)

Any instant at which assembly takes place.

07.04.63

translation duration

translation time (2)

The amount of time needed to *translate* a *program*.

07.04.55

machine hôte (3)

Ordinateur sur lequel un *programme* ou un *fichier* est installé.

07.04.56

langage objet

Langage cible conçu pour exprimer des *programmes objets*.

07.04.57

code objet

Code exprimé sous sa forme finale, avant son *exécution*.

NOTE - Un *programme objet* est constitué de code objet.

07.04.58

module objet

Tout ou partie d'un *programme objet*, suffisamment complet pour être soumis à l'*éditeur de liens*.

NOTES

1. Les *assembleurs* et les *compilateurs* produisent généralement des modules objets.
2. Cet article est une version modifiée de l'article 10.02.10 de la norme ISO 2382-10:1979.

07.04.59

programme objet

Programme cible qui doit, si nécessaire, être soumis à l'*éditeur de liens* avant de pouvoir être *exécuté* par un *ordinateur* déterminé.

07.04.60

instant de traduction

temps de traduction (1)

Tout instant où se déroule une *traduction*.

07.04.61

instant de compilation

temps de compilation (1)

Tout instant où se déroule une *compilation*.

07.04.62

instant d'assemblage

temps d'assemblage (1)

Tout instant où se déroule un *assemblage*.

07.04.63

durée de traduction

temps de traduction (2)

Durée nécessaire pour *traduire* un *programme*.

07.04.64

compilation duration

compilation time (2)

The amount of time needed to *compile* a *program*.

07.04.64

durée de compilation

temps de compilation (2)

Durée nécessaire pour *compiler* un *programme*.

07.04.65

assembly duration

assembly time (2)

The amount of time needed to *assemble* a *program*.

07.04.65

durée d'assemblage

temps d'assemblage (2)

Durée nécessaire pour *assembler* un *programme*.

07.04.66

translator directive

A *language construct* for controlling the *translation* of a *program*.

07.04.66

directive de traduction

Élément de langage fournissant des indications pour traduire un *programme*.

07.04.67

assembler directive

A *language construct* for controlling the *assembling* of a *program*.

07.04.67

directive d'assemblage

Élément de langage fournissant des indications pour assembler un *programme*.

07.04.68

compiler directive

A *language construct* for controlling the *compilation* of a *program*.

07.04.68

directive de compilation

Élément de langage fournissant des indications pour compiler un *programme*.

07.04.69

interpreter directive

A *language construct* for controlling the *interpretation* of a *program*.

07.04.69

directive d'interprétation

Élément de langage fournissant des indications pour interpréter un *programme*.

07.04.70

separate compilation

dependent compilation

The *compilation* of a *source module* using *data* representing interface and context relationships from related source modules.

07.04.70

compilation dépendante

Compilation d'un *module source* utilisant des *données* représentant les relations d'interface et de contexte fournies par les modules sources associés.

NOTE - Interface and context data are used by the *compiler* to check validity and to resolve references.

NOTE - Les données d'interface et de contexte sont utilisées par le *compilateur* pour vérifier la validité et pour résoudre les références.

07.04.71

independent compilation

separate compilation (deprecated in this sense)

The *compilation* of a *source module* not using *data* representing interface and context relationships from related source modules.

07.04.71

compilation indépendante

Compilation d'un *module source* n'utilisant pas de *données* représentant les relations d'interface et de contexte fournies par les modules sources associés.

NOTE - When independently *compiled* units are eventually combined, it may be necessary to check interface and context data for validity.

NOTE - Quand des unités *compilées* de façon indépendante sont ultérieurement combinées, il convient de vérifier la validité des données d'interface et de contexte.

07.04.72**generic unit**

A possibly parameterized model of a *language construct* from which, at *translation time* (1), a language construct proper is derived.

07.04.73**macrogenerator**

A *module*, often part of an *assembler* or *compiler*, that replaces each *macroinstruction* or *macrocall* in a *source program* with the appropriate *code* in accordance with the corresponding *macrodefinition*.

07.04.74**macroprocessor**

A *subprogram* provided in some *assemblers* and *compilers* to support *macrodefinitions*.

07.04.75**macroprogramming**

Programming using *macrodefinitions* and *macroinstructions* or *macrocalls*.

07.04.76**macro library**

A collection of *macrocalls* and *macroinstructions* together with their *macrodefinitions* available for use by a *macrogenerator*.

07.04.77**macroassembler**

An *assembler* that includes, or performs the functions of a *macrogenerator*.

07.04.78**program generator**

A *program* that can produce other programs.

07.04.79**preprocessor**

A *program* or *subprogram* that carries out some processing steps prior to a major *process*.

07.04.80**preprocessing**

Processing performed prior to a major *process*.

Example: *Translation* of an *embedded database language* * *statement*, such as SQL, into a *host language*.

07.04.72**unité générique**

Modèle éventuellement paramétré d'un *élément de langage*, dont est dérivé, au moment de la traduction, un élément de langage conforme.

07.04.73**macrogénérateur**

Module, souvent inclus dans un *assembleur* ou un *compilateur*, qui remplace chaque *macro-instruction* ou chaque *appel de macro* d'un *programme source* par la *macrodéfinition* correspondante.

07.04.74**macroprocesseur**

Sous-programme inclus dans certains *assembleurs* et *compilateurs* pour traiter les *macrodéfinitions*.

07.04.75**macroprogrammation**

Programmation utilisant des *macrodéfinitions* et des *macro-instructions* ou des *appels de macro*.

07.04.76**bibliothèque de macros**

Collection d'*appels de macros* et de *macro-instructions*, accompagnés de leurs *macrodéfinitions*, mise à la disposition d'un *macrogénérateur*.

07.04.77**macroassembleur**

Assembleur qui inclut un *macrogénérateur* ou en assume les fonctions.

07.04.78**générateur de programme**

Programme conçu pour produire d'autres programmes.

07.04.79**préprocesseur**

Programme ou *sous-programme* qui effectue certaines tâches de traitement avant un traitement plus important.

07.04.80**prétraitement**

Traitement effectué avant un traitement plus important.

Exemple: *Traduction* d'une *instruction* d'un *langage de base de données intégré*, tel que SQL, en *langage hôte*.

07.04.81

language preprocessor

A *functional unit* that effects preparatory processing of *programs*.

Example: A *macrogenerator* may function as a language preprocessor of a *translator*.

07.04.81

préprocesseur de langage

Unité fonctionnelle qui effectue des tâches préalables au traitement de *programmes*.

Exemple: Un *macrogénérateur* peut tenir lieu de préprocesseur de langage pour un *traducteur*.

07.05 Linking and loading

07.05 Lier et charger

07.05.01

to link

To interconnect *data objects* by establishing *pointers* or to interconnect portions of one or more *programs* by providing *links*.

Example: To link *object programs* by a *linkage editor*.

07.05.01

lier

Interconnecter des *objets de données* en établissant des *pointeurs*, ou interconnecter des parties d'un ou de plusieurs *programmes* en fournissant des *liens*.

Exemple: Lier des *programmes objets* par un *éditeur de liens*.

07.05.02

link

linkage

A part of a *program*, often a single *instruction* or *address*, that passes control and possibly *parameters* between separate *modules* of this program.

07.05.02

lien

Partie d'un *programme*, souvent composée d'une seule *instruction* ou d'une *adresse*, qui passe la commande, et éventuellement des *paramètres*, entre des *modules* du programme.

07.05.03

linkage editor

linker

A *program*, that, when processing one or more independently *translated* * *object modules* or *load modules*, resolves cross-references among these *modules*, provides *links* between or among them, establishes *relocatable* elements, and, if necessary, adjusts *addresses*, thus creating new load modules.

07.05.03

éditeur de liens

relieur

Programme qui, lorsqu'il est appliqué à un ou plusieurs *modules objets* ou *modules chargeables* * *traduits* indépendamment, résout les références croisées entre ces *modules*, fournit des *liens* entre eux ou parmi eux, établit des éléments *translatables* et, si nécessaire, ajuste les *adresses*, créant ainsi de nouveaux modules chargeables.

NOTE - This entry is a modified version of the entry 10.02.12 in ISO 2382-10:1979.

NOTE - Cet article est une version modifiée de l'article 10.02.12 de la norme ISO 2382-10:1979.

07.05.04

loader

A *program* that *copies* other programs from *external storage* to *internal storage* or *data* from external storage to internal storage or from internal storage to *registers*.

07.05.04

chargeur

Programme qui *copie* d'autres programmes de *mémoire externe* en *mémoire interne*, ou qui copie des *données* de mémoire externe en mémoire interne ou de mémoire interne vers des *registres*.

07.05.05

to load (in computer programming)

To *execute* a *loader*.

07.05.05

charger (en programmation des ordinateurs)

Exécuter un *chargeur*.

07.05.06**absolute loader**

A *program* that *copies* * *load modules*, within which all *addresses* are *absolute addresses*, from *external storage* to *internal storage*, thus address adjustment is unnecessary.

07.05.07**linking loader**

A *program* that combines the functional capabilities of a *linkage editor* and a *loader*.

07.05.08**load module**

All or part of a *program* that is suitable for *loading* and for *execution*.

NOTES

1. A load module is usually the result of applying a *linkage editor*.
2. This entry is a modified version of the entry 10.02.11 in ISO 2382-10:1979.

07.05.09**load-and-go**

An operating technique in which there are no stops between the *loading* and *execution* of a *program*.

07.05.10**loaded origin**

The *address* of the initial *storage location* of a *program* that has been *loaded* into *main storage*.

07.05.11**load map**

A *computer-generated* list that identifies the *storage location* or sizes of all or selected parts of *programs* or *data* residing in *memory*.

07.05.12**to relocate**

To move all or part of an *object program* in an *address space* and to make the necessary adjustment of *addresses* so that the corresponding *program* parts, resulting from this transformation, can be *executed* in the new location.

07.05.13**relocatable program**

An *object program* that is in such a form that it may be *relocated*.

07.05.06**chargeur absolu**

Programme qui *copie*, de *mémoire externe* en *mémoire interne*, des *modules chargeables* à l'intérieur desquels toutes les *adresses* sont des *adresses absolues*, ce qui rend inutile l'ajustement des *adresses*.

07.05.07**chargeur relieur**

Programme qui combine les possibilités fonctionnelles d'un *éditeur de liens* et d'un *chargeur*.

07.05.08**module chargeable**

Tout ou partie d'un *programme* prêt à être *chargé* et *exécuté*.

NOTES

1. Un module chargeable est généralement produit par un *éditeur de liens*.
2. Cet article est une version modifiée de l'article 10.02.11 de la norme ISO 2382-10:1979.

07.05.09**chargement-exécution**

Technique d'exploitation dans laquelle il n'y a pas d'interruption entre le *chargement* et l'*exécution* d'un *programme*.

07.05.10**adresse de chargement**

Adresse de l'*emplacement de mémoire* initial d'un *programme* qui a été *chargé* en *mémoire principale*.

07.05.11**plan d'occupation de la mémoire
carte de chargement**

Liste produite par un *ordinateur* et identifiant l'*emplacement de mémoire* ou la taille de tout ou partie des *programmes* ou *données* résidant en *mémoire*.

07.05.12**translater
reloger**

Transférer tout ou partie d'un *programme objet* dans un espace d'*adresses* et faire les modifications d'*adresses* nécessaires afin que les parties correspondantes du *programme*, résultant de cette transformation, puissent être *exécutées* dans le nouvel emplacement.

07.05.13**programme translatable
programme relogeable**

Programme objet dont la forme lui permet d'être *translaté*.

07.05.14
relocatable

Pertaining to all or part of an *object program* that can be loaded into any part of *main storage*.

NOTE - The starting *address* is established by the *loader*, which then adjusts the addresses to reflect the *storage locations* into which *program* parts have been loaded.

07.05.15
relocating loader

A loader that processes *relocatable programs* or *relocatable * modules*.

07.05.16
relocation dictionary

The part of an *object module* or *load module* that identifies the *addresses* that must be adjusted when it is *relocated*.

07.05.17
relocation offset

The difference between the *loaded origin* and the *assembled origin* of a *program*.

07.05.18
address offset

A number that must be added to a *relative address* to determine the *address* of the *storage location* to be accessed.

