Algorithmique et programmation

Université Nice SOPHIA ANTIPOLIS

Algorithmique

- L’**algorithme** est une méthode pour résoudre un problème
- Le **programme** est le codage lisible par l’ordinateur de cette méthode
- Avant d’écrire un programme, il est nécessaire d’avoir un algorithme

Il s'agit donc de fournir la solution à un problème, la première étape consiste donc à analyser le problème, c'est-à-dire en cerner les limites et le mettre en forme dans un langage descriptif, c'est à dire faire de l'algorithmique (simple description des étapes ou diagrammes)

L'étape suivante consiste à traduire l'algorithme dans un langage de programmation spécifique, il s'agit de la phase de programmation (dans divers langages).

Algorithmique

L'algorithme est un moyen pour le programmeur de présenter son approche du problème à d'autres personnes. En effet, un algorithme est l'énoncé dans un langage bien défini d'une suite d'opérations permettant de répondre au problème.

Un algorithme doit donc être :

- lisible: l'algorithme doit être compréhensible même par un non-informaticien

- de haut niveau: l'algorithme doit pouvoir être traduit en n'importe quel langage de programmation, il ne doit donc pas faire appel à des notions techniques relatives à un programme particulier ou bien à un système d'exploitation donné

- précis: chaque élément de l'algorithme ne doit pas porter à confusion, il est donc important de lever toute ambiguïté

- concis: un algorithme ne doit pas dépasser une page. Si c'est le cas, il faut décomposer le problème en plusieurs sous-problèmes

- structuré: un algorithme doit être composé de différentes parties facilement identifiables

Les langages de programmation

Le langage de programmation est l'intermédiaire entre l'humain et la machine, il permet d'écrire dans un langage proche de la machine, mais intelligible par l'humain, les opérations que l'ordinateur doit effectuer. Un algorithme peut toutefois aboutir à plusieurs programmes.

Le langage de programmation est destiné à l'ordinateur, doit contrairement à l'algorithme respecter une syntaxe stricte.

On appelle « langage informatique » un langage destiné à décrire l'ensemble des actions consécutives qu'un ordinateur doit exécuter. Un langage informatique est ainsi une façon pratique pour nous (humains) de donner des instructions à un ordinateur. Il est donc rigoureux : à chaque instruction correspond une action du processeur.

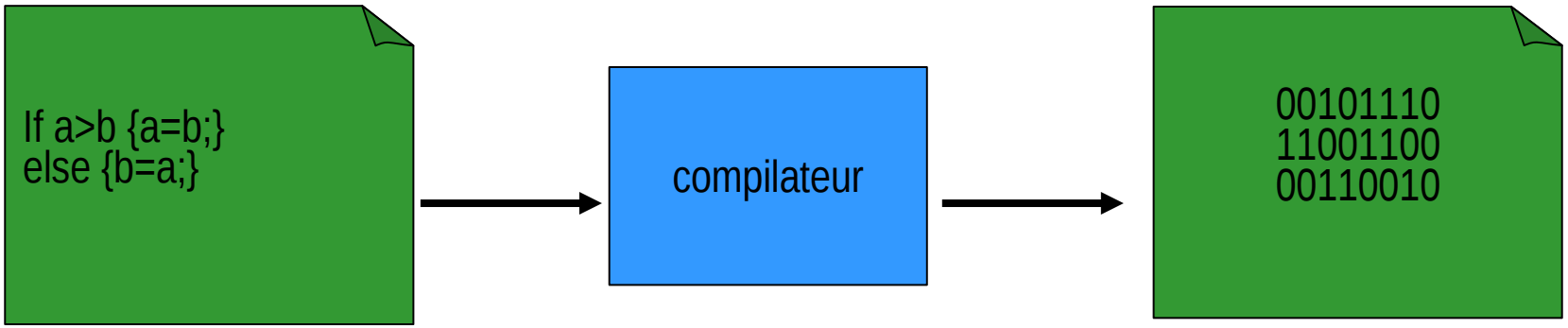
Les langages servant aux ordinateurs à communiquer entre eux n'ont rien à voir avec des langages de programmation, on parle dans ce cas de protocoles de communication (ou protocoles réseaux).

