Algorithmes et structures de données

Objectifs du cours

Adopter une démarche logique de résolution de problèmes pour la mise en œuvre d'algorithmes.

Connaître les capacités de l'ordinateur en terme d'actions élémentaires qu'il peut assurer et la logique d'exécution des instructions.

Faire des choix argumentés sur l'utilisation des principales structures de données.

Plan

1. Introduction
2. Les itérations
3. Les types scalaires
4. Les séquences
5. Les procédures et les fonctions
6. Les chaînes de caractères
7. Les enregistrements
8. Les fichiers
9. La récursivité
10. Notion de complexité
11. Les tris récursifs
12. Les pointeurs
13. Les listes
14. Les piles
15. Les files
16. Les arbres binaires

Bibliographie

- Structures de données et algorithmes
  A. Aho, J. Hopcroft et J. Ullman – Inter Editions - 1987.

- Types de données et algorithmes
  M-C. Gaudel, M. Soria et C. Froidevaux - vol 1 - INRIA.

- Eléments d'algorithmique
  D. Beauquier, J. Berstel et P. Chrétienne - Masson - 1992.

- Programmation - cours et exercices
  G. Chaty et V. Vicard - Ellipse - 1992.

1. Introduction

- Ecrire un programme informatique pour un problème, c'est :
  - Formuler le problème,
  - Spécifier les données,
  - Construire une solution,
  - Mettre en oeuvre l'algorithme,
  - Tester le programme et le documenter,
  - Evaluer la complexité de la solution.

- Connaître le problème à traiter, c'est l'avoir résolu à moitié.
- La plupart des problèmes n'ont pas de spécification simple, ni précise.
- Certains problèmes ne sont pas formulables en termes acceptables pour une solution informatique.

Algorithme, type abstrait de données et structure de données

- **Algorithme** : série d'instructions dont chacune peut-être réalisée en un nombre fini d'étapes et en un temps fini.

  → calculabilité, terminaison et preuve de programme

- **Type abstrait de données** : ensemble organisé d'objets et des opérations de manipulation sur cet ensemble (moyens d'accès aux éléments de l'ensemble et possibilités de le modifier).

- **Structure de données** : implémentation en machine d'un ensemble organisé d'objets avec la réalisation des opérations d'accès, de construction et de modification.

Démarche descendante (1/2)

- De l'analyse (en langage naturel) jusqu'à l'algorithme (en pseudo-langage), indépendamment du langage de programmation

- Au départ, données abstraites : notation + opérations et leurs propriétés

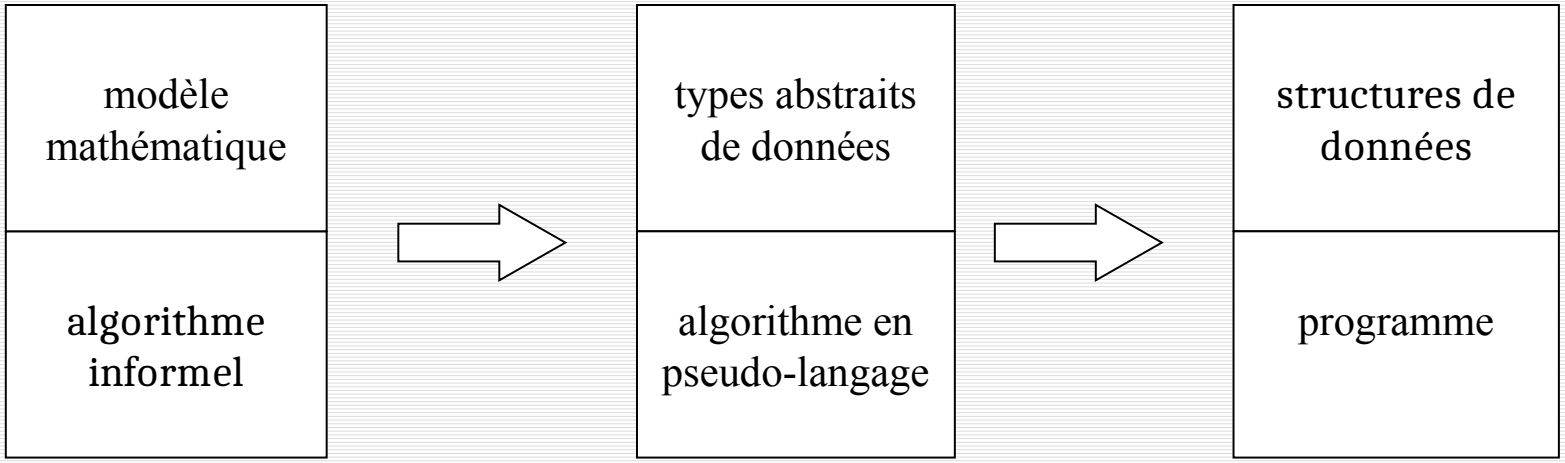
  → types abstraits de données.

- Ensuite, représentation concrète des types et des opérations pour obtenir un programme exécutable

  → structures de données.