07.05.19
segmentation

A technique for *storage* allocation in which parts of a *program* are loaded from *auxiliary storage* into *main storage* when needed.

07.05.20
segment (in computer programming)

A portion of a *program* that may be *executed* without the entire program being *resident* in *main storage*.

07.05.21
overlay segment

Each of several *segments* of a *program* that, one at a time, occupy the same area of *main storage*, when *executed*.

07.05.14
translatable
relogeable

Qualifie tout ou partie d'un *programme objet* qui peut être chargé en un emplacement quelconque de la *mémoire principale*.

NOTE - L'*adresse* de départ est déterminée par le *chargeur*, qui ajuste ensuite les *addresses* de façon à refléter les *emplacements de mémoire* dans lesquels les diverses parties du *programme* devront être chargées.

07.05.15
chargeur traducteur
chargeur relogeur

Chargeur qui traite des *programmes translatables* ou des *modules * translatables*.

07.05.16
répertoire de translation
répertoire de relogement

Partie d'un *module objet* ou d'un *module chargeable* qui identifie les *addresses* qui doivent être ajustées lorsqu'on *translate* ce module.

07.05.17
déplacement de translation
déplacement de relogement

Différence entre l'*adresse de chargement* et l'*origine d'assemblage* d'un *programme*.

07.05.18
décalage d'adresse

Nombre qui doit être ajouté à une *adresse relative* pour déterminer l'*adresse* de l'*emplacement de mémoire* auquel on veut accéder.

07.05.19
segmentation

Technique d'attribution de *mémoire* selon laquelle des parties d'un *programme* sont chargées de *mémoire auxiliaire* en *mémoire principale*, lorsque nécessaire.

07.05.20
segment

Partie d'un *programme* qui peut être *exécutée* sans que la totalité de ce programme se trouve en *mémoire principale*.

07.05.21
segment recouvrant

Chacun des *segments* d'un *programme* qui occupent tour à tour le même emplacement en *mémoire principale* durant leur *exécution*.

**07.05.22
to overlay**

To *load* an *overlay segment* from *auxiliary storage* in such a manner that other portions of a *program* are overwritten.

**07.05.23
overlay supervisor**

A *subprogram* that controls the sequencing and positioning of *overlay segments*.

**07.05.24
resident** (adjective)

Pertaining to *programs*, parts thereof, or *data* while they remain in *main storage*.

NOTE - This entry is a modified version of the entry 10.02.16 in ISO 2382-10:1979.

**07.05.25
resident program**

A *program* that remains in a particular area of a *storage device*.

**07.05.26
activation record**

The *data object* that represents an instance of a *task* or of a *subprogram* and that contains *data values* and process status *data* for that instance.

NOTE - An activation record may contain *parameters*, results, local data, etc.

07.06 Program execution**07.06.01
language processor**

A *functional unit* for *translating* and *executing* * *programs* written in a specified *programming language*.

Example: A LISP machine.

**07.06.02
execution time
run time**

Any instant at which the *execution* of a particular *program* takes place.

**07.05.22
recouvrir**

Charger un *segment recouvrant* à partir d'une *mémoire auxiliaire*, de façon à recouvrir d'autres parties du *programme*.

**07.05.23
gestionnaire de segments recouvrants**

Sous-programme qui contrôle l'ordre et la position des *segments recouvrants*.

**07.05.24
résident** (adjectif)

Qualifie des *programmes*, des parties de programme ou des *données*, dans la mesure où ils se trouvent dans la *mémoire centrale*.

NOTE - Cet article est une version modifiée de l'article 10.02.16 de la norme ISO 2382-10:1979.

**07.05.25
programme résident**

Programme qui reste dans une certaine zone de *mémoire*.

**07.05.26
article d'activation**

Objet de données qui représente une instance d'une *tâche* ou d'un *sous-programme*, et qui contient des *valeurs de données* et des *données* d'état du processus, propres à cette instance.

NOTE - Un article d'activation peut contenir des *paramètres*, des résultats, des données locales, etc.

07.06 Exécution des programmes**07.06.01
processeur de langage**

Unité fonctionnelle destinée à *traduire* et à *exécuter* des *programmes* écrits dans un *langage de programmation* déterminé.

Exemple: Une machine LISP.

**07.06.02
instant d'exécution**

Tout instant où se déroule l'*exécution* d'un *programme* déterminé.

07.06.03
execution duration
run duration
running time

The amount of time needed for the *execution* of a particular *program*.

NOTE - Execution duration may be either *elapsed time* or *processor time*.

07.06.04
elapsed time

The span of time actually elapsed from the beginning to the end of the *execution* of a *program*.

NOTE - Contrast with *processor time*.

07.06.05
processor time

The sum of the time intervals in which *processors* actually *execute* a *program*.

NOTE - Contrast with *elapsed time*.

07.06.06
execution profile

A representation of the absolute or relative *execution* frequencies or *execution durations* of the *instructions* or of the *statements* of a *program*.

07.06.07
trace

A record of the *execution* of all or part of a *program*, showing the sequence of *instructions* or *statements* * *executed*, the *operands* involved and their names, and the *results*.

07.06.08
to trace
to produce a trace

07.06.09
execution trace
control-flow trace
code trace US

A record of the sequence of *instructions* * *executed* during the *execution* of a *program*.

07.06.10
retrospective trace

A *trace* produced after *execution* of a *program* has ended, from historical *data* recorded during the *execution*.

NOTE - Retrospective trace differs from an *execution trace* which is produced cumulatively during *execution*.

07.06.03
durée d'exécution

Temps nécessaire à l'*exécution* d'un *programme* déterminé.

NOTE - La durée d'exécution peut être soit une *durée chronométrée*, soit un *temps de processeur*.

07.06.04
durée chronométrée

Temps effectivement écoulé entre le début et la fin de l'*exécution* d'un *programme*.

NOTE - Comparer avec *temps de processeur*.

07.06.05
temps de processeur

Somme des intervalles de temps durant lesquels des *processeurs* * *exécutent* effectivement un *programme*.

NOTE - Comparer avec *durée chronométrée*.

07.06.06
profil d'exécution

Représentation des fréquences absolues ou relatives d'*exécution* ou des *durées d'exécution* des *instructions* d'un *programme*.

07.06.07
trace

Enregistrement de l'*exécution* de tout ou partie d'un *programme*, montrant la succession des *instructions* * *exécutées*, les *opérandes* concernés et leurs noms, et les *résultats*.

07.06.08
tracer
Produire une trace

07.06.09
trace d'exécution

Enregistrement de la succession des *instructions* * *exécutées* durant l'*exécution* d'un *programme*.

07.06.10
trace rétrospective

Trace produite après la fin de l'*exécution* d'un *programme*, à partir des *données* historiques enregistrées durant l'*exécution*.

NOTE - La trace rétrospective diffère de la *trace d'exécution* qui est produite de manière cumulative durant l'*exécution*.

07.06.11**subprogram trace**

A record of all or selected *subprogram calls* performed during the *execution* of all or part of a *program* and, optionally, the values of *parameters* passed to and *returned* by each *subprogram* or other *module*.

07.06.12**symbolic trace**

A record of the *statements* of the *source program* and the outcome of *jumps* when the *program* is *executed*, using symbolic, rather than actual values for *input data*.

07.06.13**symbolic execution**

A process that supports the analysis of *software* by simulating the *execution* of all or part of a *program*, using *symbols* for *input data*, such as names of *variables*, rather than actual values, and expressing program *outputs* as logical or mathematical *expressions* involving these symbols.

07.06.14**variable trace****data-flow trace****data trace**

A record of the names and values of *variables* * *accessed* or changed during the *execution* of a *program*.

07.06.15**execution monitor**

A *software tool* or *hardware device* that operates concurrently with a system or *functional unit* and supervises, records, analyzes, or verifies the operation of the system or functional unit.

07.06.16**to exit**

To *execute* an *instruction* or *statement* in a *program* or part thereof that terminates the *execution* of that program or part.

07.06.17**exit point**

A point in a *program*, * *module*, or *statement* at which *execution* of this program, module, or statement can terminate.

07.06.18**entry point****entrance**

A point in a *program*, * *module*, or *statement* at which *execution* of this program, module, or statement can begin.

07.06.11**trace des sous-programmes**

Enregistrement de tout ou partie des *appels de sous-programmes* effectués durant l'*exécution* de tout ou partie d'un *programme*, et, facultativement, des valeurs des *paramètres* passés ou *renvoyés* par chaque *sous-programme* ou par tout autre *module*.

07.06.12**trace symbolique**

Enregistrement des *instructions* du *programme source* et du résultat des *sauts* au fur et à mesure de l'*exécution* du *programme*, utilisant, pour les *données d'entrée*, des valeurs symboliques plutôt que les valeurs effectives.

07.06.13**exécution symbolique**

Processus d'analyse de *logiciel* consistant à simuler l'*exécution* de tout ou partie d'un *programme*, en utilisant des *symboles* pour les *données d'entrée*, par exemple les noms des *variables* plutôt que leurs valeurs effectives, et en exprimant les *sorties* du programme sous forme d'expressions logiques ou mathématiques se référant à ces symboles.

07.06.14**trace des variables**

Enregistrement des noms et des valeurs des *variables* utilisées ou modifiées durant l'*exécution* d'un *programme*.

07.06.15**moniteur d'exécution**

Outil logiciel ou dispositif matériel qui fonctionne concurremment à un système ou à une *unité fonctionnelle*, et supervise, enregistre, analyse ou vérifie le fonctionnement du système ou de l'unité fonctionnelle.

07.06.16**sortir**

Exécuter une *instruction* qui appartient à un *programme* ou à une partie de programme et qui met fin à l'*exécution* de ce programme ou de cette partie de programme.

07.06.17**point de sortie**

Point d'un *programme*, d'un *module* ou d'une *instruction*, auquel l'*exécution* de ce programme, de ce module ou de cette instruction peut s'arrêter.

07.06.18**point d'entrée**

Point d'un *programme*, d'un *module* ou d'une *instruction*, auquel l'*exécution* de ce programme, de ce module ou de cette instruction peut commencer.

07.06.19

reentry point

A point in a *program*, * *module*, or *statement* at which this program, module, or statement resumes *execution* following the execution of another program, module, or statement.

07.06.20

breakpoint

A point in a *program*, *module*, or *statement* where *execution* may be suspended depending on a specific condition or event.

NOTES

1. A breakpoint is *set* to permit manual or *automatic* monitoring of program performance or results.
2. There may be more than one breakpoint.

07.06.21

to set (a breakpoint)

To define both a *breakpoint* and an appropriate event that suspends *execution* of a *program*.

07.06.22

to initiate (a breakpoint)

To suspend *execution* of a *program* at a *breakpoint*.

07.06.23

code breakpoint control breakpoint

A *breakpoint* that depends upon *execution* of a specific *instruction*.

07.06.24

data breakpoint

A *breakpoint* that depends upon access to a specific *data object*.

07.06.25

dynamic breakpoint

A *breakpoint* for which the specific events or conditions that *initiate* it may change during the *execution* of its own or another *program*.

07.06.26

static breakpoint

A *breakpoint* that can be *set* during *compilation*.

Example: A breakpoint may be set at a *call* of a given *subprogram*.

07.06.19

point de réentrée

Point d'un *programme*, d'un *module* ou d'une *instruction*, auquel ce programme, ce module ou cette instruction reprend son *exécution* à la suite de l'exécution d'un autre programme, module ou instruction.

07.06.20

point d'arrêt

point d'interruption

Point d'un *programme*, d'un *module* ou d'une *instruction*, où l'*exécution* peut être suspendue en fonction d'une condition ou d'un événement particulier.

NOTES

1. Un point d'arrêt est *positionné* pour permettre une supervision manuelle ou *automatique* du déroulement du programme ou de ses résultats.
2. Il peut y avoir plusieurs points d'arrêt.

07.06.21

positionner (un point d'arrêt)

Définir à la fois un *point d'arrêt* et l'événement associé qui suspend l'*exécution* d'un *programme*.

07.06.22

déclencher (un point d'arrêt)

Suspendre l'*exécution* d'un *programme* à un *point d'arrêt*.

07.06.23

point d'arrêt d'exécution

Point d'arrêt qui dépend de l'*exécution* d'une *instruction* particulière.