Les langages de programmation

Un programme informatique est une succession d'instructions exécutables par l'ordinateur. Toutefois, l'ordinateur ne sait manipuler que du binaire, c'est-à-dire une succession de 0 et de 1. Il est donc nécessaire d'utiliser un langage de programmation pour écrire de façon lisible, c'est-à-dire avec des instructions compréhensibles par l'humain car proches de son langage, les instructions à exécuter par l'ordinateur.

Ensuite, le code écrit dans ce type de langage est transformé en langage machine pour être exploitable par le processeur

La transformation d'un programme en langage machine est appelé la compilation. La compilation est une phase réalisée par l'ordinateur lui-même grâce à un autre programme appelé compilateur.



Les langages de programmation

La façon d'écrire un programme est intimement liée au langage de programmation que l'on a choisi car il en existe énormément. De plus, le compilateur devra correspondre au langage choisi: à chaque langage de programmation son compilateur (exception faite des langages interprétés...).

Le compilateur fait ensuite appel à un éditeur de liens (en anglais linker ou binder) qui permet d'intégrer dans le fichier final tous les éléments annexes (fonctions ou librairies) auquel le programme fait référence mais qui ne sont pas stockés dans le fichier source.

Puis il crée un fichier exécutable qui contient tout ce dont il a besoin pour fonctionner de façon autonome. Sous les systèmes d'exploitation Microsoft Windows le fichier ainsi créé possède l'extension ".exe"

Les langages de programmation

D'une façon générale, le programme est un simple fichier texte (écrit avec un traitement de texte ou un éditeur de texte), que l'on appelle fichier source .

Le fichier source contient les lignes de programmes que l'on appelle code source. Ce fichier source une fois terminé doit être compilé. La compilation se déroule en deux étapes :

Étant donné que chaque type d'ordinateur a son propre langage machine spécifique, donc pour un langage à haut niveau il faut autant de compilateurs (ou d'interpréteurs) qu'il y a de types d'ordinateurs (x86 comme Intel ou AMD, ARM, etc.) ou de systèmes d'exploitations.

Les langages de programmation

Il existe différents niveaux dans les langages de programmation:

Les langages de niveau élevé sont des langages plus facilement compris par l'usager, par exemple pour prendre le contenu d'une zone de mémoire A et le mettre dans une zone de mémoire B nous n'avons qu'à écrire: B = A

Il existe des milliers de langages de niveau élevé, pour tous goûts et toutes applications. Quelques uns des plus connus: C, C++, Pascal, Ada, Java, python
(http://en.wikipedia.org/wiki/Alphabetical_list_of_programming_languages)

- Haut niveau : proche de l'homme, vocabulaire et syntaxe plus riches

  - Python, Rebol, Ruby
  - Php, Perl, Tcl
  - C++, Java
  - Pascal
  - C, Fortran
  - Assembleur
  - Langage machine (binaire/hexadécimal)

- Bas niveau : proche de la machine, instructions élémentaires

Les langages de programmation

L'**assembleur** est le premier langage informatique qui ait été utilisé. Celui-ci est très proche du langage machine mais reste compréhensible pour des développeurs. Toutefois, un tel langage est tellement proche du langage machine qu'il dépend étroitement du type de processeur utilisé (chaque type de processeur peut avoir son propre langage machine).

Ainsi, un programme développé pour une machine ne pourra pas être porté sur un autre type de machine. Le terme « **portabilité** » désigne l'aptitude d'un programme informatique a être utilisé sur des machines de types différents. Pour pouvoir utiliser un programme informatique écrit en assembleur sur un autre type de machine, il sera parfois nécessaire de réécrire entièrement le programme !

