WIKIPÉDIA

Variable (informatique)

En informatique, les **variables** sont des symboles qui associent un nom (l'identifiant) à une valeur. Le nom est unique (et si le langage en possède, différents des mots-réservés [1]).

Dans la plupart des langages et notamment les plus courants, les variables peuvent changer de valeur au cours du temps (dynamique). Dans les langages de certains paradigmes, notamment la programmation fonctionnelle, leur valeur est au contraire figée dans le temps (statique).

Sommaire

Variables et constantes
Variables en programmation
Variables et pointeurs
Nom des variables
Typage
Cycle de vie des variables
Réflexivité
Notes et références
Voir aussi

Variables et constantes

Contrairement à une variable, une *constante* est un identificateur associé à une valeur fixe. Syntaxiquement, cet identificateur a tous les aspects d'une variable. Cependant, il lui est affecté une valeur définie, c'est-à-dire *constante*, comme la taille d'un plateau d'échecs (8x8). Une constante contient une valeur qui peut avoir des valeurs différentes suivant les exécutions, à la manière du jeu démineur dont le joueur peut choisir la taille du plateau.

Variables en programmation

Dans un langage de programmation, une variable est un espace de stockage pour un résultat. Cependant les possibilités d'une variable sont intimement liées au langage de programmation auquel on fait référence. Par exemple une variable en C++ aura six caractéristiques :

- son **nom** c'est-à-dire sous quel nom est déclarée la variable ;
- son **type**, c'est la convention d'interprétation de la séquence de bits qui constitue la variable. Le type de la variable spécifie aussi la longueur de cette séquence (8 bits, 32 bits, 64 bits) ;
- sa **valeur**, c'est la séquence de bits elle-même, elle ne peut varier au cours du temps si on utilise le mot-clef *const* ;
- son **adresse**, c'est l'endroit dans la mémoire où elle est stockée ;
- sa **portée**, c'est la portion de code source où elle est accessible, par exemple, la portée d'une variable (non globale) en C s'entend de sa définition à la fin du bloc où elle est définie.
- sa **visibilité**, c'est un ensemble de règles qui fixe qui peut utiliser la variable (exemple : mots-clefs *public, private, protected,* ou le masquage d'une variable par une autre) ;
- sa **durée de vie**, c'est le temps d'exécution pendant laquelle la variable existe. En C et en C++ une variable contenue dans un bloc de code limité par des accolades "{}" possède la durée de vie correspondant au temps

d'exécution de ce bloc. Il ne faut pas confondre la durée de vie d'une variable locale et sa visibilité, ou sa portée : une variable hors de portée (ex : masquée par une autre), existe toujours.

Toutefois on peut trouver des langages qui restreignent ces caractéristiques :

- le PHP ne possède pas un typage fort, comme une grande partie des langages scripts ;
- le Prolog ne permet pas qu'une variable change de valeur au cours du temps une fois la première identification faite ;
- la visibilité de la variable est public par défaut en C ;
- la notion de pointeur en C++ est remplacée par la notion de référence en Java, en Prolog l'adresse d'une variable n'est pas accessible au programmeur.

Variables et pointeurs

Dans tous les langages qui permettent la création dynamique d'adresses (new en C++, Java), la notion de variable est complétée par la notion de pointeur (type de donnée associé à une variable, dont les valeurs sont des adresses). L'adresse contenue dans une variable de type pointeur peut être inaccessible au programmeur (PHP, Java,Python), ou directement accessible (C++) [Comment ?].

Nom des variables

Dans certains langages, les noms de variables (comme ceux des identificateurs) doivent nécessairement commencer par une lettre (majuscule ou minuscule) ou par un _ (souligné). Les autres caractères composant le nom de la variable doivent être une lettre, un chiffre ou un _. La différenciation des majuscules et des minuscules (sensibilité à la casse) dans le nom d'une variable dépend du langage considéré.

