Programmation informatique

Un article de Wikipédia, l'encyclopédie libre.

Aller à : navigation, rechercher

Cet article ne cite pas suffisamment ses sources (août 2016).

Si vous disposez d'ouvrages ou d'articles de référence ou si vous connaissez des sites web de qualité traitant du thème abordé ici, merci de compléter l'article en donnant les **références utiles à sa** vérifiabilité et en les liant à la section « Notes et références » (modifier l'article, comment ajouter mes sources ?).

Pour les articles homonymes, voir Programmation (homonymie).

Dans le domaine de l'informatique, la **programmation** est l'ensemble des activités qui permettent l'écriture des programmes informatiques. C'est une étape importante du développement de logiciels (voire de matériel).

Pour écrire un programme, on utilise un langage de programmation. Un logiciel est un ensemble de programmes (qui peuvent être écrits dans des langages de programmation différents) dédié à la réalisation de certaines tâches par un (ou plusieurs) utilisateurs du logiciel.

La programmation représente donc ici la rédaction du (ou des) code source d'un logiciel. On utilise plutôt le terme développement pour dénoter l'ensemble des activités liées à la création d'un logiciel et des programmes qui le composent (cela inclut la spécification du logiciel, sa conception, puis son implémentation proprement dite au sens de l'écriture des programmes dans un langage de programmation bien défini et aussi la vérification de sa correction)...

Sommaire

[masquer]

- 7.1 Articles connexes

Exemple de programme[modifier | modifier le code]

L'immense majorité des programmes qui s'exécutent sur nos ordinateurs, téléphones et autres outils électroniques sont écrits dans des langages de programmation dits impératifs : les lignes du programme sont exécutées les unes après les autres. Chaque ligne du programme effectue soit une opération simple, soit exécute une fonction qui est elle-même une suite d'opérations simples.

Le programme suivant écrit en langage Java (légèrement simplifié et auquel des commentaires ont été rajoutés), demande simplement à l'utilisateur d'entrer au clavier deux nombres entiers, et affiche leur quotient.

```
void main() { // fonction 'main' : c'est toujours ici qu'un programme commence
    // 'int' signifie integer : nombre entier en anglais
    int A, B; // on déclare deux variables A et B qui sont des nombres entiers
    WriteLine("entrez deux entiers : "); // 'WriteLine' permet d'écrire à
l'écran
    A = ReadIntFromKeyboard(); // on attend que l'utilisateur tape un entier au
clavier,
                               // et on l'enregistre dans A
    B = ReadIntFromKeyboard(); // puis on fait la même chose pour B
    if (B == 0) {  // si B est égal à 0
        WriteLine("erreur : division par zéro");
    } else { // sinon
        float C = CalculerDivision(A,B); // on exécute la fonction
'CalculerDivision'
            // que l'on a programmée ci-dessous,
            // et on enregistre le résultat dans C qui est un 'float' : un
nombre à virgule
        WriteLine("le résultat est : " + C); // on affiche C
    }
}
float CalculerDivision(float U, float V) { // U et V sont les paramètres de
notre fonction 'CalculerDivision' : ce sont des nombres à virgules (float).
   // et celle-ci renvoie un 'float' : un nombre à virgule
   // dans la fonction 'main', A et B étaient des nombres entiers,
   // U et V sont des copies des valeurs de A et B,
   // et qui ont été converties en nombres à virgule (22 deviendrait simplement
22.0000000)
   float Resultat;
   Resultat = U / V;  // on effectue la division
   return Resultat; /// on renvoie le résultat
}
```

Dans ce programme les principales fonctionnalités de la programmation impérative sont utilisées : des variables de type nombre entier, nombre à virgule, chaîne de caractère, fonction calculant un résultat à partir de paramètres, fonction effectuant une tâche telle qu'afficher un message à l'écran, instruction `if` permettant d'exécuter un code ou un autre en fonction de la valeur de telle ou telle variable.

Dans un programme informatique typique, on trouvera également des boucles `while` ou `for` qui permettent d'exécuter un morceau de code en boucle ou simplement un certain nombre de fois, des `new` pour l'allocation dynamique de données (par exemple des tableaux), et très souvent des éléments de programmation objet permettant de structurer différemment le code et de créer des types de données personnalisés, ou encore des exceptions pour gérer certain cas d'erreurs plus facilement.