Un langage informatique a donc plusieurs avantages :

- il est plus facilement compréhensible que le langage machine

- il permet une plus grande portabilité, c'est-à-dire une plus grande facilité d'adaptation sur des machines de types différents

Les familles de languages de programmation

On distingue habituellement quelques grandes familles de langages de programmation:

- **les langages impératifs**
- **les langages orientés procédures** (=langages fonctionnels ou procéduraux)
- **les langages orientés objets** (=langages objets)

Langages impératifs

Premiers langages utilisés en informatique. Ils sont désuètes même si quelques langages de ce type existent encore aujourd'hui (éventuellement en électronique pour programmer des puces).
Un langage impératif organise le programme sous forme d'une série d'instructions, regroupées par blocs et comprenant des sauts conditionnels permettant de revenir à un bloc d'instructions si la condition est réalisée.
Les langages impératifs structurés souffrent néanmoins d'un manque de souplesse étant donné le caractère séquentiel des instructions. Il ont été remplacés par les langages fonctionnels.

Les familles de languages de programmation

Langages fonctionnels

Un langage fonctionnel (ou langage procédural) est un langage dans lequel le programme est construit par fonctions (blocs réutilisables), retournant un nouvel état en sortie et prenant en entrée la sortie d'autres fonctions par exemple. Ce type de langage permet de diviser un problème complexe en sous-problèmes plus simples. Lorsqu'une fonction s'appelle elle-même, on parle alors de récursivité.

Langages objets

Ils incluent les éléments des langues fonctionnels, mais ajoute la notion « d'objets ».
C'est une manière de programmer qui part du principe que des choses peuvent avoir des points communs, des similarités en elles-mêmes ou en leur façon d'agir.
L'idée de la programmation objet est donc de regrouper de tels éléments afin d'en simplifier leur utilisation.
Un regroupement est appelé classe, les entités qu'il regroupe sont appelées objets.
Ainsi, au lieu de définir des actions à exécuter pour chacun, on définira des actions pour toute une classe et chaque objet pourra les effectuer (image des « poupées russes » emboitées).

Interprétation et compilation

Les langages informatiques peuvent être soit : des langages compilés, des langages interprétés

Langages compilés

Un programme écrit dans un langage dit « compilé » va être traduit une fois pour toutes par un programme annexe, appelé compilateur, afin de générer un nouveau fichier qui sera autonome, c'est-à-dire qui n'aura plus besoin d'un programme autre que lui pour s'exécuter; on dit d'ailleurs que ce fichier est exécutable.

+ La traduction étant faite une fois pour toute, il est plus rapide à l'exécution.

+ Un programme compilé a pour avantage de garantir la sécurité du code source car l'exécutable est illisible (seule le code source contient le programme en clair).

-- Mais, il est peu souple car pour le changer il faudra modifier le fichier source, le sauvegarder, le recompiler, puis le réexécuter.

-- Sans compter que une fois compilé, il sera spécifique de l'OS et du type d'ordinateur.

Interprétation et compilation

Langages interprétés

Un programme écrit dans un langage interprété a besoin d'un programme auxiliaire, l'interpréteur, pour traduire au fur et à mesure les instructions du programme : c'est une exécution ligne à ligne en temps réel (pseudo compilation / masquée pour l'utilisateur).

-- Les langages interprétés sont moins rapides que les compilés

-- Étant directement lisible (code source), permet à n'importe qui de connaître les secrets de fabrication d'un programme et donc de copier le code

++ plus souples car on n'a pas besoin recompiler si on fait des modifications

++ souvent plus faciles à apprendre

En général (mais pas toujours!) les langages compilés sont utilisés pour les logiciels commerciaux (jeux, appli, Word , etc...), les langages interprétés sont pour les logiciels open source / scientifiques libres

Exemples de langages de programmation

C# (C Sharp)

Concurrent de Java, tourne sur .Net ou compatible. Plus facile que C++. Peut partager les ressources de .Net avec d'autres langages.