07.06.24

point d'arrêt de référence

Point d'arrêt qui dépend de l'accès à un *objet de donnée* particulier.

07.06.25

point d'arrêt dynamique

Point d'arrêt pour lequel les conditions ou les événements particuliers qui le *déclenchent* peuvent changer durant l'*exécution* de son *programme* ou d'un autre programme.

07.06.26

point d'arrêt statique

Point d'arrêt qui peut être *positionné* durant la *compilation*.

Exemple: Un point d'arrêt peut être positionné lors d'un *appel* d'un *sous-programme* déterminé.

07.06.27**programmable breakpoint**

A *breakpoint* that *automatically* invokes a previously specified *debugging* process when *initiated*.

07.06.27**point d'arrêt programmable**

Point d'arrêt qui, lorsqu'il est *déclenché*, appelle automatiquement un processus, préalablement spécifié, de mise au point.

07.06.28**preamble breakpoint**

A *breakpoint* that is placed at an *entry point* to a *program* or *subprogram*.

07.06.28**point d'arrêt d'entrée**

Point d'arrêt placé sur un *point d'entrée* d'un *programme* ou d'un *sous-programme*.

07.06.29**postamble breakpoint**

A *breakpoint* that is placed at an *exit point* from a *program* or *subprogram*.

07.06.29**point d'arrêt de sortie**

Point d'arrêt placé sur un *point de sortie* d'un *programme* ou d'un *sous-programme*.

07.06.30**checkpoint**

A point in a *program*, suitable for interrupting the *execution* of this program, at which a *sequence* of *instructions* is inserted to record the status and the results, to inspect them, and to *restart*.

07.06.30**point de contrôle**

Point d'un *programme* où l'on peut *interrompre* l'*exécution* du programme, et où l'on a inséré une séquence d'*instructions* pour enregistrer l'état et les résultats, les inspecter, et *reprendre*.

07.06.31**to restart**

To resume the *execution* of a *program* using the *data* recorded at a *checkpoint*.

07.06.31**reprendre
redémarrer**

Poursuivre l'*exécution* d'un *programme* en utilisant les *données* enregistrées à un *point de contrôle*.

07.06.32**restart point
rescue point**

A point in a *program* at which its *execution* can be continued or resumed after having been interrupted at a *breakpoint* or a *checkpoint*.

07.06.32**point de reprise**

Point d'un *programme* où l'on peut continuer ou reprendre son *exécution* après qu'elle a été interrompue à un *point d'arrêt* ou à un *point de contrôle*.

07.06.33**to recover**

To establish a previous or new status of all or part of a system, *file*, * *database*, or other *resource*, or of the *execution* of a *program* so that required functions can be performed.

07.06.33**recupérer**

Établir ou rétablir un état de tout ou partie d'un système, d'un *fichier*, d'une *base de données* ou de toute autre ressource, ou de l'*exécution* d'un *programme*, de telle sorte que les fonctions requises puissent être effectuées.

07.06.34**recovery** (in computer programming)

The process or the result of *recovering*.

07.06.34**récupération** (en programmation des ordinateurs)

Action de *recupérer*, ou résultat de cette action.

07.06.35

forward recovery

A kind of *recovery* in which a system, *program*, * *file*, * *database*, or other *resource* is brought to a new, not previously occupied state in which it can perform required functions.

Example: The reconstruction of a file to a given state by updating an earlier version, using *data* recorded in a chronological record of changes made to the file.

07.06.36

backward recovery

A kind of *recovery* in which a system, *program*, * *file*, * *database*, or other *resource* is restored to a previous state in which it can perform required functions.

Example: The reconstruction of a file to a given state by reversing all changes made to the file since it was in that state.

07.06.37

inline recovery

Recovery performed by resuming work at a safe point preceding the occurrence of a *failure*.

07.06.38

starvation

A situation in which the *execution* of an *asynchronous* * *procedure* is incapable of proceeding within any predictable interval of time because *concurrent* asynchronous procedures retain required *resources*.

07.06.39

deadlock

A situation in which *data processing* is suspended because two or more devices or *concurrent* * *processes* are each awaiting *resources* assigned to the other(s) or because of other mutual dependencies.

Example: A situation in which a *program* A, with an exclusive lock on record X, asks for a lock on record Y, which is allocated to program B. Likewise, program B is waiting for exclusive control over record X before giving up control over record Y.

07.06.35

récupération par progression

Type de *récupération* dans lequel un système, un *programme*, un *fichier*, une *base de données* ou toute autre *ressource* est amené dans un état nouveau, non atteint auparavant, et dans lequel il pourra effectuer les fonctions requises.

Exemple: L'action d'amener un fichier à un état donné, en mettant à jour une version antérieure au moyen de *données* conservées dans un enregistrement chronologique des modifications apportées à ce fichier.

07.06.36

récupération par régression

Type de *récupération* dans lequel un système, un *programme*, un *fichier*, une *base de données* ou toute autre *ressource* est ramené dans un état antérieur dans lequel il pourra effectuer les fonctions requises.

Exemple: L'action d'amener un fichier à un état donné, en inversant toutes les modifications faites sur ce fichier depuis qu'il était dans cet état.

07.06.37

récupération en direct

Récupération effectuée en reprenant le travail à un point sûr précédant l'apparition d'une *défaillance*.

07.06.38

étrangement

Situation dans laquelle l'*exécution* d'une *procédure* * *asynchrone* est bloquée parce que des procédures *asynchrones concurrentes* mobilisent, pour une durée imprévisible, les *ressources* nécessaires.

07.06.39

interblocage

étreinte fatale

Situation dans laquelle tout *traitement des données* est suspendu parce que plusieurs appareils ou plusieurs *processus* * *concurrents* attendent chacun des *ressources* affectées à l'un des autres, ou à cause d'une autre dépendance mutuelle.

Exemple: Situation dans laquelle un *programme* A, avec un verrouillage exclusif sur l'enregistrement X, demande un verrouillage sur l'enregistrement Y qui est attribué au programme B. De même, le programme B attend un contrôle exclusif de l'enregistrement X avant de renoncer au contrôle de l'enregistrement Y.

07.06.40**lockout**

A technique for allocation of *resources* in which shared resources are protected by permitting *access* by only one device or *process* at a time and excluding others.

Example: To prohibit *reading* of *data* while they are being updated.

07.06.41**bootstrap****initial program load****IPL (abbreviation)**

A short *program* that is permanently *resident* or easily *loaded* into a *computer* and whose *execution* brings a larger program, such as an *operating system* or its *loader*, into *memory*.

07.06.42**to bootstrap**

To *execute* a *bootstrap*.

07.06.43**bootstrap loader**

A short *program* used to *load* a *bootstrap*.

07.06.44**to boot**

To initialize a *computer* by *loading* the *operating system* and possibly *clearing* * *memory*.

07.06.45**exception**

A condition that may arise during *execution* of a *program*, that may cause a deviation from the normal *execution sequence*, and for which means exist to define, *raise*, recognize, ignore, or *handle* it.

Examples: "(ON ERROR) condition" in PL/I; *overflow*; range error.

07.06.46**to raise (an exception)**

To cause an *exception* to be signaled based upon the occurrence of a specified condition.

07.06.47**exception handler**

A portion of a *program* * *executed* in response to a specific kind of *exception*.

07.06.40**verrouillage**

Technique d'allocation des *ressources* d'un *ordinateur* dans laquelle des ressources partagées sont protégées par le fait qu'on ne permet l'*accès* qu'à un seul appareil ou un seul *processus* à la fois, à l'exclusion de tout autre.

Exemple: Interdiction de *lire* des *données* pendant une mise à jour.

07.06.41**amorçe**

Programme bref, *résidant* permanent ou facile à *charger* dans un *ordinateur*, et dont l'*exécution* a pour effet d'amener en *mémoire* un programme plus gros, tel qu'un *système d'exploitation* ou son *chargeur*.

07.06.42**amorcer**

Exécuter une *amorçe*.

07.06.43**chargeur d'amorçe**

Programme bref utilisé pour *charger* une *amorçe*.

07.06.44**lancer**

Faire démarrer un *ordinateur* en *chargeant* le *système d'exploitation* et éventuellement en *effaçant* la *mémoire*.

07.06.45**exception**

Situation qui peut se produire pendant l'*exécution* d'un *programme*, qui peut provoquer un écart par rapport à la *séquence d'exécution* normale, et pour laquelle il existe des moyens permettant de la définir, de la *déclencher*, de la reconnaître, de la *traiter* ou de ne pas en tenir compte.

Exemples: «(ON ERROR) condition» en PL/I; *dépassement de capacité*; dépassement de limite.

07.06.46**déclencher (une exception)**

Faire qu'une *exception* soit signalée lors de l'apparition d'une condition spécifiée.

07.06.47**processeur d'exception****gestionnaire d'exception**

Partie de *programme* * *exécutée* en réponse à un type particulier d'*exception*.

07.06.48

to handle (an exception)

To take direct action as the result of the occurrence of an *exception*.

NOTE - Normally, control is transferred to an *exception handler* that takes action.

07.06.49

to propagate (an exception)

To transfer control to the *exception handler* of a prior *calling * module* or *nesting module* due to lack of required *handling* within a given module, or to explicitly *raise* the *exception* again within an exception handler.

07.06.50

addressing exception

An *exception* that occurs when a *program* calculates an *address* outside the bounds of the space available to it.

07.06.51

data exception

An *exception* that occurs when a *program* attempts to use or *access * data* incorrectly.

07.06.52

operation exception

An *exception* that occurs when a *program* encounters an invalid *operation part*.

07.06.53

protection exception

An *exception* that occurs when a *program* attempts to *access* a protected area in a *storage device*.

07.06.54

overflow exception

An *exception* that occurs when the result of an *operation* causes an *overflow*.

07.06.55

underflow exception

An *exception* that occurs when the result of an *operation* causes an *arithmetic underflow*.

07.06.48

traiter (une exception)

gérer (une exception)

Agir immédiatement à la suite d'une *exception*.

NOTE - Normalement, la commande est transférée à un *processeur d'exception* qui agit de manière appropriée.

07.06.49

propager (une exception)

Transférer la commande à un *processeur d'exception* d'un *module * appelant* antérieur ou d'un module *emboîtant*, par suite du manque de traitement approprié à l'intérieur d'un module donné, ou bien *déclencher* à nouveau, et explicitement, l'*exception* à l'intérieur d'un *processeur d'exception*.

07.06.50

exception d'adressage

Exception qui se produit lorsqu'un *programme* calcule une *adresse* située en dehors des limites de l'espace qui est disponible pour ce programme.

07.06.51

exception de donnée

Exception qui se produit lorsqu'un *programme* essaie d'utiliser, ou d'accéder, de façon incorrecte à des *données*.

07.06.52

exception d'opération

Exception qui se produit lorsqu'un *programme* rencontre une *partie opérateur* invalide.

07.06.53

exception de protection

Exception qui se produit lorsqu'un *programme* essaie d'accéder à une zone de *mémoire* protégée.

07.06.54

exception de dépassement

Exception qui se produit lorsque le résultat d'une *opération* provoque un *dépassement de capacité*.

07.06.55

exception de soupassement

Exception qui se produit lorsque le résultat d'une *opération* provoque un *soupassement de capacité*.

07.07 Debugging and checking**07.07.01 (01.05.07)****to debug**

To detect, to locate, and to eliminate *errors* in *programs*.

07.07.02**debugger**

Software designed to assist *debugging*.

07.07.03**to dump**

To record or *display* in a format that facilitates analysis, the contents of all or part of a *storage device* at a particular instant.

Example: Dump formats include *internal storage*, such as the contents of *memory* and *general-purpose registers* and *external storage*, such as detailed structures of *data* on *disks* or *magnetic tapes*.

NOTE - Dumping is usually for the purpose of *debugging*.

07.07.04**dump (1)**

The process of *dumping*.

07.07.05**dump (2)****datadump**

Data that have been *dumped*.

07.07.06**selective dump**

A *dump* of designated *storage location* areas only.

07.07.07**change dump**

A *dump* of those *storage locations* whose contents have changed during a specified period.

07.07.08**postmortem dump**

A *dump* that is produced upon *abnormal termination* of the *execution* of a *program*.

