Algorithmes et Programmes

Algorithmes et Programmes

- Vie d'un programme
- Algorithme
- Programmation : le langage
- Exécution et test des programmes

Cycle de vie d'un programme (d'un logiciel)

- Conception - Modélisation
  - Analyse du problème
  - Solution algorithmique
    - langage d'algorithmes
- Programmation
  - Programme
    - langage de « haut niveau »
- Compilation – Interprétation
  - Exécution sur machine
    - langage machine de « bas niveau »
    - assembleur et code machine
- Mise au point
  - Vérification par test pour corriger
  - Evaluation du coût pour optimiser

Cycle de vie d'un programme (d'un logiciel)

- Conception - Modélisation
  - Langage de description d'algorithme
    - simplicité , précision
    - indépendant de la programmation et de la machine
    - Exemple : diagramme , pseudo C, …
- Programmation
- Exécution

Cycle de vie d'un programme (d'un logiciel)

- Conception - Modélisation
- Programmation
  - Langage de programmation (langages « évolués »)
    - syntaxe contraignante, différents styles d'abstraction
    - indépendant de la machine
  - Types de langages
    - impératifs : Fortran, Cobol, Pascal, C
    - fonctionnels : Lisp, ML, Caml
    - logiques : Prolog
    - objets : C++, Java
- Exécution

Cycle de vie d'un programme (d'un logiciel)

- Conception - Modélisation
- Programmation
- Exécution
  - Langage assembleur
    - dépendant de la machine, du processeur
    - Exemples : Assembleur pour PC (IA-32), PowerPC, MIPS, SPARC, etc.

L'entier N est-il pair ?

- Conception - Modélisation
  - Analyse du problème
    - Un nombre N est pair si le reste de la division de N par 2 est nul
  - Solution algorithmique
    - 1. calculer le reste R de la division de N par 2
    - 2. si R est égal à 0 alors N est pair
    - 3. sinon N n'est pas pair

Algorithme

- Un algorithme n'est pas forcément destiné à décrire la solution d'un problème pour la programmation et l'informatique ...
  - Un algorithme en cuisine s'appelle une recette
  - Un algorithme en musique s'appelle une partition
  - Un algorithme en tissage s'appelle un point

Peu de mécanismes de base

- Faire A ; Faire B ; Faire C … en séquence
- a←10 affectation
- + - * / operations de math
- {Faire A ; Faire B };{Faire C ; Faire D} groupés
- Si (…) Alors {…} Sinon
- Tant que (…) Faire {…}
- Pour i allant de 0 jusqu'à 100 faire {…i…}
- f(a, b, c) Fonctions (appel et déclaration)

Comment est-ce possible que l'informatique tienne en si peu de mécanismes de base ?

Algorithme (historique)

- Le mot algorithme vient du nom d'un mathématicien perse : Muhammad ibn Musa al-Khowârizmî.
- La signification du mot évolue au cours des temps :
  - pratique de l'algèbre (d'Alembert, XVIIIe siècle)
  - méthode et notation de toute espèce de calcul
  - tout procédé de calcul systématique, voire automatique

Algorithme de la mousse au chocolat

- Ingrédients :
  - 250g de chocolat, 125g de beurre, 6 œufs, 50 g de sucre, café
- Etapes :
  - Si chocolat a dessert
    - faire fondre le chocolat avec 2 cuillères d'eau
  - Sinon
    - Faire tièdir le chocolat liquide au micro-onde
  - Ajouter le beurre, laisser refroidir puis ajouter les jaunes
  - Ajouter le sucre et comme parfum un peu de café
  - Battre les blancs jusqu'à former une neige uniforme
  - Ajouter au mélange.
- A partir des ingrédients (données en entrée), appliquer la recette (les étapes) va produire une mousse au chocolat (le résultat).

Algorithme : un peu de méthodologie

- identifier les données fournies / nécessaires (données en entrée)
- identifier le résultat (données en sortie)
- déterminer les actions ou opérations élémentaires
- spécifier l'enchaînement des actions

- langage d'algorithmes = langage de description des données, des actions et des enchaînements

Langage de description d'algorithmes

```
Algorithme   titre
      %  commentaire
      Lexique :       variables   // entrée
                  :       variables   // sortie
                :      variables // auxiliaire
       actions :       noms des opérations
début
      liste d'instructions
fin
```