  Différentes représentations d'un type abstrait pour différentes versions de l'algorithme.

Démarche descendante (2/2)



Programme : suite d'instructions résolvant le problème donné avec les déclarations des objets manipulés (constantes, types, variables, procédures et fonctions).

Type abstrait de données (TAD)

- Définition

  Signature : syntaxe du type → nom des opérations, type de leurs arguments

  - sorte : nom du type
  - utilise : types utilisés
  - opérations : listes des opérations avec le domaine des paramètres

  Propriétés des opérations : axiomes donnant la sémantique des opérations (sous forme logique) pour toute implémentation du type abstrait

  - N'y a-t-il pas d'axiomes contradictoires ? (consistance)
  - A-t-on donné un nb suffisant d'axiomes pour décrire toutes les propriétés du type abstrait (complétude)
  - Opération interne (constructeur) : résultat du type abstrait (modification physique)
  - Observateur : un argument du type abstrait et résultat d'un autre type (pas de modification physique)
  - Pour éviter les problèmes de complétude, il faut écrire les axiomes qui définissent le résultat de la composition des observateurs avec les opérations internes.

- Importation de types

  - Lorsqu'un type de données est implémenté, on peut se servir de ses opérations comme opérations élémentaires dans la définition de types de données plus complexes.
  - Hiérarchie de types, avec en bas les types élémentaires (entier, réel, booléen, ...).

Exemple : coordonnées d'un point

Sorte : Coord

Utilise : Entier, Booléen

Opérations :

Créer(Entier, Entier) → Coord *[opération interne]*

X(Coord) → Entier

Y(Coord) → Entier

Eq(Coord, Coord) → Booléen

Axiomes :

X(Créer(x,y)) = x

Y(Créer(x,y)) = y

Eq(Créer(x1,y1), Créer(x2,y2)) = ((x1=x2) et (y1=y2))

Schéma d'un programme

```
Programme bidule

Const constantes {Déclaration des constantes}
Type types {Déclaration des types}
Var variables {Déclarations des variables globales}
Procédures et Fonctions

Début
Programme principal
Fin
```

Instruction (1/3)

- **Instruction** : ordre donné à la machine.
- **Exécution** : action effectuée par la machine en réponse à une instruction.

  Il faut distinguer les objets (variables) manipulés par l'algorithme et les actions effectuées par la machine (instructions).

- Affectation

  Action élémentaire de donner une valeur (contenu) à une variable (contenant)

  Symbolisée par ← : ranger une valeur en mémoire.

  x←4 met la valeur 4 dans la case identifiée par x

  La valeur initiale de x est « écrasée » par la nouvelle valeur.

Exemple : échange de 2 variables

Soit 2 variables entières X et Y ayant respectivement les valeurs x et y ; quelles sont les affectations qui donneront à X la valeur y et à Y la valeur x ?

Il faut utiliser une variable intermédiaire Z.

```
Z←X
X←Y
Y←Z
```

Instruction (2/3)

- Lecture et écriture

  La machine doit connaître la valeur d'une variable.

  Si cette valeur n'a pas été initialisée ou calculée, il faut que l'utilisateur la fournisse.

  → instruction lire

  lire(x) mettre dans la case x la valeur présente sur le port d'entrée de la machine

  De la même façon, la machine doit pouvoir fournir un résultat.

  → instruction écrire

  écrire(x) mettre sur le port de sortie de la machine le contenu de la case x

Exemple : échange de 2 variables

```
Programme échange
Var x,y,z : entier
Début
    lire(x)
    lire(y)
    Z←X
    X←Y
    Y←Z
    écrire(x)
    écrire(y)
Fin
```

Instruction (3/3)

- Alternatives : pour choisir les exécutions en fonction des valeurs des données.

  - Si alors

```
Si cond
   Alors inst
FinSi
```

  - Si alors sinon

```
Si cond
   Alors inst1
   Sinon inst2
FinSi
```

  - Selon

pour faire un choix parmi n possibles (n>2)

```
Selon exp
   val1 : inst1
   val2 : inst2
   ...
   valn : instn
FinSelon.
```

Exemple : $ax^2 + bx + c = 0$

Rappel :

$\Delta = b^2 - 4ac$

Si $\Delta \geq 0$

$X_1 = (-b+\text{sqrt}(\Delta)) / 2a$

$X_2 = (-b-\text{sqrt}(\Delta)) / 2a$

Exemple : $ax^2 + bx + c = 0$

```
Programme équation
Var a, b, c : réel
Début
  lire(a,b,c)
  Si a=0
     Alors Si b=0
             Alors Si c=0
                   Alors écrire('Tout réel est solution')
                   Sinon écrire('Pas de solution')
              FinSi
             Sinon écrire(–c/b)
           FinSi
    Sinon Si (b*b-4*a*c)>=0
             Alors écrire('2 solutions : ',(-b+sqrt(b*b-4*a*c))/(2*a),
                      ' et ', (-b-sqrt(b*b-4*a*c))/(2*a))
             Sinon écrire('Pas de solution')
          FinSi
  FinSi  Fin
```