On remarque que pour effectuer une tâche très simple, le code informatique est très laborieux, et encore ici on ne traite pas les erreurs (si l'utilisateur tape un mot au lieu d'un nombre), et l'affichage

est minimaliste. C'est pourquoi les langages de programmation n'ont jamais cessé d'évoluer, dans le but d'aider le programmeur : qui souhaite réaliser des programmes rapides à s'exécuter, sans dysfonctionnements, et surtout qui soient le plus simple à écrire.

Histoire[modifier | modifier le code]

La première machine programmable (c'est-à-dire machine dont les possibilités changent quand on modifie son « programme ») est probablement le métier à tisser de Jacquard, qui a été réalisé en 1801. La machine utilisait une suite de cartons perforés. Les trous indiquaient le motif que le métier suivait pour réaliser un tissage ; avec des cartes différentes le métier produisait des tissages différents. Cette innovation a été ensuite améliorée par Herman Hollerith d'IBM pour le développement de la fameuse carte perforée d'IBM.

En 1936, la publication de l'article fondateur de la science informatique *On Computable Numbers with an Application to the Entscheidungsproblem*[1] par Alan Mathison Turing allait donner le coup d'envoi à la création de l'ordinateur programmable. Il y présente sa machine de Turing, le premier calculateur universel programmable, et invente les concepts et les termes de programmation et de programme.

Les premiers programmes d'ordinateur étaient réalisés avec un fer à souder et un grand nombre de tubes à vide (plus tard, des transistors). Les programmes devenant plus complexes, cela est devenu presque impossible, parce qu'une seule erreur rendait le programme entier inutilisable. Avec les progrès des supports de données, il devient possible de charger le programme à partir de cartes perforées, contenant la liste des instructions en code binaire spécifique à un type d'ordinateur particulier. La puissance des ordinateurs augmentant, on les utilisa pour faire les programmes, les programmeurs préférant naturellement rédiger du texte plutôt que des suites de 0 et de 1, à charge pour l'ordinateur d'en faire la traduction lui-même.

Avec le temps, de nouveaux langages de programmation sont apparus, faisant de plus en plus abstraction du matériel sur lequel devaient tourner les programmes. Ceci apporte plusieurs facteurs de gains : ces langages sont plus faciles à apprendre, un programmeur peut produire du code plus rapidement, et les programmes produits peuvent tourner sur différents types de machines.

Phases[modifier | modifier le code]

Conception[modifier | modifier le code]

Articles détaillés : Conception de logiciel et Algorithmique.

La phase de conception définit le but du programme. Si on fait une rapide analyse fonctionnelle d'un programme, on détermine essentiellement les données qu'il va traiter (données d'entrée), la méthode employée (appelée l'algorithme), et le résultat (données de sortie). Les données d'entrée et de sortie peuvent être de nature très diverses. On peut décrire la méthode employée pour accomplir le but d'un programme à l'aide d'un algorithme. La programmation procédurale et fonctionnelle est basée sur l'algorithmique. On retrouve en général les mêmes fonctionnalités de base :

Programmation impérative[modifier | modifier le code]

Article détaillé : Programmation impérative.

"Si"

```
Si prédicat
    Alors faire ceci
    Sinon faire cela
```

"Tant que"

```
Tant que prédicat
    Faire ...
```

"Pour"

```
Pour variable allant de borne inférieure à borne supérieure
    Faire ...
```

"Pour" (variante)

```
Pour variable dans conteneur
    faire ...
```

Implémentation[modifier | modifier le code]

Article détaillé : Langage de programmation.

Une fois l'algorithme défini, l'étape suivante est de coder le programme. Le codage dépend de l'architecture sur laquelle va s'exécuter le programme, de compromis temps-mémoire, et d'autres contraintes. Ces contraintes vont déterminer quel langage de programmation utiliser pour « convertir » l'algorithme en code source.

Transformation du code source[modifier | modifier le code]

Le code source n'est (presque) jamais utilisable tel quel. Il est généralement écrit dans un langage "de haut niveau", compréhensible pour l'homme, mais pas pour la machine.

Compilation[modifier | modifier le code]

Articles détaillés : Compilation (informatique) et Machine virtuelle.

Certains langages sont ce qu'on appelle des langages compilés. En toute généralité, la compilation est l'opération qui consiste à transformer un langage source en un langage cible. Dans le cas d'un programme, le compilateur va transformer tout le texte représentant le code source du programme, en code compréhensible pour la machine, appelé code machine.

Dans le cas de langages dits compilés, ce qui est exécuté est le résultat de la compilation. Une fois effectuée, l'exécutable obtenu peut être utilisé sans le code source.