Calculer les intérêts d'un prêt bancaire

- Analyse
  - ValF = (ValIni * (1+interet/100))* (1+interet/100)… 30 fois
  - Interet = 4% si valeur<10000 et 5% si >=10000
- Algorithme InteretsBanquairesVariables

```
%Calcul des interets qnnee qpres qnnee
Lexique : ValIni   entier  // Entrée
          ValF     entier  //Auxiliaire
Action : +, *, /, lire ,ecrire
Début
        Lire ValIni        //Demander ValIni a l'utilisateur
        Faire 30 fois :
            Si ValF<10000 Alors
               ValF ← ValF *1.04
            Sinon
               ValF ← ValF *1.05
        Ecrire "a la fin des 30 ans vous avez : ", ValF, " euros"
Fin
```

Commentaires

Variable (ICI)

- Une variable est le nom d'un «récipient» destiné à contenir une valeur. Lorsque nous nous intéresserons un peu plus à l'ordinateur, le récipient sera une «zone» mémoire.
- Le type d'une variable sert à préciser la nature des valeurs acceptables par le récipient. Un type est un nom pour un ensemble de valeurs.
  - Exemple : A est une variable de type entier. La valeur de (dans) A est un entier. La valeur de Carré ne peut être un caractère (‘a’, ‘b’, ‘c’…) ou un réel (2008,3)

Affectation par une valeur

- L'affectation **variable** ← **valeur** est une instruction qui permet de changer la valeur d'une variable. L'affectation modifie le contenu du récipient désigné par la variable.
  - La valeur de la variable à gauche de ← est remplacée par la valeur à droite de ←.
  - Exemple : Carré ← 0 « se lit » le récipient Carré reçoit la valeur 0.
- Avertissement
  - L'affectation est une instruction qui est dite «destructrice».
  - L'ancienne valeur de la variable est détruite, écrasée, effacée par la nouvelle valeur !
  - Carré ← N Copie de la valeur de N. La valeur de N (par exemple 7) existe en double

Affectation par une expression

- L'affectation variable ← expression est effectuée par :
  - 1. évaluation de l'expression
  - 2. placement du résultat dans le récipient désigné par la variable à gauche.
- Attention
  - A droite de ←, dans l'expression, les variables sont abusivement utilisées pour désigner les valeurs qu'elles contiennent. **Ceci est une convention.**
  - Exemple : Carré ← Carré + N a pour effet de mettre le résultat de la somme de la **valeur** de Carré avec la **valeur** de N dans le **récipient** Carré.
  - La valeur de Carré évolue dans le temps
    - Contrairement en math bien souvent
    - L' évolution n'est JAMAIS continu

Instruction conditionnelle

- **Si** « condition » **alors** faire liste d'instructions
  sinon faire liste d'instructions

  FINSI

  - Exemple : l'instruction 5 de l'algorithme Test-Carré-Parfait est une conditionnelle.
- Condition est une expression booléenne
  - Exemple : Reprenons l'exécution de Test-Carré-Parfait pour N=7.
    - La première évaluation de la condition $J \geq 7$ produit la valeur booléenne «faux» donc les instructions 2. et 3. sont exécutées.

Algèbre de Boole et Logique

- Utiliser **faux** et **vrai** (ou F et V) à la place de 0 et 1
- Renommer l'addition, la multiplication et la complémentation par **ou**, **et** et **non** respectivement appelée disjonction, conjonction et négation.

- Attention au « OU » ≠ fromage ou dessert

x	y	**ou**	**et**	**Non (x)**
F	F	F	F	V
F	V	V	F	V
V	F	V	F	F
V	V	V	V	F

Condition et Expression booléenne

- Expression booléenne élémentaire par l'exemple
  - (J < 7 et J > 4) est une expression booléenne.
    - C'est la conjonction de deux expressions booléennes élémentaires.
    - Elle est évaluée à vraie si la valeur de la variable J appartient à $]4,7[$.
  - Considérons les variables cv pour la couleur de ma voiture, mv pour la marque et div pour l'immatriculation (département).
    - Que signifie l'expression ci-dessous ?

(cv = blanc et mv = peugeot) ou

((cv = noir) et (div=75 ou div=92 ou div = 93 ou div = 94))

Condition et Expression booléenne

- Expression booléenne élémentaire par l'exemple
  - (J < 7 et J > 4) est une expression booléenne.
    - C'est la conjonction de deux expressions booléennes élémentaires.
    - Elle est évaluée à vraie si la valeur de la variable J appartient à $]4,7[$.
  - Considérons les variables cv pour la couleur de ma voiture, mv pour la marque et div pour l'immatriculation (département).
    - Que signifie l'expression ci-dessous ?

