Conventions algorithmiques

29 septembre 2000

1 Démarche

Le développement de programmes comportent les trois phases suivantes à respecter dans vos travaux (en particulier dans les compte-rendus de TP) :

Analyse Il s'agit d'analyser le problème :
- identifier les informations manipulées, les nommer et décrire leur nature
- recenser les méthodes de résolution du problème et formuler leur principe (principales étapes, formulation mathématique, ...)
- s'il existe plusieurs méthodes, les évaluer et déterminer la meilleure retenue pour l'étape de conception

Conception Il s'agit de traduire algorithmiquement la méthode retenue. Les informations doivent être représentées par des structures de données adéquates. Les algorithmes sont exprimés de façon structurée, commentée et à un haut niveau en pseudo-langage (voir ci-dessous) indépendant des langages de programmation.

Programmation En théorie il suffit simplement de traduire votre algorithme dans un langage de programmation, C en ce qui nous concerne. Les difficultés rencontrées, les particularités liées au langage et les solutions retenues sont étudiées à ce niveau.

2 Pseudo-langages

Vous pouvez utiliser l'un des deux formalismes suivants pour décrire les algorithmes. Le premier (PDL) est un langage textuel proche des langages de programmation tels que C, Pascal ou ADA. Le principal défaut de PDL est sa linéarité (textuelle), qui implique une "conception en profondeur d'abord" : il faut détailler un bloc avant de passer au suivant. Le deuxième formalisme des arbres programmatiques permet plus facilement une "conception par niveaux" ou descendante : vous commencez par définir les blocs de haut niveau puis vous les expliciter peu à peu, éventuellement en les découpant en sous-blocs.

Exemple Cet exemple servira pour illustrer ces deux formalismes. Soit à calculer le nombre d'occurences d'une lettre dans une suite de caractères terminée par un '.'. Ce problème peut se modéliser par le calcul d'une suite récurrente $\{N_i\}$ donnant pour chaque pas de traitement le nombre d'occurences de la lettre recherchée :

informations : soit l la lettre recherchée, $\{c_i\}$ la suite de caractères

valeur initiale : $N_0=0$

relation de récurrence (ordre i quelconque) :
si $l=c_i$ alors $N_i=N_{i-1}+1$ sinon $N_i=N_{i-1}$

terminaison : c_i='.'

La formulation récurrente se prête facilement ici à l'analyse du problème, il s'agit maintenant d'écrire l'algorithme en pseudo-langage.

3 Le pseudo-langage PDL

Les types élémentaires de variables sont :
- les entiers : Entier
- les réels : Reel
- les caractères : Caractere
- les booléens : Booleen

Un algorithme est décomposé en actions et fonctions.

– Forme générale d'une action

action <nom> (p_1, p_2, ...)
données : p_i : <type>
résultats : p_j : <type>
données/résultats : p_k : <type>
locales : v_i : <type>
<instructions>
fin action

– Forme générale d'une fonction

fonction <nom> (p_1, p_2, ...) : <type résultat>
données : p_i : <type>
locales : v_i : <type>
<instructions>
résultat : <expression>
fin fonction

- On dispose des opérateurs habituels : +, -, *, <, >, =, ...
- Instructions de lecture : **lire**(x, y, ...)
  et d'écriture : **ecrire**(x,y,...)
  sur les flots d'entrée/sortie.
- Instruction **d'affectation** :
  <variable> : = <expression>
- Les **commentaires** se notent entre **{accolades}**.

Les structures de contrôle se notent comme suit :

– Séquence

<instruction 1>
<instruction 2>
...
<instruction n>

– Alternative (conditionnelle)

si <condition> **alors**
    <instructions>
sinon
    <instructions>
fsi

– Boucle dynamique

tantque <condition> **faire**
    <instructions>
fait

– Boucle statique

pour i **de** n **à** m **pas** p **faire**
    <instructions>
fait

4 Exemples en PDL

– Une action qui calcule le min et le max de deux entiers :

action minmax (x,y,min,max)
{calcule le min et le max de x, y dans min et max}
données : x,y : Entier
résultats : min,max : Entier
si x<y **alors**
    min :=x
    max :=y
sinon
    min :=y
    max :=x
fsi
fin action

– Une action qui permute deux variables :

action permuter (x,y)
{permute les variables x et y}
données/résultats : x,y : Entier
locales : temp : Entier
temp :=x
x :=y
y :=temp
fin action

– Conception de l'exemple d'introduction : les caractères c_i sont obtenus par lecture séquentielle au travers d'une variable c (instruction lire(c)) et on associe à la suite $\{N_i\}$ une variable informatique n.

action nboccur (l,n)

données : l : Caractère {on considère toujours le flot d'entrée implicite}

résultats : n : Entier {nombre d'occurences de la lettre l}

locales : c : Caractère {caractère courant}

{init}

n :=0

lire(c)

{calcul}

tantque c <> '.' **faire**

    si c = l **alors** n :=n+1 **fsi**
    lire(c)

fait

fin action

– une fonction qui calcule le maximum de deux entiers :

fonction maximum (x,y) : Entier

données : x,y : Entier

locales : max : Entier {le maximum}

si x < y **alors** max :=y **sinon** max :=x

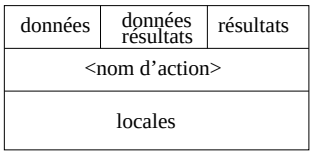
résultat : max

fin fonction

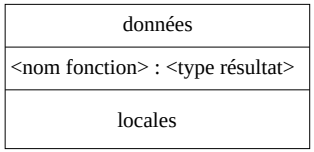
5 Les arbres programmatiques

Les types élémentaires et les instructions de base sont les mêmes. Les actions et structures de contrôle se notent ("se dessinent") comme suit :

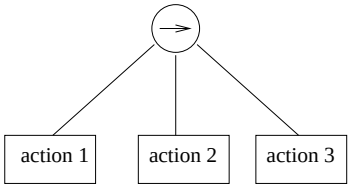
– Action



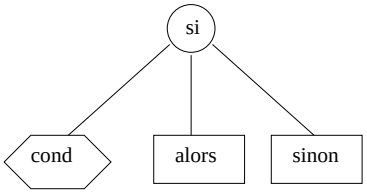
– Fonction



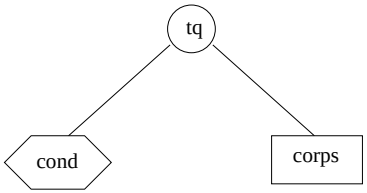
– Séquence



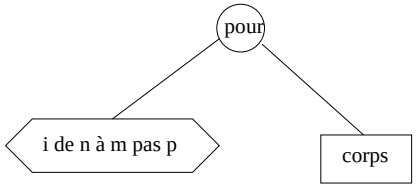
– Alternative



– Boucle dynamique



– Boucle statique



6 Exemples d'arbres programmatiques

Les mêmes exemples se présentent comme suit. La déclaration commentée des paramètres et des variables locales peut se faire en dehors des boites. On remarque que la notation graphique des arbres programmatiques est moins condensée mais plus structurée. On peut facilement isoler les blocs (sous-arbres) de l'algorithme et leur donner un intitulé (comme "init" et "calcul").