Il faut également noter que le résultat de la compilation n'est pas forcément du code machine correspondant à la machine réelle, mais peut être du code compris par une machine virtuelle (c'est-à-dire un programme simulant une machine), auquel cas on parlera de bytecode. C'est par exemple le cas en Java. L'avantage est que, de cette façon, un programme peut fonctionner sur n'importe quelle machine réelle, du moment que la machine virtuelle existe pour celle-ci.

Dans le cas d'une requête SQL, la requête est compilée en une expression utilisant les opérateurs de l'algèbre relationnelle. C'est cette expression qui est évaluée par le système de gestion de bases de données.

Interprétation[modifier | modifier le code]

Article détaillé : Interprétation.

D'autres langages ne nécessitent pas de phase spéciale de compilation. La méthode employée pour exécuter le programme est alors différente. La phase de compilation est la plupart du temps incluse

dans celle d'exécution. On dit de ce programme qu'il interprète le code source. Par exemple, Python ou Perl sont des langages interprétés.

Avantages, inconvénients[modifier | modifier le code]

Les avantages généralement retenus pour l'utilisation de langages "compilés", est qu'ils sont plus rapides à l'exécution que des langages interprétés, car l'interprète doit être lancé à chaque exécution du programme, ce qui mobilise systématiquement les ressources.

Traditionnellement, les langages interprétés offrent en revanche une certaine portabilité (la capacité à utiliser le code source sur différentes plates-formes), ainsi qu'une facilité pour l'écriture du code. En effet, il n'est pas nécessaire de passer par la phase de compilation pour tester le code source.

Appellation impropre[modifier | modifier le code]

Il faut noter qu'on parle abusivement de langages compilés ou interprétés. En effet, le caractère compilé ou interprété ne dépend pas du langage, qui n'est finalement qu'une grammaire et une certaine sémantique. D'ailleurs, certains langages peuvent être utilisés interprétés ou compilés. Par exemple, il est très courant d'utiliser Ruby avec un interprète, mais il existe également des compilateurs pour ce langage[2].

Néanmoins, l'usage qu'on fait des langages est généralement fixé.

Test du programme[modifier | modifier le code]

Article détaillé : Test (informatique).

C'est l'une des étapes les plus importantes de la création d'un programme. En principe, tout programmeur se doit de vérifier chaque partie d'un programme, de le tester. Il existe différents types de test. On peut citer en particulier :

- Test unitaire
- Test d'intégration
- Test de performance

Il convient de noter qu'il est parfois possible de vérifier un programme informatique, c'est-à-dire prouver, de manière plus ou moins automatique, qu'il assure certaines propriétés.

Pratiques[modifier | modifier le code]

- Algorithmique
- Gestion de versions
- Optimisation du code
- Programmation système
- Refactoring
- Test d'intégration
- Test unitaire

Paradigmes[modifier | modifier le code]

Article détaillé : Paradigme (programmation).

Un paradigme est un style fondamental de programmation, définissant la manière dont les programmes doivent être formulés.

- Programmation concurrente
- Programmation déclarative

- Programmation fonctionnelle
- Programmation impérative
- Programmation logique
- Programmation orientée aspect
- Programmation orientée composant
- Programmation orientée objet
- Programmation orientée prototype
- Programmation par contraintes
- Programmation par contrat
- Programmation par intention
- Programmation procédurale
- Programmation réactive
- Programmation structurée

Notes et références[modifier | modifier le code]

1. ↑ Alan Turing, « On Computable Numbers, with an Application to the Entscheidungsproblem », *Proceedings of the London Mathematical Society*, 1937 (ISSN 0024-6115 et 1460-244X, DOI 10.1112/PLMS/S2-42.1.230, lire en ligne [archive]) et « [*idem*] : A Correction », *Proc. London Math. Soc.*, 2e série, vol. 43, 1938, p. 544-546 (DOI 10.1112/plms/s2-43.6.544, lire en ligne [archive])
2. ↑ (en) *XRuby* [archive]

Voir aussi[modifier | modifier le code]

Articles connexes[modifier | modifier le code]

Sur les autres projets Wikimedia :

- Programmation informatique, sur Wikiversity
- Programmation informatique, sur Wikibooks

- Paradigme (programmation)
- Style de programmation
- Chronologie des langages de programmation
- Liste des langages de programmation
- Forme de Backus-Naur (BNF), une grammaire de description de langage
- Google Code Jam, un concours international annuel de programmation très prisé des mordus d'algorithmique et de programmation