Licence 1

Informatique, Développement d'Application

Cours : Introduction à l'Algorithme

Séquence 1 : Introduction à l'algorithme

UVS
UNIVERSITÉ VIRTUELLE DU SÉNÉGAL

Introduction générale à l'algorithmique

I. LA NOTION D'ALGORITHME

L'algorithmique est une science très ancienne. Son nom vient d'un mathématicien arabe du IXème siècle EL KHAWRISMI. Des mathématiciens grecs comme Euclide ou Archimède en ont été les précurseurs (calcul du PGCD de 2 nombres, calcul du nombre p).

Lors du traitement de l'information le but sera évidemment de manipuler l'information de façon aussi rationnelle et efficace que possible. Ceci implique une analyse formelle du problème. Un problème concret ne peut être résolu par un ordinateur que si les opérations nécessaires à cette résolution peuvent être décomposées en un nombre fini d'étapes élémentaires (l'addition et la comparaison de deux nombres en sont des exemples typiques) dont chacune peut être traitée individuellement. En d'autres termes, on doit indiquer à l'ordinateur de façon précise et détaillée comment un problème donné doit être résolu. Un tel procédé de résolution est appelé ***algorithme.***

Un algorithme est une suite de règles, de raisonnements ou d'opérations, qui transforment des grandeurs données (données d'entrée)[angl. input] en d'autres grandeurs (données de sortie)[angl. output].

II. EXEMPLES D'ALGORITHMES ELEMENTAIRES

Les exemples suivants montrent que beaucoup d'algorithmes décrivent des tâches quotidiennes. Pour la présentation de ces algorithmes nous utiliserons tout simplement des instructions rédigées en langue française. Ces instructions seront suffisamment claires pour permettre la compréhension de leur intention.

Exemple 1: Prise d'un médicament contre la toux

Algorithme:

"En cas de toux, prendre, sauf avis contraire du médecin, un comprimé toutes les 4 heures, jusqu'à disparition des symptômes. Pour les enfants, un comprimé toutes les 8 heures suffit."

Exemple 2: Tri d'un jeu de cartes suivant la couleur

Algorithme:

(1) Prendre la première carte;
(2) La carte est-elle rouge?
Si oui, poser la carte sur le premier tas;
Sinon, poser la carte sur le second tas;
(3) Reste-t-il des cartes?
Si oui, prendre la carte suivante et continuer sous 2; Sinon, fin du tri.

Introduction à l'algorithmique

III. Propriétés d'un algorithme

Un algorithme doit être:

PRECIS: Il doit indiquer:

- l'ordre des étapes qui le constituent
- à quel moment il faut cesser une action
- à quel moment il faut en commencer une autre
- comment choisir entre différentes possibilités

DETERMINISTE

- Une suite d'exécutions à partir des mêmes données doit produire des résultats identiques.

FINI DANS LE TEMPS

- c'est-à-dire s'arrêter au bout d'un temps fini.

Exemple : *Résolution de l'équation du premier degré* A X + B = 0

```
1. Lire les coefficients A et B
2.    Si A est non nul alors
        affecter à X la valeur - B / A
        afficher à l'écran la valeur de X
      sinon
        si B est nul
          alors
            écrire "tout réel est solution"
          sinon
            écrire "pas de solution"
```

IV. Quelques règles pour concevoir un algorithme

1) Entrée

L'énoncé d'un algorithme doit comprendre une définition des données auxquelles il s'applique. Normalement, un algorithme possède une ou plusieurs données d'entrée [angl. input data], c'est-à-dire des valeurs qui sont connues avant son exécution et sur lesquelles l'algorithme est appliqué. Comme un algorithme doit tenir compte des propriétés des données d'entrée,

Introduction à l'algorithmique

l'ensemble de ces données doit être spécifié de façon exacte. Evidemment cet ensemble peut être l'ensemble vide.

2) Sortie

Un algorithme possède une ou plusieurs données de sortie [angl. output data], c'est-à-dire des valeurs produites par lui-même. Ces données sont en relation exactement spécifiée avec les données d'entrée.

3) Terminaison

Un algorithme doit se terminer après un nombre fini de pas.