C

Offre une grande liberté, mais aussi un bon exercice de déboguage à cause des pointeurs et la gestion de la mémoire. Programmation laborieuse mais programmes rapides. Surtout utilisés pour la programmation système.

C++

C'est du C plus les objets, les templates, une bibliothèque étendue, la surcharge des opérateurs. Pour la programmation système comme C mais permet de plus grand projets et la création d'applications.

Java

Equivalent du C++ ou C# mais conçu pour être portable. Applications multi-plateformes (mais plus lentes que les applications natives). Peut également être utilisé pour programmer des services Web (servlets, jsp).

Exemples de langages de programmation

JavaScript

Pour faire des pages web dynamiques coté client.

Basic

Ce langage très ancien (1964) a été amélioré par Microsoft, avec un environnement de développement complet (Visual Basic). C'est un langage qui est souvent utilisé pour faire des scripts sur les applications de Microsoft (Word, etc...)

Php

Conçu pour être inséré dans le html et construire des pages web dynamiques coté serveur, notamment à partir de bases de données. Conçu pour produire rapidement des pages html dynamiques à partir de données.

ASP

Langage de scripts coté serveur par Microsoft, avec la syntaxe de Basic. Permet de produire des pages Web dynamiques à partir de données sous serveur Windows seulement.

Exemples de langages de programmation

Assembler

Proche du langage machine et le plus rapide. Vous ne devriez par avoir à l'utiliser, comme le faisaient les anciens programmeurs. Réaliser des pilotes de matériel ou pour la programmation de machines industrielles.

Eiffel, Sather

Orienté objets, avec "programmation par contrat" et autres options de sécurité. Sather est une implémentation open-source qui est surtout utilisé par des hôpitaux.

Perl

Langage interprété de scripts. Utilisé essentiellement pour traiter les chaînes de caractères ou utilisé sous forme de CGI (pages Web dynamiques).

Tcl

Langage de script, facile à apprendre, disponible sur toutes les plateformes. Pour faire des scripts (type unix), éventuellement graphique (avec TK) et portable.

Exemples de langages de programmation

Fortran

Un des premier langages, toujours utilisé pour des tâches mathématiques. Il dispose d'une large bibliothèque mathématique.

Lisp, Prolog

Langages utilisés en intelligence artificielle qui traitent de essentiellement de listes.

Scheme

Scheme est une version modernisée de Lisp. Les fonctions récursives sont fortement utilisées

Pascal

Vieux langage (1970), amélioré avec les objets, qui impose une programmation structurée stricte. Pour l'apprentissage ou créer des applications client/serveur avec Delphi et Kylix.

Cobol

Langage de gestion, toujours utilisé sur de gros ordinateurs. Sert à maintenir de vieux programmes sur les mainframes.

Exemples de langages de programmation

SQL

C'est LE langage d'interrogation de bases de données (ce n'est pas un langage de programmation à proprement parlé)

Smalltalk

Un des premiers langages orientés objets. Ne peut fonctionner sans son environnement de développement.

Rebol

Langage interprété pour faire des scripts de réseau. Le langage lui-même peut être augmenté. Il permet de traiter de l'information sur systèmes distribués, avec des programmes compacts. Langages « original » (créé par Carl Sassenrath)

Ruby

Conception avec la simplicité en vue. Il est interprété et donc à une bibliothèque propriétaire, mais extensible.

Exemples de langages de programmation

Python

Un langage interprété moderne avec des fonctions intégrées puissantes et la simplification du code unique par l'indentation.

La version Jython permet des faire des scripts dans Java.