(cv = blanc et mv = peugeot) ou

((cv = noir) et (div=75 ou div=92 ou div = 93 ou div = 94))

Algorithme: Suite finie d'instructions vérifiant:

- Chaque étape est décrite de façon précise;
- Chaque étape est déterministe: produit des résultats uniques;
- L'algorithme s'arrête après un nb fini d'instructions
- Reçoit des données en entrée;
- Produit des données en sortie;
- Généralité: Applicable à des ensembles différents de données d'entrée

Différence entre problème et instance du problème

- Exemple d'un problème: Tri d'un ensemble d'éléments
  - Entrée: Suite de *n* éléts $a_1,...a_n$
  - Sortie: La suite réordonnée

- Instances du problème:
  - Suite de nombres: *475, 787, 34, 245, 56, 350*
  - Suite de noms: *Pierre, Christine, Sylvie, Samuel, Fabien*

Exemple d'un algorithme

1. *x := a;*
2. **Si** *b>x*, **alors** *x := b;*
3. **Si** *c>x*, **alors** *x := c;*

:= Opérateur d'assignation

y := z signifie ``copie la valeur de z dans y''.

Valeur de z inchangée

Paramètres d'entrée: a, b, c

Valeur de sortie: x = max (a,b,c)

Pseudo-Code

Algorithme maximum: Retourne le maximum de 3 entiers

- Entrée: Trois entiers a, b, c;
- Sortie: x contenant la valeur maximale parmi a, b, c

1. **Procédure** *max(a,b,c)*
2. *x := a;*
3. **Si** *b>x* **alors** // Si b plus grand que x, mettre x à jour
4. *x := b;*
5. **Si** *c>x* **alors** // Si c plus grand que x, mettre x à jour
6. *x := c;*
7. **Retourner** (x)
8. **Fin** max

Pseudo-Code: Notation proche du code des langages de programmation (C, Pascal). Notation standard mais pas rigoureuse

- Titre de l'algorithme, description brève, entrées, sorties
- Procédures consécutives
- Numéroter les lignes (facultatif)
- Procédure commence par le mot ``**Procédure"**, suivit du nom, suivit des paramètres
- Instructions (lignes exécutables) entre ``**Procédure"** et ``**Fin",** exécutées l'une après l'autre
- Bloc d'instructions entre ``**Début**" et ``**Fin**"
- Ligne de commentaires signalée par // (ou /* */)
- ``**Retourne** (x)" termine une procédure et retourne la valeur de x

Instruction Si-Alors-Sinon (If-Then-Else)

Si *p* **alors**

action

Si condition *p* vérifiée, exécuter *action*.

Si *p* **alors**

action 1;

Sinon

action 2;

Si condition *p* vérifiée, exécuter *action 1*. Sinon, exécuter *action 2.*

Si $x \geq 0$ **alors**

Début

x := x-1;

a := b+c;

Fin

Bloc de conditions entre **Début** et **Fin**.

Instruction Tant que

Tant que *p* **Faire**
    action;

Tant que *p* est vrai exécuter *action*

Algorithme Plus-Grand-Element: Retourne la plus grande valeur d'une liste

Entrée: *n* entiers $S_1,\ldots, S_n$

Sortie: *grand* contenant le plus grand élément

Procédure *plus-grand (S,n)*
grand := S_1;
i:=2;
Tant que *i* ≤ *n* **Faire**
    Début
    Si S_i > *grand* **alors** // une plus grande valeur a été trouvée
        grand := S_i;
    i := *i*+1;
    Fin
Retourne *(grand)*
Fin *plus-grand;*

Instruction Pour (For)

Pour *var := init* **à** *limite* **Faire**

action;

À chaque passage dans la boucle, *var* est incrémenté de 1. On s'arrête dès que *var* > *limite*

Algorithme Plus-Grand-Element: Réécriture de l'algorithme précédent mais avec une boucle ``Pour''

Entrée: *n* entiers S1,…, Sn

Sortie: *grand* contenant le plus grand élément

Procédure *plus-grand (S,n)*
grand := S_1;
Pour *i* =1 **à** *n* **Faire**
  Si S_i > *grand* **alors** // une plus grande valeur a été trouvée
    grand := S_i;
Retourne *(grand)*
Fin *plus-grand;*

Conclusions

- L'algorithmique repose sur peu de fondements
  - Les combinaisons sont infinis
  - L'analyse du problème est primordiale
  - Les types de données multiplient les possibilités de façon exponentielle !
- L'algorithmique peut rarement être traitée sous l'angle du formalisme
  - Bug légers, bug sévères, Optimisations
  - L'analyse laisse souvent la place à des imprécisions
  - Les besoins évoluent souvent en même temps que l'écriture

FIN