4) Clarté

Le concepteur d'un algorithme doit vérifier attentivement que l'algorithme est fixé dans tous ses détails, c'est-à-dire que l'algorithme décrit précisément la procédure qu'il doit suivre, que tous les cas ont été prévus et, en particulier, que l'ordre dans lequel les règles devront être appliquées n'est pas ambigu.
Chaque opération doit donc être définie d'une manière précise et claire.
Cette tâche est connue sous le nom d'***affinement progressif*** de l'algorithme ou ***conception descendante***.
Exemple: Considérons l'exemple, pris de la vie courante, qui est celui de la personne qui indique à un étranger l'itinéraire pour aller dans une certaine rue. L'algorithme qu'elle donne est par exemple le suivant: "Tourner à droite au magasin, allez jusqu'au prochain croisement, puis prenez la troisième rue à gauche . . . ". Cette description est presque parfaite à l'exception de quelques détails près (à quel magasin tourner, par exemple) et risque fort d'entraîner l'étranger vers une toute autre rue. Or, l'étranger a la possibilité de demander plus de renseignements à la personne. Malheureusement les ordinateurs n'ont pas encore cette ressource.

5) Efficacité

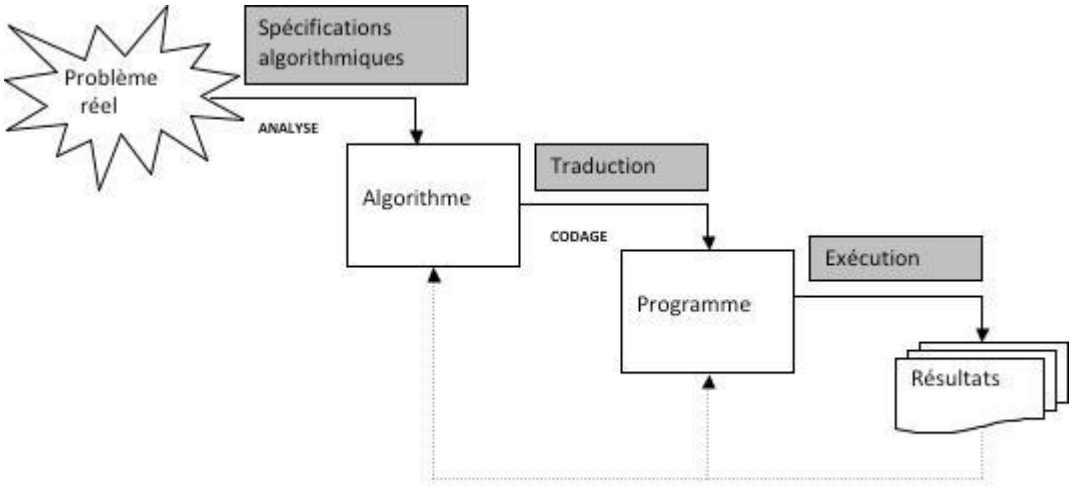
Pour des raisons pratiques, un algorithme doit pouvoir être exécuté de manière efficace.
On dit qu'un algorithme doit être efficace ce qui signifie que:
• chaque opération doit être suffisamment simple, afin qu'elle soit exécutable en un temps fini,
• le nombre d'opérations doit être fini et limité de façon à ce que la machine soit capable de terminer leur exécution dans un temps raisonnable.

V. Place de l'algorithme dans la résolution d'un problème informatique

Un algorithme doit être exprimé dans un langage de programmation pour être compris et exécuté par un ordinateur§.Le programme constitue le codage d'un algorithme dans un langage de programmation donné, et qui peut être traité par une machine donnée.

Introduction à l'algorithmique

L'écriture d'un programme n'est qu'une étape dans le processus de programmation, comme le montre le schéma suivant :



Les différentes étapes du processus de programmation

L'analyse du problème consiste à définir les différentes étapes de sa résolution. Elle permet de définir le contenu d'un programme en terme de données et d'actions. Le problème initial est décomposé en sous problèmes plus simples à résoudre auxquels on associe une spécification formelle ayant des conditions d'entrées et le(s) résultat(s) que l'on souhaiterait obtenir. L'ensemble de ces spécifications représente l'algorithme.