Scripts web possibles si couplé avec le serveur web "Zope"

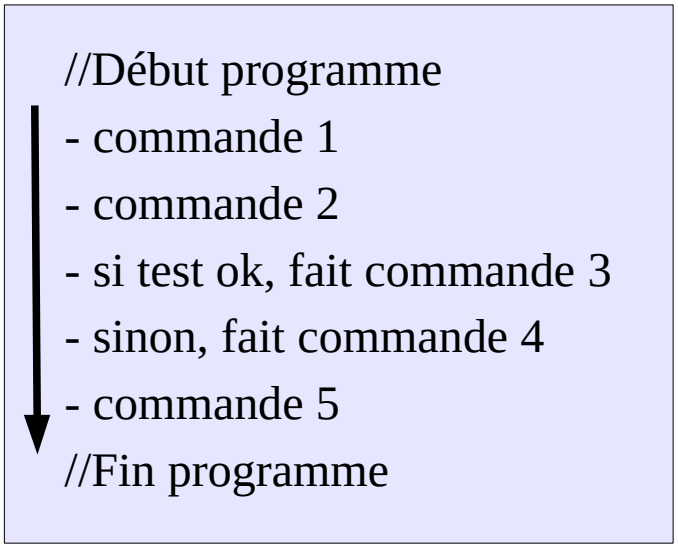
Il existe plusieurs dérivé comme "Boo" qui fournit quelques compléments. Permet de fonctionner sous ".net" ou "Mono" (biliothèques Windows de Microsoft) avec une bonne compatibilité.

Langage très utilisé en biologie

Constitution d'un programme

L'allure d'un programme dépend du type de langage utilisé pour faire le programme...

Toutefois, à peu près tous les langages de programmation sont basés sur le même principe :
Le programme est constitué d'une suite d'instructions que la machine doit exécuter. Celle-ci exécute les instructions au fur et à mesure qu'elle lit le fichier (donc de haut en bas) jusqu'à ce qu'elle rencontre une instruction de branchement qui lui indique d'aller un endroit précis du programme. Il s'agit donc d'une sorte de jeu de piste dans lequel la machine doit suivre le fil conducteur et exécuter les instructions qu'elle rencontre jusqu'à ce qu'elle arrive à la fin du programme et celui-ci s'arrête.



Une fois appris à programmer dans un langage, il est facile d'apprendre à programmer dans un autre les syntaxes sont différentes mais les principes de base des langages sont identiques.

Voir: http://www.lopr.net/informatique.php?n=8

Notions de base de la programmation

La notion de variables et les types de données

Dans la plupart des langages, on travaille avec des variables, c'est-à-dire que l'on associe à un nom un contenu.

Exemple en langage C :

```
int i  =  4 ;
```

Ici on **déclare** la variable « i » et on lui **assigne** la valeur « 4 »

"int" est le "**type**" est signifie que "i" est un entier (abréviation de "integer", entier en anglais). Elle ne pourra donc contenir que des nombres entiers (="typage en entier").

```
string s  =   « Bonjour »;
```

La variable "s" contient la chaine « Bonjour ».
Le type "string" dit que cette variable contient une chaine de caractères.

A noter que le " ;" à la fin sert à terminer la commande (=l'instruction).

Notions de base de la programmation

- Les variables sont des adresses mémoires. La déclaration consiste à réserver un espace mémoire à cette adresse. L'assignation consiste à mettre des données dans cet espace réservé.

- Le type de donnée conditionne le nombre d'octets sur laquelle la donnée est codée, c'est-à-dire l'occupation en mémoire de cette donnée ainsi que le format dans lequel elle est représentée.

- C'est la raison pour laquelle certains langages (C, Java) sont des **langages typés**, cela signifie qu'à une variable est associé non seulement un nom mais aussi un type de données qu'il faudra préciser lorsque l'on déclarera la variable, comme dans l'exemple précédent.

- Certains langages acceptent que l'on associe à un nom de variable n'importe quel type de donnée (c'est-à-dire aussi bien un nombre entier qu'un caractère). Ces langages sont dits peu ou **pas typés**, il n'y a pas besoin de déclarer le type des variables (déclaration automatique), comme le langage Python.