07.07 Débogage et vérification**07.07.01 (01.05.07)****déboguer****mettre au point**

Détecter, localiser et éliminer des *erreurs* dans un *programme*.

07.07.02**débogueur**

Logiciel permettant le *débogage*.

07.07.03**clicher****vider**

Enregistrer ou *afficher* le contenu à un instant donné de tout ou partie d'une *mémoire* sous une forme qui en facilite l'analyse.

Exemple: Le vidage peut concerner d'une part la *mémoire interne*, notamment le contenu de la *mémoire centrale* ou des *registres généraux*, et d'autre part la *mémoire externe*, notamment les structures de *données* enregistrées sur *disque* ou *bandes magnétiques*.

NOTE - Un *clichage* se fait, en général, pour aider au *débogage*.

07.07.04**clichage****vidage**

Action de *vider*.

07.07.05**résultat de clichage****données clichées**

Données qui ont été vidées.

07.07.06**clichage sélectif****vidage sélectif**

Clichage qui ne concerne que certains *emplacements de mémoire*.

07.07.07**clichage après changement****vidage après changement**

Clichage des *emplacements de mémoire* qui ont été modifiés au cours d'une certaine période.

07.07.08**clichage d'autopsie****vidage d'autopsie**

Clichage produit à la suite de l'arrêt anormal de l'*exécution* d'un *programme*.

07.07.09

snapshot dump

A copy of all or portions of the *data* contained in *memory* or in a *database* at a particular point in time.

07.07.10

memory dump

A *dump* of the contents of all or part of the *internal storage* of a *computer*.

NOTE - Usually in *binary*, * *octal*, or *hexadecimal* form.

07.07.11

desk checking

A static analysis technique, perhaps including manual simulation of *program* * *execution*, in which *source code*, test results, or other documentation are visually examined, usually by the person who generated them, to identify *faults*, violations of development standards, or other problems.

NOTE - This entry is a modified version of the entry 20.05.02 in ISO/IEC 2382-20:1990.

07.07.12

playback

A technique in which a history of *execution* of all or part of a *program* is recorded in such a manner that the *input* and *output* can be regenerated under the user's control, perhaps in either the forward or backward direction.

NOTE - Playback is used in *debugging*.

07.07.13

replay

A technique in which the *input data* are captured and can be reintroduced into a *program* in such a manner as to cause *execution* of the program under controlled conditions for analysis.

07.07.14

single-step operation

single-step execution

step-by-step operation

A mode of operation of a *computer* in which a single *instruction*, or part of an instruction, is *executed* in response to an external *signal*.

NOTE - Single-step operation is used in *debugging*.

07.07.09

instantané

Copie de tout ou partie des *données* contenues à un instant précis dans une *mémoire* ou dans une *base de données*.

07.07.10

image mémoire

clichage mémoire

vidage mémoire

Clichage du contenu de tout ou partie de la *mémoire interne* d'un *ordinateur*.

NOTE - Il se fait souvent sous une forme *binnaire*, * *octale* ou *hexadécimale*.

07.07.11

contrôle de programmation

vérification pas à pas

Technique d'analyse statique, pouvant comporter une simulation manuelle de l'*exécution* d'un *programme*, qui consiste en l'examen visuel, en principe par la personne qui les a produits, de *codes sources*, de résultats de tests ou autres, pour identifier des *anomalies*, des écarts par rapport aux normes de développement ou tout autre problème.

NOTE - Cet article est une version modifiée de l'article 20.05.02 de la norme ISO/CEI 2382-20:1990.

07.07.12

exécution inversée

Technique consistant à enregistrer l'historique de l'*exécution* de tout ou partie d'un *programme* de telle sorte que les *entrées* et les *sorties* puissent être régénérées sous le contrôle de l'utilisateur, parfois même en allant en avant ou en arrière.

NOTE - On utilise cette technique pour le débogage.

07.07.13

réexécution

Technique consistant à saisir les *données d'entrée* que l'on peut ensuite réintroduire dans un *programme* de façon à provoquer l'*exécution* contrôlée de ce programme dans le but de l'analyser.

07.07.14

exécution pas à pas

Mode de fonctionnement d'un *ordinateur* dans lequel, en réponse à un *signal* externe, on exécute une seule *instruction* ou partie d'instruction.

NOTE - L'exécution pas à pas est utilisée pour le débogage.

07.07.15

diagnostic program

A *program* that is designed to detect, locate, and describe *faults* in equipment or *errors* in programs.

07.07.15

programme de diagnostic

Programme qui permet de détecter, de localiser et de décrire des *anomalies* dans les matériels ou des *erreurs* dans les programmes.

07.07.16

trace program

A *program* that produces a *trace*.

07.07.16

programme de trace

programme de jalonnement

Programme qui produit une *trace*.

07.07.17

operation code trap

A specific modification of the *operation part* of a *machine instruction* that causes an *interrupt* when that machine instruction is *executed*.

07.07.17

dérouteur

Modification spécifique de la *partie opérateur* d'une *instruction machine* qui engendre une *interruption* lors de l'*exécution* de cette instruction machine.

07.07.18

checking program

A *diagnostic program* that examines *source programs* or *data* for incorrect syntax, semantics, or lack of conformity to specified requirements.

07.07.18

programme de contrôle

Programme de diagnostic qui analyse des *programmes sources* ou des *données* dans le but d'y trouver une syntaxe ou une sémantique incorrectes ou un défaut de conformité à certaines spécifications.

07.07.19

patch

A direct modification of an *object module*, or a *loaded * program* without *assembling* or *compiling* anew from the *source program*.

07.07.19

retouche

pièce

Modification directe d'un *module objet* ou d'un *programme * chargé*, sans effectuer un nouvel assemblage ou une nouvelle *compilation* du *programme source*.

07.07.20

to patch

To make a *patch*.

07.07.20

rapiercer

Faire une *retouche*.

07.07.21

assertion

A *language construct* specifying a particular status that must exist or a particular condition that must be satisfied at a particular point of a *program* when it will be *executed*.

07.07.21

assertion

Élément de langage qui spécifie l'état particulier qui doit exister ou la condition particulière que l'on doit satisfaire en un certain point de l'*exécution* d'un *programme*.

07.07.22

loop assertion

An *assertion* that has to be verified throughout the *execution* of a *loop*.

07.07.22

assertion de boucle

Assertion qui doit être vérifiée tout au long de l'*exécution* d'une *boucle*.

07.07.23

invariant (adjective)

Pertaining to the property that something does not vary within a specified environment.

07.07.23

invariant

fixe

Qualifie ce qui ne varie pas dans un environnement donné.

07.07.24

loop invariant

A condition that is *invariant* throughout a *loop*.

07.07.25

precondition

An *assertion* that pertains to a point immediately preceding, in the *execution sequence*, a specified portion of a *program*.

07.07.26

postcondition

An *assertion* that pertains to a point immediately following, in the *execution sequence*, a specified portion of a *program*.

07.07.27

correctness proving

A formal mathematical demonstration that the semantics of a *program* is consistent with the specifications of that program.

07.07.28

proof of correctness

A proof that results from applying *correctness proving*.

07.07.29

formal specification (in computer programming)

A specification written in a formal notation, often for use in *correctness proving*.

07.07.30

partial correctness

Correctness proving indicating that a *program's* * *output* * *assertions* follow logically from its *input* assertions and processing steps.

07.07.31

total correctness

Correctness proving indicating that a *program's* * *output* * *assertions* follow logically from its *input* assertions and processing steps, and that, in addition, the program terminates under all specified input conditions.

07.07.24

invariant de boucle

Condition qui est *invariante* tout au long de l'*exécution* d'une *boucle*.

07.07.25

précondition

postulat d'entrée

Assertion qui doit être vérifiée en un point qui, dans la *séquence d'exécution*, précède immédiatement une partie déterminée du *programme*.

07.07.26

postcondition

Assertion qui doit être vérifiée en un point qui, dans la *séquence d'exécution*, suit immédiatement une partie déterminée du *programme*.

07.07.27

démonstration d'exactitude

Démonstration mathématique formelle pour démontrer que la sémantique d'un *programme* est conforme aux spécifications de ce programme.

07.07.28

preuve d'exactitude

preuve de correction

Preuve résultant d'une *démonstration d'exactitude*.

07.07.29

spécification formelle (en programmation des ordinateurs)

Spécification écrite en notation formelle, souvent utilisée pour une *démonstration d'exactitude*.

07.07.30

correction partielle

exactitude partielle

Démonstration d'exactitude montrant que les *assertions* * *de sortie* d'un *programme* découlent logiquement de ses *assertions d'entrée* et des étapes de traitement.

07.07.31

correction globale

exactitude complète

Démonstration d'exactitude montrant que non seulement les *assertions* * *de sortie* d'un *programme* découlent logiquement des *assertions d'entrée* et des étapes de traitement, mais encore que le programme se termine pour toutes les conditions d'entrée spécifiées.

07.07.32
error seeding
bug seeding
fault seeding

The process of intentionally adding known *faults* in a *program* for the purpose of monitoring the rate of detection and removal, and estimating the number of unknown faults remaining in the program.

07.07.33
indigenous error
indigenous fault

A *fault* in a *program* that has not been purposely inserted as part of an *error seeding* process.

07.07.34
error control software

Software that monitors a *data processing system* to detect, record, and possibly to correct *errors*.

07.07.35
error prediction

A quantitative statement about the expected number or nature of *errors* in a system or component.

07.07.36
unrecoverable error

An *error* for which *recovery* is impossible without the use of recovery techniques external to the *program*.

07.08 Microprogramming

07.08.01
microinstruction

A directive that specifies one or more of the basic *operations* needed to carry out a *machine instruction* or another self-contained *hardware* function, and that denotes the *operands* belonging to these operations.

NOTE - Microinstructions are the true machine instructions, and *microcode* is used to create a *virtual machine* that appears to have a more *user-friendly* * *instruction set*.

07.08.02
microprogramming

Programming using *microinstructions*.

NOTE - Microprogramming is an alternative to hard-wiring the control *signals* necessary to carry out *machine instructions*.

07.07.32
injection d'anomalies
implantation d'anomalies

Ajout intentionnel, dans un *programme*, d'*anomalies* connues dans le but d'en suivre le taux de détection et de suppression, et d'estimer ainsi le nombre d'anomalies inconnues résiduelles.

07.07.33
anomalie intrinsèque
faute intrinsèque

Dans un *programme*, * *anomalie* qui ne provient pas d'une *injection d'anomalies*.

07.07.34
logiciel de surveillance

Logiciel qui assure la surveillance d'un *système informatique* pour déceler, enregistrer et éventuellement corriger des *erreurs*.

07.07.35
prédiction des erreurs

Évaluation quantitative du nombre et de la nature des *erreurs* dans le fonctionnement d'un système ou d'un composant.

07.07.36
erreur irréparable
erreur non redressable

Erreur pour laquelle la *récupération* n'est possible que par des moyens extérieurs au *programme*.

07.08 Microprogrammation

07.08.01
micro-instruction

Directive spécifiant les *opérations* de base nécessaires pour effectuer une *instruction machine* ou toute autre fonction matérielle autonome, et identifiant les *opérandes* de ces opérations.

NOTE - Les micro-instructions sont les véritables instructions machine, et le *microcode* est utilisé pour créer une *machine virtuelle* qui semble disposer d'un *jeu d'instructions* plus *convivial*.

07.08.02
microprogrammation

Programmation utilisant des *micro-instructions*.

NOTE - La microprogrammation s'oppose au câblage des *signaux* de commande nécessaires pour effectuer une *instruction machine*.

07.08.03

microprogram

A sequence of *microinstructions* that, together with corresponding *hardware* components, controls the performance of a *machine instruction* or another self-contained hardware function.

07.08.04

microcode

A collection of *microinstructions*, comprising part of, all of, or a set of *microprograms*.

07.08.05

microcode assembler

A *program* that *translates* * *microprograms* from symbolic form into *binary* form.

07.08.06

microoperation

In *microprogramming*, one of the basic *operations* needed to carry out a *machine instruction* or another self-contained *hardware* function.

07.08.07

microprogrammable computer

A *computer* in which *microprograms* can be created or altered by the user.

07.09 Instructions and addresses

07.09.01

instruction

statement (deprecated in this sense)

The specification of an *operation* and the identification of any associated *operands*.