La phase (**Codage**) suivante consiste à traduire l'algorithme dans un langage de programmation donné tout en respectant strictement la syntaxe du langage.

Lors de l'étape d'**exécution**, soit des erreurs syntaxiques sont signalées, ce qui entraîne des corrections en général simples à effectuer, soit des erreurs sémantiques plus difficiles à déceler. Dans le cas d'erreurs syntaxiques les retours vers le programme peuvent être fréquents. Dans le cas d'erreurs sémantiques, le programme produit des résultats qui ne correspondent pas à ceux escomptés : les retours vers l'analyse (l'algorithme) sont alors inévitables.

VI. Les Différentes phases d'élaboration d'un algorithme

On peut distinguer quatre phases principales dans l'élaboration d'un algorithme

- ð Analyse du problème
- ð Expression d'une solution en langage courant
- ð Expression d'une solution en pseudo-langage
- ð Tests et Vérification de l'adéquation de la solution

Analyse du problème:

Introduction à l'algorithmique

L'analyse consiste à bien comprendre l'énoncé du problème: Il est inutile et dangereux de passer à la phase suivante si vous n'avez pas bien discerné le problème.

Expression du raisonnement

Bien souvent, quelques lignes écrites en langage courant suffisent pour décrire succinctement l'essentiel du problème. L'intérêt de cette étape est qu'elle permet de vérifier rapidement que l'on se trouve sur la bonne voie. De plus, ces quelques lignes seront un support efficace lors de l'écriture de l'algorithme.

Expression d'une solution en pseudo-langage

Il peut arriver que plusieurs solutions répondent à un problème donné. Il faudra choisir la solution la plus judicieuse et rester cohérent jusqu'au bout.

Tests et Vérification de l'adéquation de la solution

Vérifier l'exactitude du comportement de l'algorithme, son bon déroulement. Si l'algorithme ne répond pas parfaitement à toutes les requêtes exprimées dans l'énoncé du problème, retournez à la phase n°1.

VII. Notion de Type d'un objet

En informatique, les objets manipulés par un algorithme doivent appartenir à un type connu au moment de leur utilisation. Tout objet peut être caractérisé par un type qui indique :

- les ensembles de valeurs que peut prendre l'objet.
- les actions autorisées sur cet objet.

Les différents types simples sont les suivants :

- **Type entier** :

prend ses valeurs dans un sous-ensemble des entiers relatifs. C'est un ensemble fini dans lequel chaque élément possède un successeur et un prédécesseur.

- **Type réel** :

prend ses valeurs dans un sous-ensemble de réels décimaux signés. Dans la plupart des langages, cet ensemble n'est pas un ensemble fini. On ne peut trouver de successeur ou de prédécesseur à un réel donné.

- **Type caractère** :

prend ses valeurs dans l'ensemble des caractères de la table ASCII.

- **Type chaîne de caractère** :

se compose d'une suite de symboles de type caractère

- **Type booléen :**

type logique qui peut prendre les valeurs VRAI ou FAUX.

Introduction à l'algorithmique

VIII. Les objets

Pour désigner ces différents objets, on utilisera des chaînes de caractères qui seront appelées les identificateurs des objets. Pour différentier un identificateur d'un nombre, un identificateur commence par une lettre et ne comporte pas d'espace. De plus, on essaiera toujours de choisir des noms explicites afin de faciliter la relecture et l'éventuelle maintenance de l'algorithme par un tiers.