En python, on aurait simplement mis :

```
i = 4
s = "Bonjour"
```

(pas de typage et pas de " ;" pour terminer l'instruction)

Notions de base de la programmation

La syntaxe

Les langages demandent une syntaxe rigoureuse, spécifique de chaque langage.

- La plupart des langages sont **case sensitive** (en français "sensibles à la casse"), cela signifie qu'un nom ne comportant que des minuscules ne sera pas considéré comme équivalent au même nom comprenant des majuscules.
Ainsi la variable nommée "**T**oto" sera une variable différente de la variable "**t**oto".

- Les noms de variables admettent généralement une longueur maximale (qui dépend du langage), en général 256 caractères maximum

- Le nom des variables ne doit comporter que des caractères alpha-numériques : abcdefghijklmnopqrstuvwxyz, ABCDEFGHIJKLMNOPQRSTUVWXYZ, 1234567890 et éventuellement quelques caratères spéciaux comme "_", mais pas d'espace

Notions de base de la programmation

Mots réservés

Dans la plupart des langages, il existe quelques mots que l'on ne peut pas attribuer aux noms de variables, on les appelle mots réservés. Exemple "SYSTEM32" sous windows, ou "ENV_PATH"

Les constantes

Les constantes sont des données dont la valeur ne peut être modifiée (en fait c'est des variables invariables...). On les définit généralement en début de programme.

Les commentaires

C'est du texte inclus dans le programme qui ne sera pas pris en compte par le compilateur ou l'interpréteur.

Les commentaires servent à clarifier un programme en donnant des explications. Ils serviront aussi à une autre personne pour comprendre plus facilemnt le programme de quelqu'un d'autre, ou bien au programmeur lui-même départ si par exemple il reprends son code longtemps après l'avoir écrit. Ces lignes de textes sont généralement précédées (ou encadrées) par des caractères spéciaux.

// ceci est un commentaire en langage C, car les 2 premiers caractères sont //
ceci est un commentaire en python, car le premier caractere est un

Notions de base de la programmation

Les instructions

L'instruction est l'élément clé de l'ordinateur car c'est elle qui permet de spécifier au processeur l'action à effectuer.

Une instruction est généralement composée de deux éléments :

- l'**opérateur**: action à effectuer par le processeur

- la ou les **opérandes**: une ou plusieurs données sur lequel on va effectuer l'opération

Notions de base de la programmation

Les types d'opérateurs:

=> On distingue généralement deux ou trois types d'opérateurs :

- Les opérateurs unaires : ce sont des opérateurs qui n'admettent qu'une seule opérande
- Les opérateurs binaires : ce sont des opérateurs qui admettent deux opérandes (binaire désigne le nombre d'opérandes manipulées, l'addition, souvent notée +, est donc un opérateur binaire)
- Les opérateurs ternaires : ce sont des opérateurs qui admettent trois opérandes (les opérateurs conditionnels sont par exemple des opérateurs ternaires)

=> Les opérateurs peuvent être aussi répartis selon plusieurs catégories selon le type d'action que leur exécution déclenche :

- les opérateurs arithmétiques : +, -, *, /, ^ (puissance), / (division entière), % (modulo)
- les opérateurs de comparaison : == (égal), != (différent), >, <, .<=, >=
- les opérateurs logiques : & (et), | (ou), ! (non), xor (ou exclusif)
- les opérateurs de bits : <<, >>, >>>, <<<
- les opérateurs d'affectation : =, +=, -=, *=, %=, <<=, >>=, >>>=, &=, ^=, |=
- les opérateurs conditionnels, les opérateurs séquentiels, ...