07.09.02

machine instruction

An *instruction* that can be directly *executed* by a *computer*.

NOTE - A machine instruction is an element of a *machine language*.

07.09.03

instruction format

The layout of the constituent parts of an *instruction*.

07.08.03

microprogramme

Séquence de *micro-instructions* qui, associée aux composants matériels correspondants, effectue une *instruction machine* ou une autre fonction matérielle autonome.

07.08.04

microcode

Ensemble de *micro-instructions* comprenant une partie, ou la totalité ou un ensemble de *microprogrammes*.

07.08.05

assembleur de microcode

micro-assembleur

Programme qui *traduit* des *microprogrammes* de forme symbolique en forme *binaire*.

07.08.06

microopération

micro-opération

En *microprogrammation*, chacune des *opérations* de base nécessaires pour effectuer une *instruction machine* ou une autre fonction matérielle autonome.

07.08.07

ordinateur microprogrammable

Ordinateur dans lequel des *microprogrammes* peuvent être créés ou modifiés par l'utilisateur.

07.09 Instructions et adresses

07.09.01

instruction

Spécification d'une *opération* et identification de ses *opérandes*.

07.09.02

instruction machine

Instruction qui peut être directement *exécutée* par un *ordinateur*.

NOTE - Une instruction machine fait partie d'un *langage machine*.

07.09.03

format d'instruction

modèle d'instruction

Disposition des différentes parties d'une *instruction*.

07.09.04**instruction set****instruction repertoire**

The complete set of *instructions* recognized by a given *computer* or provided by a given *programming language*.

07.09.04**jeu d'instructions**

Ensemble des *instructions* reconnues par un *ordinateur* déterminé ou fournies par un *langage de programmation* donné.

07.09.05**instruction length**

The number of *words*, * *bytes*, or *bits* needed to *store* a *machine instruction*.

07.09.05**longueur d'instruction**

Nombre de *mots*, de *multiplets* ou de *bits* nécessaires pour *ranger en mémoire* une *instruction machine*.

07.09.06**operation part****operation field**

The part of a *machine instruction* or *microinstruction* that specifies the *operation* to be performed.

07.09.06**partie opérateur****partie type d'opération**

Partie d'une *instruction machine* ou d'une *micro-instruction* qui spécifie l'*opération* à effectuer.

07.09.07**address**

A value that identifies a location.

Exemples: A *register* number, the address of a particular part of *storage device*, a device address, a *network* address.

07.09.07**adresse**

Valeur identifiant un emplacement.

Exemples: Un numéro de *registre*, l'adresse d'une partie déterminée de *mémoire*, l'adresse d'un dispositif, une adresse *réseau*.

07.09.08**address part**

The part of a *machine instruction* or *microinstruction* that specifies the *address* of an *operand*.

07.09.08**partie adresse****zone adresse**

Partie d'une *instruction machine* ou d'une *micro-instruction* qui spécifie l'*adresse* d'un *opérande*.

07.09.09**address format**

The number and arrangement of elements within an *address*.

Exemples: *Page* and offset in a *virtual-address* system; *channel*, device, *sector*, and *record* in *magnetic disk storage*.

07.09.09**format d'adresse****structure de l'adresse**

Nombre et disposition des éléments d'une *adresse*.

Exemples: Numéro de *page* et décalage dans un système à *adresses virtuelles*; canal, disque, *secteur* et *enregistrement* dans une *mémoire à disques magnétiques*.

07.09.10**instruction code****computer instruction code****machine code** (deprecated in this sense)

The set of *bytes* for representing the permissible different *machine instructions* of a particular *computer*.

07.09.10**code des instructions d'un ordinateur**

Ensemble de *multiplets* représentant les diverses *instructions machine* acceptables pour un *ordinateur* donné.

07.09.11
operation code
opcode

The *encoded* representation of the *operation part* of a *machine instruction*.

Exemple: In an *assembly language*, BNZ might be used to designate the *operation* "branch if not zero", which might be ultimately represented in *machine code* as a specific *bit* pattern.

07.09.12
zero-address instruction

An *instruction* that has no *address part*.

Exemples: Certain instructions for a stack machine; a HALT instruction.

07.09.13
one-address instruction
single-address instruction

An *instruction* that contains one *address part*.

Exemple: An instruction to *load* the contents of *storage location A*.

07.09.14
two-address instruction

An *instruction* that contains two *address parts*.

Exemple: An instruction to add the contents of *storage location A* to the contents of *storage location B*.

07.09.15
three-address instruction

An *instruction* that contains three *address parts*.

Exemple: An instruction to add the contents of *storage locations A* and *B*, and place the result in *storage location C*.

07.09.16
N-address instruction

An *instruction* that contains *N address parts*, where *N* may be any non-negative *integer*.

07.09.17
one-plus-one address instruction

An *instruction* that contains two *address parts*, the second containing the *address* of the instruction to be *executed* next.

Exemple: An instruction to *load* the contents of *storage location A*, then execute the instruction in *storage location B*.

07.09.11
code opération

Représentation *codée* de la *partie opérateur* d'une *instruction machine*.

Exemple: Dans un *langage d'assemblage*, l'expression BNZ peut être utilisée pour désigner l'*opération* «branchement si non zéro» qui est représentée en *code machine* par une configuration de *bits* particulière.

07.09.12
instruction sans adresse

Instruction ne contenant pas de *partie adresse*.

Exemples: Certaines instructions dans une machine à pile; une instruction HALTE.

07.09.13
instruction à une adresse

Instruction contenant une seule *partie adresse*.

Exemple: Instruction de chargement du contenu d'un *emplacement de mémoire A*.

07.09.14
instruction à deux adresses

Instruction contenant deux *parties adresses*.

Exemple: Instruction pour additionner le contenu de l'*emplacement de mémoire A* au contenu de l'*emplacement de mémoire B* déterminé.

07.09.15
instruction à trois adresses

Instruction contenant trois *parties adresses*.

Exemple: Instruction pour additionner les contenus des *emplacements de mémoire A* et *B* avec rangement du résultat en *C*.

07.09.16
instruction à N adresses

Instruction contenant *N parties adresses*, où *N* peut être tout nombre entier non négatif.

07.09.17
instruction à 1 + 1 adresses

Instruction contenant deux *parties adresses*, la seconde contenant l'*adresse* de l'instruction suivante à *exécuter*.

Exemple: Instruction qui ordonne de *charger* le contenu de l'*emplacement de mémoire A*, puis d'exécuter l'instruction située en *B*.

07.09.18**implicit addressing****implied addressing**

A method of *addressing* in which the *operation part* of an *instruction* also denotes the location of one or more of the *operands*.

Exemple: If a *computer* has only one *accumulator*, an instruction that refers to the accumulator needs no address information describing it.

07.09.18**adressage implicite**

Méthode d'*adressage* selon laquelle la *partie opérateur* d'une *instruction* indique aussi l'emplacement d'un ou plusieurs *opérandes*.

Exemple: Si un *ordinateur* n'a qu'un *accumulateur*, une instruction concernant l'accumulateur n'a pas besoin d'autre information pour le décrire.

07.09.19**one-ahead addressing**

A method of *implicit addressing* in which the *operands* for an *instruction* are understood to be in the *storage locations* following the locations of the operands used for the last instruction *executed*.

07.09.19**adressage à progression automatique**

Méthode d'*adressage implicite* selon laquelle les *opérandes* d'une *instruction* sont supposés se trouver aux emplacements de mémoire suivant ceux des opérandes utilisés pour la dernière instruction *exécutée*.

07.09.20**repetitive addressing**

A method of *implicit addressing* in which the *operation* of an *instruction* is understood to address the *operands* of the last instruction *executed*.

07.09.20**adressage sous-entendu****adressage répétitif**

Méthode d'*adressage implicite* selon laquelle une *instruction* utilise les mêmes *opérandes* que la dernière instruction *exécutée*.

07.09.21**direct instruction****immediate instruction**

An *instruction* that contains the value of an *operand* rather than its *address*.

07.09.21**instruction à opérande immédiat**

Instruction qui contient la valeur d'un *opérande* plutôt que son *adresse*.

07.09.22**immediate operand**

An *operand* whose value rather than its *address* is contained in an *instruction*.

07.09.22**opérande immédiat**

Opérande dont la valeur et non l'*adresse* est contenue dans une *instruction*.

07.09.23**immediate data**

Data contained in an *instruction*.

07.09.23**donnée immédiate**

Donnée contenue dans une *instruction*.

07.09.24**indirect instruction**

An *instruction* that contains an *indirect address*.

07.09.24**instruction indirecte**

Instruction qui contient une *adresse indirecte*.

07.09.25**no-op****no-operation instruction**

An *instruction* whose *execution* causes the *computer* to perform no other *operation* other than to proceed to the next instruction to be *executed*.

07.09.25**instruction ineffective**

Instruction qui donne l'ordre à l'*ordinateur* de ne pas faire d'autre *opération* que de traiter l'instruction qui suit immédiatement.

07.09.26

privileged instruction

An *instruction* that can be *executed* only in a specific mode.

07.09.27

jump instruction

An *instruction* that specifies a *jump*.

07.09.28

unconditional jump instruction

A *jump instruction* that specifies a mandatory *jump*.

07.09.29

conditional jump instruction

A *jump instruction* that specifies a condition for the *jump*.

07.09.30

calling sequence

A sequence of *instructions* that causes the *execution* of a *subprogram*, provides it, if necessary, with *data* to be processed, and that controls the delivery of results (if any) and the *return* to the *calling * program*.

07.09.31

address space

The set of *addresses* that can be used by a particular *program* or *functional unit*.

NOTE - The address space may include *virtual addresses*.

07.09.32

symbolic address

An *identifier* that represents an *address*.

07.09.33

direct address

An *address* that identifies a location without reference to a *storage location* containing another address.

NOTE - The location may be a storage location or a device.

07.09.34

base address

An *address* used as the origin in the calculation of addresses.

07.09.26

instruction privilégiée

Instruction qui ne peut être *exécutée* que dans un mode particulier.

07.09.27

instruction de saut

Instruction qui spécifie un *saut*.

07.09.28

instruction de saut inconditionnel

Instruction de saut qui spécifie un *saut* obligatoire.

07.09.29

instruction de saut conditionnel

Instruction de saut qui spécifie une condition pour le *saut*.

07.09.30

séquence d'appel

Séquence d'*instructions* qui provoque l'*exécution* d'un *sous-programme*, lui fournit, si nécessaire, les *données* à traiter et contrôle la sortie des résultats éventuels et le *retour* au *programme * appelant*.

07.09.31

espace adresse

espace d'adressage

Ensemble des *adresses* qui peuvent être utilisées par un *programme* déterminé ou une *unité fonctionnelle*.

NOTE - L'espace adresse peut inclure des *adresses virtuelles*.

07.09.32

adresse symbolique

Identificateur qui représente une *adresse*.

07.09.33

adresse directe

Adresse qui identifie un emplacement sans recourir à un *emplacement de mémoire* contenant une autre adresse.

NOTE - L'emplacement peut être un emplacement de mémoire ou un dispositif.

07.09.34

adresse de base

Adresse utilisée comme point de départ pour le calcul des adresses.

07.09.35**absolute address**

A *direct address* that identifies a location without reference to a *base address*.

NOTE - An absolute address may itself be a base address.

07.09.36**relative address**

A *direct address* that identifies a location by means of its displacement from a *base address*.

07.09.37**indirect address****multilevel address**

An *address* that identifies the *storage location* of another address.

NOTE - The designated storage location may contain the address of the desired *operand* or another indirect address; this chain of addresses eventually leads to the operand.

07.09.38**relocatable address**

An *address* that needs to be adjusted when *data* to which it refers are *relocated* or the *program* containing that address is relocated.

07.09.39**generated address**

An *address* that has been calculated during the execution of a *program*.

07.09.40**address modification**

Any *arithmetic operation*, * *logic operation*, or syntactic operation performed on an *address*.

07.09.41**effective address**

The *address* that results from performing any required indexing, indirect addressing, or other *address modification* on a specified address.

NOTE - If the specified address requires no address modification, it is also the effective address.

07.09.42**virtual address**

In a *virtual storage system*, the *address* assigned to a *storage location* in *external storage* to allow that location to be accessed as though it were part of *main storage*.