Un objet peut être :

- une constante ou une variable s'il s'agit d'une donnée
- au sens large, une fonction (ou une procédure) s'il s'agit d'un algorithme

1. Les variables

Dans un programme informatique, on va avoir en permanence besoin de stocker provisoirement des valeurs : données issues du disque dur, fournies par l'utilisateur, résultats intermédiaires ou définitifs obtenus par le programme, etc. Ces données peuvent être de plusieurs types : nombres, texte, etc. On utilise toujours une variable pour stocker une information au cours d'un programme (dans un langage donné).
Dans l'ordinateur, physiquement, il y a un emplacement de mémoire, repéré par une adresse binaire réservé à chaque variable déclarée. Exemple : un octet pour un nombre - Déclaration :
La première chose à faire avant de pouvoir utiliser une variable est de lui donner un nom et de créer l'emplacement mémoire qui lui correspond, ce mécanisme s'appelle la déclaration de la variable.
Syntaxe :

```
Variable
        nomVariable: Type
```

```
Exemple
Variable
          x:Entier
          v1,Y: Réel
           c: char
           S : chaine
           Ok: Booléen
```

2. Actions élémentaires

Les actions élémentaires sur une donnée dépendent évidemment du type de cette donnée et de sa catégorie (variable ou constante).

Opérateurs sur les types simples

Introduction à l'algorithmique

Opérateur	Notation	Type des opérandes	Type du résultat
Multiplication Division entière Division reste(modulo)	* DIV / MOD	entier ou réel entier entier ou réel entier	entier ou réel entier réel entier
Addition Soustraction	+ -	entier ou réel	entier ou réel
Comparaison	< <= > >= = <>	tout type	booléen
et logique ou logique	ET OU	booléen	booléen

Affectation

L'affectation a pour rôle d'attribuer une valeur, résultat d'une évaluation, à un objet. La valeur doit être compatible avec le type de la valeur à gauche de l'affectation. Le symbole utilisé pour l'affectation est ←

Exemple

X <-- 1 (X prend la valeur 1)

X <-- 2*3+5 (X prend la valeur du résultat de l'opération 2*3+5)

D <-- D+1 (D augmente de 1)

prix_total <- nb_kg * prix_du_kg

(Si nb_kg est de type entier et prix_du_kg est de type réel alors prix_total doit être de type réel)

2. Constante

Une constante est un objet dont la valeur n'évolue pas tout au long de l'algorithme.

Syntaxe de la déclaration

Introduction à l'algorithmique

Constante

NomConstante= valeur

Exemple
Constante
PI=3,14

Expressions :
Une expression est un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur (typée) :
Exemples d'expressions :
12+5 ;
A+2 ;

IX. Lecture et écriture

Il existe des d'instructions pour permettre à la machine de dialoguer avec l'utilisateur : les instructions d'entrées/sorties ou tout simplement de lecture/écriture.

Dans un sens, ces instructions permettent à l'utilisateur de rentrer des valeurs au clavier pour qu'elles soient utilisées par l'algorithme. Cette opération est la lecture.

Dans l'autre sens, d'autres instructions permettent au programme de communiquer des valeurs à l'utilisateur en les affichant à l'écran. Cette opération est l'écriture.

Syntaxes

Lire(nomVariable)

Lire(nomVariable1, nomVariable2 …)

Ecrire(nomVariable1, nomVariable1 …)

Ecrire("chaine constante")

Ecrire ("Chaine constante",prix_total)

Avant de Lire une variable, il est très fortement conseillé d'écrire des libellés à l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper
Exemple
Ecrire("saisir une valeur")

Lire(nomVariable)

Remarque:

Afin d'améliorer la lisibilité d'un algorithme, on peut utiliser des commentaires. Un commentaire est une suite de caractères quelconques encadrée par les symboles (% et %). Cette suite de caractère ne sera pas exécutée. Elle sert seulement à documenter le programme pour le rendre lisible par un tiers.

Exemple

(%ceci est un commentaire %)

X. Structure générale d'un algorithme

Algorithme : NomAlgorithme

Constante %déclaration des constantes
Type %création des types%
Variable %Déclaration des variables%
%définition des sous programmes%

Début

Instructions

Fin

Exemple

```
ALGORITHME facture
VARIABLE
        prix_total , prix_du_kg : REEL
        nb_kg : ENTIER
DEBUT
        Ecrire(« Entrez le prix d'un kilogramme de choux»)
        Lire(prix_du_kg)
        Ecrire (« Entrez le nombre de kilogramme de choux : »)
        Lire (nb_kg)
        prix_total <- prix_du_kg * nb_kg
        Ecrire (« Le prix total de l''achat est : »,prix_total)
FIN .
```