Notions de base de la programmation

Les priorités des opérateurs:

Dans chaque langage il existe généralement des priorités d'évaluation des opérateurs, afin que l'ordinateur sache dans quel sens évaluer les opérateurs lorsque plusieurs d'entre eux sont présents dans une même expression

Comme en mathématiques, on peut aussi spécifier l'ordre des opérations en mettant des parenthèses.

*	Opérateur de multiplication
/	Opérateur de division
%	Opérateur de modulo
+	Opérateur d'addition
-	Opérateur de soustraction
<<	Opérateur de décalage à gauche
>>	Opérateur de décalage à droite
<	Opérateur d'infériorité
>	Opérateur de supériorité
<=	Opérateur d'infériorité ou d'égalité
>=	Opérateur de supériorité ou d'égalité
==	Opérateur d'égalité
!=	Opérateur d'inégalité
&	Opérateur et binaire
^	Opérateur ou exclusif binaire
|	Opérateur ou inclusif binaire
&&	Opérateur et logique
||	Opérateur ou logique
?:	Opérateur ternaire
=	Opérateur d'affectation
*=	Opérateur de multiplication et d'affectation
/=	Opérateur de division et d'affectation
%=	Opérateur de modulo et d'affectation
+=	Opérateur d'addition et d'affectation

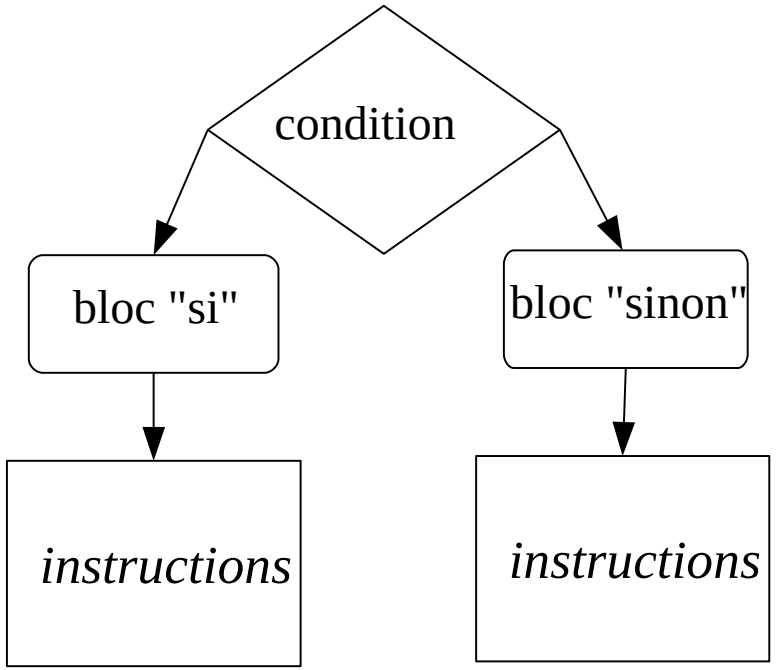
Notions de base de la programmation

Instructions de contrôle:

- Branchement conditionnel :
  - if … then… else (si … alors … sinon)

  Les instructions du bloc "si" ne sont exécutées que si la condition est vérifiée.

  Les instructions du bloc "sinon" sont exécutées seulement dans le cas contraire.
  - switch case
- Boucle conditionnelle :
  - while … do ou do … while (fait tant que …)
- Boucle inconditionnelle
  - For



Une condition est une expression booléenne. Elle est vérifiée si elle vaut vrai (true).

Opérateurs relationnels : <, <=, >, >=, =, /= ; et (and), ou (or), non (not); ou exclusif (xor); et alors (and then), ou sinon (or else); dans (in), dehors (not in)

Notions de base de la programmation

Un « bon » programme:

- Aisément lisible, donc bien commenté
- Clair, donc bien structuré
- Aisément modifiable (modulaire)
- Robuste (sans bug)
- Correct, c'est à dire retourne toujours les bons résultats
- Efficace et rapide