07.09.35**adresse absolue**

Adresse directe qui identifie un emplacement sans utiliser d'*adresse de base*.

NOTE - Une adresse absolue peut être elle-même une adresse de base.

07.09.36**adresse relative**

Adresse directe qui identifie un emplacement au moyen de son écart par rapport à une *adresse de base*.

07.09.37**adresse indirecte**

Adresse qui identifie l'*emplacement de mémoire* d'une autre adresse.

NOTE - L'emplacement de mémoire désigné peut contenir l'adresse de l'*opérande* souhaité ou une autre adresse indirecte; cette chaîne d'adresses conduit finalement à l'opérande.

07.09.38**adresse translatable****adresse relogeable**

Adresse qui devra être adaptée lorsque les *données* auxquelles elle se réfère, ou le *programme* qui la contient, seront *translatés*.

07.09.39**adresse calculée**

Adresse évaluée pendant l'*exécution* d'un *programme*.

07.09.40**modification d'adresse**

Toute *opération arithmétique*, * *opération logique* ou *opération* syntaxique réalisée sur une *adresse*.

07.09.41**adresse effective**

Adresse qui résulte de l'exécution d'une indexation, d'un adressage indirect ou de toute autre *modification d'adresse* sur une adresse particulière.

NOTE - Si l'adresse particulière ne nécessite pas de modification d'adresse, elle est aussi l'adresse effective.

07.09.42**adresse virtuelle**

Dans un système à *mémoire virtuelle*, adresse affectée à un *emplacement de mémoire* dans une *mémoire externe* pour en permettre l'accès comme s'il faisait partie d'une *mémoire centrale*.

07.09.43

real address

The *address* of a *storage location* in the *main storage* part of a *virtual storage* system.

07.09.44

index (in programming)

An *integer* that identifies the position of a *data* item in a *sequence* of data items.

07.09.45

indexed address

An *address* that is to be modified by the contents of one or more *index registers*.

07.09.46

self-relative address

An *address* that must be added to the address of the *instruction* in which it appears to obtain the address of the *storage location* to be accessed.

07.09.47

structure chart

hierarchy chart

A diagram that identifies the *modules*, activities, or other entities in a system or *program* and shows how larger or more general entities break down into smaller, more specific entities.

NOTES

1. The result is not necessarily the same as that shown in a *call graph*.
2. See Figure 1.

07.09.48

call graph

call tree

A diagram that identifies the *modules* in a system or *program* and shows which modules *call* one another.

NOTES

1. The result is not necessarily the same as that shown in a *structure chart*.
2. See Figure 2.

07.09.49

control flow diagram

control flow graph

A diagram that depicts the set of all possible *sequences* in which *operations* may be performed during the *execution* of a *program*.

07.09.43

adresse réelle

Adresse d'un *emplacement de mémoire* dans la *mémoire centrale* d'un système à *mémoire virtuelle*.

07.09.44

index (en programmation)

Nombre entier qui identifie la position d'une *donnée* dans une *séquence* de données.

07.09.45

adresse indexée

Adresse qui doit être modifiée par le contenu d'un ou plusieurs *registres d'index*.

07.09.46

adresse autorelative

adresse différentielle

Adresse qu'il faut ajouter à celle de l'*instruction* dans laquelle elle apparaît pour obtenir l'*adresse* de l'*emplacement de mémoire* concerné.

07.09.47

organigramme hiérarchique

Diagramme qui identifie les *modules*, les mouvements, ou les autres entités dans un système ou un *programme* et montre comment des entités générales et importantes se décomposent en entités plus spécifiques et plus petites.

NOTES

1. Le résultat n'est pas nécessairement le même que dans un *graphe d'appel*.
2. Voir Figure 1.

07.09.48

graphe d'appel

arbre d'appel

Dans un système ou un *programme*, diagramme qui identifie les *modules* et présente ceux d'entre eux qui peuvent mutuellement s'*appeler*.

NOTES

1. Le résultat n'est pas nécessairement le même que dans un *organigramme hiérarchique*.
2. Voir Figure 2.

07.09.49

diagramme de contrôle

graphe de contrôle

Diagramme représentant toutes les *séquences* possibles dans lesquelles les *opérations* peuvent être effectuées pendant l'*exécution* d'un *programme*.

07.09.50**box diagram****Chapin chart****Nassi-Shneiderman chart**

A *control flow diagram* consisting of *sequenced* and *nested* boxes that represent sequential steps, repetition, and *conditional statements*.

NOTE - See Figure 3.

07.09.51**data flow diagram****data flowchart****data flow graph**

A diagram that depicts *data sources*, * *data sinks*, * *data storage*, and *processes* performed on *data* as *nodes*, and logical flow of data as links between the nodes.

NOTE - See Figure 4.

07.09.52**bubble chart**

A diagram in which entities are depicted with circles (bubbles) and relationships are represented by links drawn between the circles.

NOTE - See Figure 5.

07.09.53**input-process-output chart****IPO chart**

A diagram of a *software* system or *module*, consisting of a rectangle on the left listing *inputs*, a rectangle in the center listing processing steps, a rectangle on the right listing *outputs*, and arrows connecting inputs to processing steps and processing steps to outputs.

NOTE - See Figure 6.

07.09.54**state transition diagram****state diagram**

A diagram that depicts the state that a system or component can assume, and shows the events or circumstances that cause or result from a change from one state to another.

NOTE - See Figure 7.

07.10 Concurrent processes**07.10.01****task state**

One of the conditions in which a *task* can be during its lifetime.

07.09.50**diagramme à pavés**

Diagramme de contrôle constitué de pavés en séquence et *emboîtés* représentant des étapes séquentielles, des répétitions et des *instructions conditionnelles*.

NOTE - Voir Figure 3.

07.09.51**diagramme de flux de données**

Diagramme qui représente les *sources de données*, les *collecteurs de données*, le *stockage des données* et les traitements sur les données sous forme de *nœuds*, et qui représente les flots logiques de données sous forme d'arcs reliant ces nœuds.

NOTE - Voir Figure 4.

07.09.52**diagramme à bulles**

Diagramme dans lequel les entités sont représentées sous forme de bulles reliées entre elles par des arcs représentant leurs relations.

NOTE - Voir Figure 5.

07.09.53**diagramme d'entrée-sortie**

Dans un *programme* ou un *module*, diagramme composé d'un rectangle à gauche listant les entrées, d'un rectangle au centre listant les phases de traitement, d'un rectangle à droite listant les *sorties*, et de flèches reliant les entrées aux phases de traitement et les phases de traitement aux sorties.

NOTE - Voir Figure 6.

07.09.54**diagramme des transitions d'état**

Diagramme qui représente l'état que peut prendre un système ou un composant et montre ce que provoque le changement d'un état en un autre ou ce qui résulte de ce changement.

NOTE - Voir Figure 7.

07.10 Processus concurrents**07.10.01****état de tâche**

Une des conditions dans lesquelles une *tâche* peut se trouver pendant sa durée de vie.

07.10.02

activation (in computer programming)
The establishment of an *activation record*.

07.10.03

elaboration

The process by which a *declaration* achieves its effect prior to *execution*, such as resolution of references, *data type* checking, or storage allocation.

07.10.04

executable (qualifier)

Pertaining to the *task state* in which a *task* exists after *activation* and before it is *completed*.

NOTES

1. An executable task is either *ready*, * *running*, or *blocked*.
2. See Figure 7.
3. The term used in Ada is "callable".

07.10.05

blocked (qualifier)

Pertaining to the *task state* of an *executable* * *task* in which the task is *delayed* or waiting for an event.

NOTE - See Figure 7.

07.10.06

ready (qualifier)

Pertaining to the *task state* of an *executable* * *task* in which the task is waiting for processing and is not *blocked*.

NOTE - See Figure 7.

07.10.07

running (qualifier)

Pertaining to the *task state* of an *executable* * *task* in which the task is currently assigned to a *processor*.

NOTE - See Figure 7.

07.10.08

delayed (qualifier)

Pertaining to the *task state* of an *executable* * *task* that is *blocked* by a *delay statement*.

NOTE - See Figure 7.

07.10.02

activation

Établissement d'un *article d'activation*.

07.10.03

élaboration

Processus par lequel une *déclaration* réalise son but avant l'*exécution*, tel que la résolution de références, la vérification du *type de données*, ou l'affectation de mémoire.

07.10.04

exécutable (qualificatif)

Relatif à l'*état de tâche* dans lequel se trouve une *tâche* après son *activation* et avant qu'elle ne soit *finie*.

NOTES

1. Une tâche exécutable est soit *prête*, soit *en cours*, soit *bloquée*.
2. Voir Figure 7.
3. Le terme utilisé en Ada est «appelable».

07.10.05

bloqué (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* * *exécutable*, dans lequel la tâche est *retardée* ou en attente d'un événement.

NOTE - Voir Figure 7.

07.10.06

prêt (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* * *exécutable*, dans lequel la tâche est en attente de traitement et non *bloquée*.

NOTE - Voir Figure 7.

07.10.07

en cours (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* * *exécutable* qui est actuellement affectée à un *processeur*.

NOTE - Voir Figure 7.

07.10.08

retardé (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* * *exécutable* qui est *bloquée* par une *instruction d'attente*.

NOTE - Voir Figure 7.

07.10.09**completed** (qualifier)

Pertaining to the *task state* of a *task* that is finished, all events dependent on that task having been resolved.

NOTES

1. In Ada, an *exception* * *raised* during *activation* would be such an unusual event causing a task to be completed.
2. See Figure 7.

07.10.10**terminated** (qualifier)

Pertaining to the *task state* of a *task* that is *completed*, all events dependent on that task having been resolved and its *activation record* being released.

NOTE - See Figure 7.

07.10.11**master task**

A *module* of a *program* whose *execution* creates a *task*.

07.10.12**task entry**

A place in a *task* that provides an interface for a *calling* * *module*.

07.10.13**guard**

A *conditional expression* used to determine the open or closed nature of an alternative in a *selective wait statement*.

07.10.14**open guard**

A *guard* whose condition evaluates to TRUE.

07.10.15**closed guard**

A *guard* whose condition evaluates to FALSE.

07.10.16**thread**

A *process* within another process that uses the *resources* of the latter process.

07.10.09**fini** (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* qui est achevée et dont tous les événements qui en dépendent ont été résolus.

NOTES

1. Dans Ada, une *exception* * *déclenchée* durant l'*activation* d'une tâche serait un événement inhabituel qui entraînerait la fin de la tâche.
2. Voir Figure 7.

07.10.10**terminé** (qualificatif)

Relatif à l'*état de tâche* d'une *tâche* qui est *finie*, dont tous les événements qui en dépendent ont été résolus, et dont l'*article d'activation* est libéré.

NOTE - Voir Figure 7.

07.10.11**tâche maître**

Dans un *programme*, * *module* dont l'*exécution* crée une *tâche*.

07.10.12**point d'entrée d'une tâche**

Endroit dans une *tâche* qui offre une interface pour un *module* * *appelant*.

07.10.13**sentinelle**

Expression conditionnelle utilisée pour déterminer la nature ouverte ou fermée d'une alternative dans une *instruction d'attente sélective*.

07.10.14**sentinelle ouverte**

Sentinelle dont la condition s'évalue à VRAI.

07.10.15**sentinelle fermée**

Sentinelle dont la condition s'évalue à FAUX.

07.10.16**fil**

Processus inclus dans un autre processus et qui utilise les *ressources* de cet autre processus.

07.11 Support environments

07.11.01 stub

A substitute component that is used temporarily in a *program* so that progress can be made.

Example: In *compilation* or testing, a stub is used until the actual component becomes available.

07.11.02 scaffolding

Programs and *data* designed to support *software* development and testing, but not intended to be included in the final product.

Examples: Dummy *routines* or *files*, test case generators, software monitors, *stubs*.

07.11.03 programming system

In a *programming environment*, the *programming languages* and *software tools* required for the development and use of *programs* expressed in these programming languages.

07.11.04 program library

An organized collection of *programs*, or parts of programs, and possibly information pertaining to their use.

NOTE - A program library is often called according to the characteristic of its elements, for example, a *procedure* library, a *source program* library.

07.11.05 software library

A controlled collection of *software* and related documentation designed to aid in software development, use, or maintenance.