Exemples de noms de variables valides, en C :

- `_var`
- `__var2`
- `Var`
- `vArIAbLe`
- `v_a_r`

Exemple de nom de variable non valide en C :

- `2var`

Ainsi, le premier caractère ne peut être un chiffre, car cela permet de faciliter la compilation ou l'interprétation du programme en ôtant une ambiguïté : quand le compilateur lit un chiffre, il sait que les caractères qui suivront constitueront une valeur numérique. De même, s'il lit une lettre ou un souligné, il saura qu'il a affaire à une variable.

Tous ces noms de variables sont valides en Lisp.

Typage

Lorsque le type d'une variable est déterminé à la compilation (explicitement par le programmeur ou automatiquement par inférence de types), on parle de typage statique. Les valeurs de cette variable devront être obligatoirement de ce type (au sens large, c'est-à-dire du même type ou d'un type dérivé)

Le typage statique aide à la génération de code objet plus efficace (en consommation mémoire et vitesse d'exécution). Il interdit toutefois la réflexivité à l'exécution.

Autrement, dans les cas où ce ne sont pas les variables qui ont un type, mais les valeurs, on parle de typage dynamique, ou typage latent.

On parle de typage fort lorsque le langage impose que les variables soient déclarées dans un type et utilisées dans ce type (ex : Ada ou C++). On parle de typage faible lorsque le langage admet qu'une variable puisse changer de type au cours de son existence (en particulier pour se conformer à la sémantique d'une expression).

Cycle de vie des variables

On distingue généralement cinq opérations sur les variables, chacune pouvant revêtir des formes syntaxiques différentes.

- la *déclaration* permet de déclarer un nom de variable, éventuellement de lui associer un type,
- la *définition* permet d'associer une zone mémoire qui va être utilisée pour stocker la variable, comme lorsqu'on lui donne une valeur initiale,
- l'*affectation* consiste à attribuer une valeur à une variable,
- la *lecture* consiste à utiliser la valeur liée à la variable,
- la *suppression* réalisée soit automatiquement soit par une instruction du langage.

Les langages, comme le C, Caml ou Pascal, imposent de déclarer une variable voire de lui donner un type avant son usage. La déclaration imposée des variables permet au compilateur ou à l'interpréteur d'identifier les erreurs typographiques comme des variables non déclarées ou des variables homonymes. D'autres langages effectuent la déclaration au moment de la première affectation (c'est le cas de la plupart des langages de script) ou lors de leur première apparition dans le code (comme dans Prolog).

En ce qui concerne l'initialisation des variables, c'est-à-dire l'association d'une première valeur, certains langages imposent d'initialiser une variable avant sa première lecture alors que d'autres fournissent une valeur implicite (spécifiée ou indéterminée). Des langages comme Oz ou Prolog ne réalisent pas à proprement parler d'initialisation. Lors de la déclaration des variables aucune valeur n'est associée, on dit que la variable n'est pas *liée*. La valeur de la variable est déterminée au fur et à mesure de l'exécution du programme, on parle alors d'*unification*.

Dans les langages de programmation fonctionnelle ou de programmation logique, les variables ne peuvent être associées qu'à une seule valeur au cours de leur existence.

Réflexivité

Dans la plupart des langages, les variables n'existent qu'en tant qu'outils pour le programmeur. Ainsi, renommer toutes les occurrences d'une variable ne modifiera pas le fonctionnement du programme.

Au contraire, pour offrir une expressivité supplémentaire, certains langages permettent de considérer un nom de variable comme une valeur comme une autre (par exemple, ce sont les symboles de Common Lisp et Smalltalk). C'est une technique très utile pour implémenter efficacement des algorithmes de calcul symbolique.

Notes et références

1. http://fr.wikibooks.org/wiki/Catégorie:Mots_réservés

Voir aussi

- Convention de nommage (programmation)
- Variable d'instance

Sur les autres projets Wikimedia :

variable, sur le Wiktionnaire

Variable (informatique), sur Wikibooks

La dernière modification de cette page a été faite le 26 septembre 2018 à 14:11.