07.11.06 system library

A *software library* that is *resident* in a *data processing system* and that can be *accessed* for use or incorporated into other *programs* by reference.

Example: A *macro library*.

07.11 Environnements de support

07.11.01 élément de remplacement module de remplacement

Élément de substitution qui est employé de manière temporaire dans un *programme* pour permettre de progresser dans le développement de ce programme.

Exemple: Pendant la *compilation* ou le test, un élément de remplacement est utilisé jusqu'à ce qu'un élément réel soit disponible.

07.11.02 échafaudage

Programmes et *données* conçus pour soutenir le développement et les tests d'un *logiciel*, mais qui ne sont pas destinés à être inclus dans le produit final.

Exemples: Des *routines* ou des *fichiers* factices, des générateurs de cas de test, des moniteurs de logiciel, des *éléments de remplacement*.

07.11.03 système de programmation

Dans un *environnement de programmation*, ensemble des *langages de programmation* et des *outils logiciels* nécessaires au développement et à l'utilisation de *programmes* écrits dans ces langages de programmation.

07.11.04 bibliothèque de programmes programmathèque

Collection organisée de *programmes* ou de parties de programmes associés éventuellement à des informations relatives à leur utilisation.

NOTE - Une bibliothèque de programmes est souvent désignée en fonction des caractéristiques de ses éléments, par exemple une bibliothèque de *procédures*, une bibliothèque de *programmes sources*.

07.11.05 bibliothèque de logiciels

Collection contrôlée de *logiciels* et documentation associée, conçue pour aider au développement, à l'utilisation et à la maintenance du logiciel.

07.11.06 bibliothèque système

Bibliothèque de logiciels * *résidant* dans un *système informatique* et à laquelle on peut avoir accès pour l'utiliser ou pour l'incorporer par référence dans d'autres *programmes*.

Exemple: Une *bibliothèque de macros*.

07.11.07**operating environment**

The external surroundings of a *program* existing or expected to exist during its *execution*.

07.11.08**batch-processing environment**

An *operating environment* in which *input data* are collected and processed in groups, rather than being processed as each input arrives.

07.11.09**interactive environment**

An *operating environment* in which a *program* during its *execution* responds to each user and in which the user has the perception of directly influencing *operations* during the *process*.

07.11.10**real-time environment**

An *operating environment* that supports the *execution* of *real-time * programs*.

07.11.11**utility program**

A *program* that provides general, frequently needed services for *computer users* and service personnel.

Examples: A *diagnostic program*; a *trace program*; a *sort program*.

07.11.12**utility routine**

A *routine* that provides general, frequently needed services for *computer users* and service personnel.

Example: An *input routine*.

07.12 Goals and principles**07.12.01****modifiability**

A measure of the ease with which changes can be made to a *program*.

07.12.02**understandability**

A measure of the ease with which a person may read a *program* and of the ease of isolation of the *data structures* or *data objects* and *algorithms* in the solution that map to the real-world *data* and algorithms.

07.11.07**environnement opérationnel**

Environnement extérieur d'un *programme*, existant, ou dont l'existence est attendue, durant son *exécution*.

07.11.08**environnement de traitement par lots**

Environnement opérationnel dans lequel des *données d'entrée* sont rassemblées et traitées en groupes plutôt que d'être traitées au fur et à mesure des arrivées.

07.11.09**environnement interactif**

Environnement opérationnel dans lequel un *programme*, durant son *exécution*, réagit à chaque utilisateur et dans lequel l'utilisateur a la perception qu'il influence directement les opérations pendant le traitement.

07.11.10**environnement temps réel**

Environnement opérationnel qui permet l'*exécution* de *programmes * en temps réel*.

07.11.11**programme de service**

Programme qui fournit des services généraux, d'usage fréquent, pour les utilisateurs d'*ordinateurs* et le personnel informatique.

Exemples: Un *programme de diagnostic*, un *programme de trace*, un *programme de tri*.

07.11.12**routine de service**

Routine qui fournit des services généraux, d'usage fréquent, pour les utilisateurs d'*ordinateurs* et le personnel informatique.

Exemple: Une *routine d'entrée*.

07.12 Buts et principes**07.12.01****modifiabilité**

Mesure de la facilité avec laquelle des modifications peuvent être apportées à un *programme*.

07.12.02**intelligibilité**

Mesure de la facilité avec laquelle une personne peut lire un *programme* et de la facilité avec laquelle on peut identifier les *structures de données* ou les *objets de données* et les *algorithmes* dans la solution qui associe au monde réel les *données* et les algorithmes.

**07.12.03
modularity**

A measure of the extent to which a *program* is composed of *modules*, such that a change to one module has minimal impact on other modules.

**07.12.04
cohesion
module strength**

The manner and degree to which the activities of a single *module* are related to one another.

NOTES

1. Strong cohesion implies extensive relationships between activities of the module.
2. Kinds of cohesion may be ranked from strong to weak as follows:
functional cohesion
informational cohesion
communicational cohesion
temporal cohesion
logical cohesion
coincidental cohesion
3. Contrast with *coupling*.

**07.12.05
functional cohesion**

Cohesion in which the activities of a *module* all contribute to the performance of a single specified objective.

**07.12.06
informational cohesion**

Cohesion in which the activities of a *module* are performed on a common *data structure* but use independent *entry points* and *code*.

**07.12.07
communicational cohesion**

Cohesion in which the activities of a *module* use the same *input data* or contribute to producing the same *output data*.

**07.12.08
temporal cohesion**

Cohesion in which the activities of a *module* are all required at a particular time.

Example: A module that contains all the initializations of a *program*.

**07.12.09
logical cohesion**

Cohesion in which the activities of a *module* are logically similar.

Example: The processing of *data* from different *input media* in one module.

**07.12.03
modularité**

Mesure de la façon dont un *programme* est composé de *modules*, de telle sorte qu'une modification apportée à un module a un effet minimal sur les autres modules.

**07.12.04
cohésion**

Manière dont les activités d'un *module* unique sont en relation les unes avec les autres, et degré auquel elles sont en relation.

NOTES

1. Une cohésion forte signifie une interrelation étroite entre les activités du module.
2. Les différents types de cohésion peuvent être classés de fort à faible comme suit:
cohésion fonctionnelle
cohésion informationnelle
cohésion communicationnelle
cohésion temporelle
cohésion logique
cohésion accidentelle
3. Cohésion s'oppose à *couplage*.

**07.12.05
cohésion fonctionnelle**

Cohesion selon laquelle les activités d'un *module* contribuent toutes à remplir un objectif défini et unique.

**07.12.06
cohésion informationnelle**

Cohesion selon laquelle les activités d'un *module* sont appliquées à une *structure de données* unique mais utilisent des *points d'entrée* et des *codes* indépendants.

**07.12.07
cohésion communicationnelle**

Cohesion selon laquelle les activités d'un *module* utilisent les mêmes *données d'entrée* ou contribuent à produire les mêmes *données de sortie*.

**07.12.08
cohésion temporelle**

Cohesion selon laquelle les activités d'un *module* sont toutes nécessaires à un moment donné.

Exemple: Module qui contient toutes les initialisations d'un *programme*.

**07.12.09
cohésion logique**

Cohesion selon laquelle les activités d'un *module* sont logiquement semblables.

Exemple: Le traitement, dans un même module, de *données* provenant de différents supports *d'entrée*.

07.12.10**coincidental cohesion**

Cohesion in which the activities of a *module* have no functional relationship to one another.

07.12.11**procedural cohesion**

Cohesion in which the activities of a *module* all contribute to a given *procedure*, such as an *iteration* or decision process.

07.12.12**sequential cohesion**

Cohesion in which the result of one activity of a *module* serves as an *operand* for another subsequent activity performed by the same module.

07.12.13**coupling**

Interconnection or interdependence of different *modules*.

NOTES

1. Loose coupling implies little or no interconnection or interdependence.
2. Kinds of coupling may be ranked from loose to tight as follows:
 - no coupling
 - data coupling*
 - control coupling*
 - external coupling*
 - common-environment coupling*
 - content coupling*
3. Contrast with *cohesion*.

07.12.14**data coupling**

Coupling in which *data* are shared between *modules*.

07.12.15**control coupling**

Coupling in which one *module* passes *data* to another module for the explicit purpose of influencing the operation of the latter module.

07.12.16**external coupling**

Coupling in which the coupling of *variables* can be controlled by limiting it to those variables that are formally declared to be external.

NOTE - PL/1 is one of the *programming languages* with this capability.

07.12.10**cohésion accidentelle**

Cohésion selon laquelle les activités d'un *module* n'ont pas de relation fonctionnelle entre elles.

07.12.11**cohésion procédurale**

Cohésion selon laquelle les activités d'un *module* contribuent toutes à une *procédure* déterminée, comme une *itération* ou un processus de décision.

07.12.12**cohésion séquentielle**

Cohésion selon laquelle le résultat d'une activité d'un *module* sert d'*opérande* pour une autre activité accomplie ultérieurement par le même module.

07.12.13**couplage**

Interconnexion ou interdépendance de différents *modules* d'un *programme*.

NOTES

1. Un couplage lâche signifie peu ou pas d'interconnexion ou d'interdépendance.
2. Les types de couplage peuvent être classés du plus lâche au plus étroit comme suit:
 - absence de couplage
 - couplage de données*
 - couplage de commande*
 - couplage externe*
 - couplage d'environnement commun*
 - couplage de contenu*
3. Couplage s'oppose à *cohésion*.

07.12.14**couplage de données**

Couplage par lequel des *données* sont partagées entre des *modules*.

07.12.15**couplage de commande**

Couplage par lequel un *module* transmet des *données* à un autre module dans le but explicite d'influer sur le fonctionnement de cet autre module.

07.12.16**couplage externe**

Couplage par lequel le couplage de *variables* peut être contrôlé en le limitant aux variables qui sont formellement déclarées comme externes.

NOTE - PL/I est un des *langages de programmation* qui ont cette faculté.

07.12.17

**common-environment coupling
common coupling**

*Coupling in which modules * access common data.*

07.12.18

content coupling

Coupling in which one module refers to or changes the code of another module.

07.12.19

fan-in

The number of *modules* controlling a particular module.

NOTE - A high fan-in value suggests that *coupling* is high, because it is a measure of module dependencies.

07.12.20

fan-out

The number of *modules* controlled by a module.

NOTE - A high fan-out value suggests that the complexity of the *calling* module may be high because of the complexity of the logic required to control and coordinate the subordinate components.

07.12.21

localization

The principle applied to a set of *modules* which have the qualities of strong *cohesion* and loose *coupling*.

07.12.22

confirmability

A measure of the extent to which a *program* is designed and structured in such a manner that all of its parts can be readily tested.

07.12.17

couplage d'environnement commun

Couplage par lequel des modules ont accès à des données communes.

07.12.18

couplage de contenu

Couplage par lequel un module fait référence au code d'un autre module ou le modifie.

07.12.19

entrance

Nombre de *modules* commandant un module donné.

NOTE - Une valeur d'entrance élevée suggère que le *couplage* est important, car c'est une mesure des dépendances du module.

07.12.20

sortance

Nombre de *modules* commandés par un module.

NOTE - Une valeur de sortance élevée suggère que la complexité du module *appelant* peut être élevée en raison de la complexité de la logique nécessaire pour commander et coordonner les composants subordonnés.

07.12.21

localisation

Principe appliqué à un ensemble de *modules* qui ont une *cohésion* forte et un *couplage* lâche.

07.12.22

testabilité

Mesure de la manière dont un *programme* est conçu et structuré de façon que toutes ses parties puissent être facilement testées.

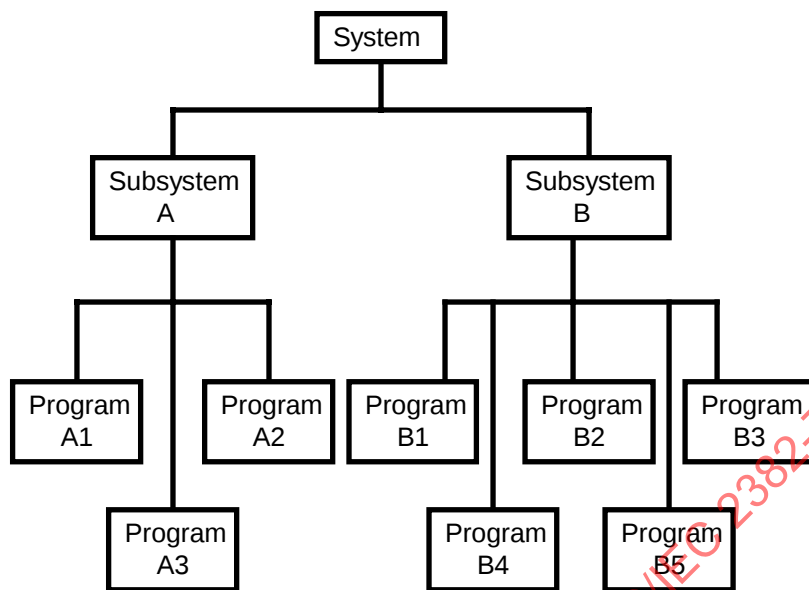


Figure 1 — An example of a structure chart

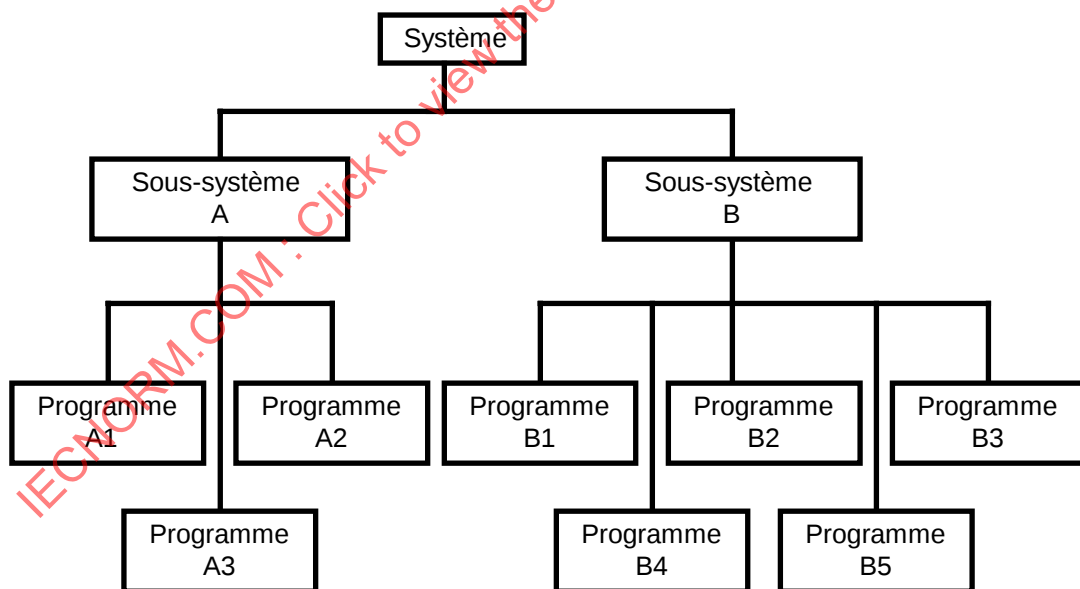


Figure 1 — Exemple d'organigramme hiérarchique

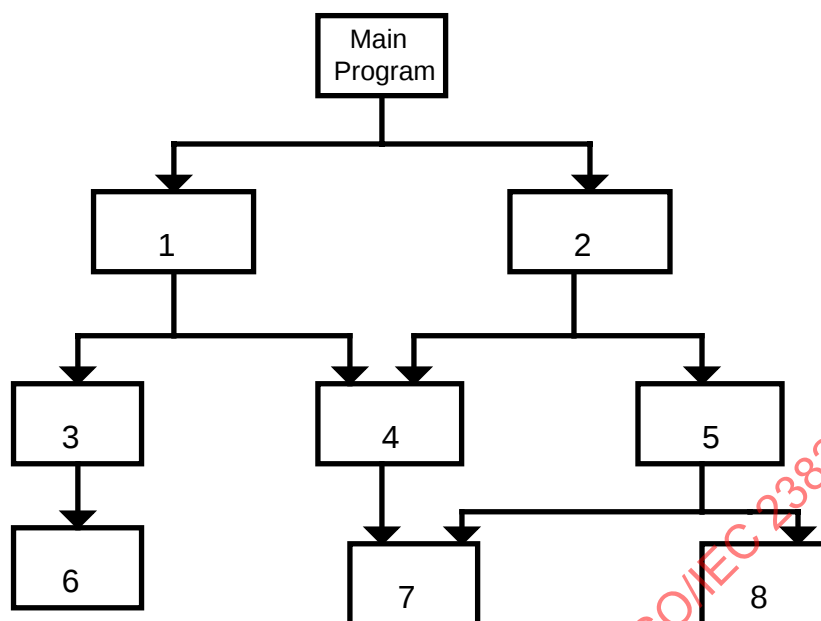


Figure 2 — An example of a call graph

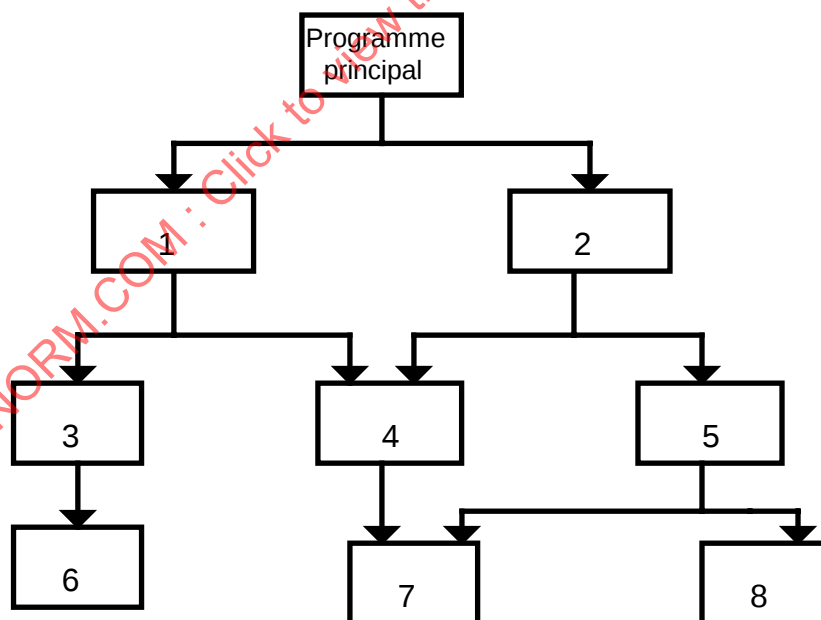


Figure 2 — Exemple de graphe d'appel

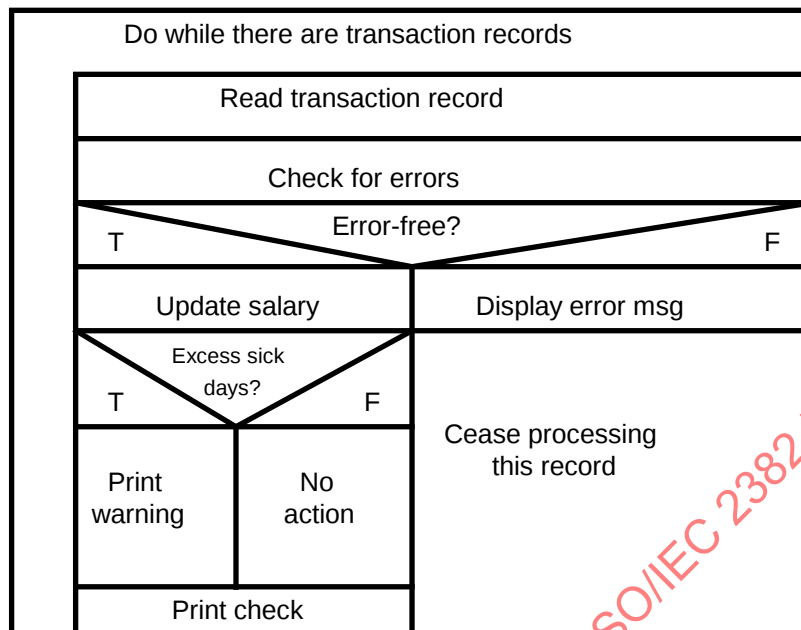


Figure 3 — An example of a box diagram

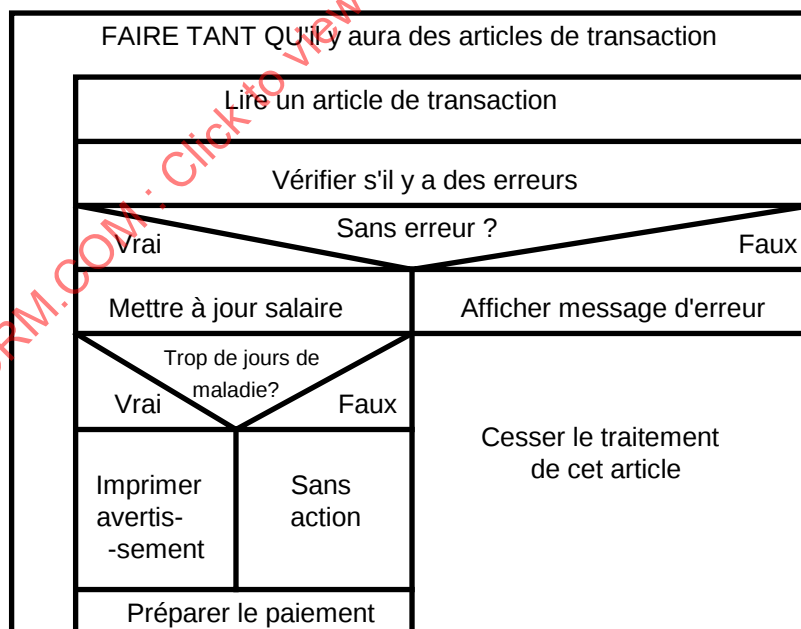


Figure 3 — Exemple de diagramme à pavés

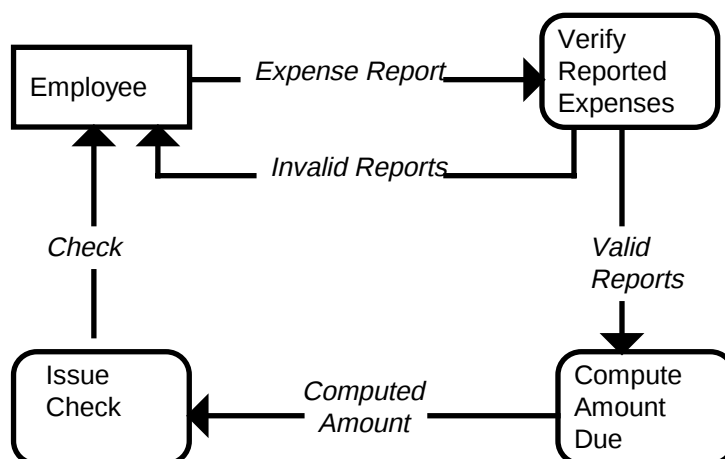


Figure 4 — An example of a data flow diagram

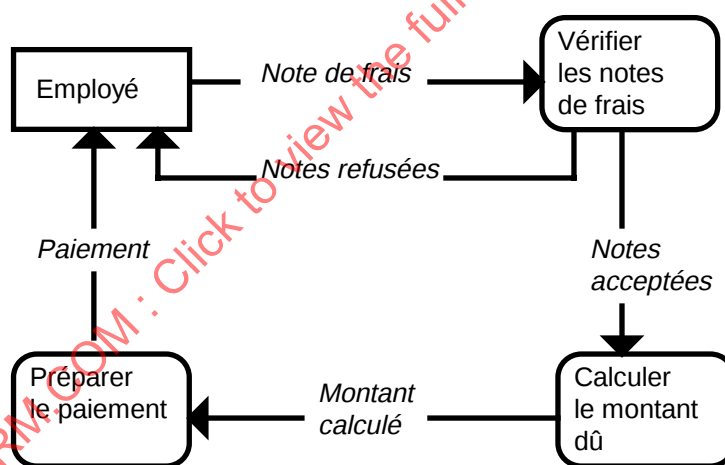


Figure 4 — Exemple de diagramme de flux de données