Paquet Algorithmique

1 Présentation

Le paquet *algorithmique* est une traduction française de *pseudocode*, un paquet LaTeX créé par D.L. Kreher et D.R. Stinson pour taper une thèse sur l'algorithmique en 1999. Ce paquet est essentiellement un environnement qu'on appelle en insérant

```
\usepackage{algorithmique}
```

dans le préambule (avant le début du document) et en insérant le pseudocode entre

```
\begin{pseudocode}
```

et

```
\end{pseudocode}
```

Le paquet utilise lui-même deux autres paquets, qui sont donc nécessaires pour l'utiliser:

1. Le paquet *fancybox*, qui sert à placer les algorithmes dans des boîtes;
2. Le paquet *ifthen*, qui permet d'avoir des réglages dépendant de la boîte choisie. Le pseudolangage utilisé est inspiré du langage de programmation *Pascal*, spécialisé dans l'algorithmique.

2 Exemples

2.1 Boucle "pour"

Pour calculer $\sum_{k=1}^{n} k$ avec la méthode algorithmique, on utilise le fait que la suite $\sum_{k=1}^{n} k$ est récurrente, avec $\sum_{k=1}^{n} k = \sum_{k=1}^{n-1} k + n$; on a donc besoin d'une variable s destinée à stocker les termes successifs de la suite, et dans une boucle sur k, on ajoute k à s: L'algorithme est décrit dans une boîte ombrée:

Algorithme 2.1: SOMME D'ENTIERS(n)

Commentaire: Calcul de nombres triangulaires

variables entières k, s
pour $k \leftarrow 0$ **jusqu'à** n
  faire $s \leftarrow s + k$
retourne (s)

Le pseudocode ci-dessus a été obtenu en tapant dans LaTeX:

```
\begin{pseudocode}[shadowbox]{Somme d'entiers}{n}
\COMMENT{Calcul de nombres triangulaires}\\
\ENT k,s\\
\POUR k \PREND 0 \JUSQUE n \FAIRE
s \PREND s+k\\
\RETURN{s}
\end{pseudocode}
```

2.2 Boucle "Tant que"

La méthode de Héron pour calculer $\sqrt{2}$ consiste à calculer des termes suffisamment lointains des suites r_n et s_n définies récursivement par $r_{n+1} = \dfrac{r_n + s_n}{2}$ et $s_{n+1} = \dfrac{2}{r_{n+1}}$. Bien que les suites r_n et s_n ne soient pas adjacentes, elles convergent toutes les deux vers $\sqrt{2}$ et on estime que lorsque $|s_n - r_n| < 10^{-10}$, r_n est une valeur approchée de $\sqrt{2}$ suffisamment bonne (à 10^{-10} près en tout cas). L'algorithme de Héron est présenté dans une boîte rectangulaire à bords arrondis:

Algorithme 2.2: MÉTHODE DE HÉRON()

Commentaire: Calcule une valeur approchée de $\sqrt{2}$

variables réelles r, s
$r \leftarrow 1$
$s \leftarrow \dfrac{2}{r}$
répéter
$\begin{cases} r \leftarrow \dfrac{r+s}{2} \\ s \leftarrow \dfrac{2}{r} \end{cases}$
jusqu'à $|r-s| \leqslant 10^{-10}$
Afficher (r)

Pour obtenir l'algorithme ci-dessus, on a entré

```
\begin{pseudocode}[ovalbox]{Méthode de Héron}{ }
\COMMENT{Calcule une valeur approchée de }\sqrt{2}\\
\REEL r,s\\
r \PREND 1\\
s \PREND \dfrac{2}{r}\\
\REPETE \DEBUT
r \PREND \dfrac{r+s}{2}\\
s \PREND \dfrac{2}{r}\\
\FIN
\JUSQUA \left| r-s \right| \leqslant 10^{-10}\\
\AFF r\\
\end{pseudocode}
```

2.3 Récursivité

Voici la version récursive de l'agorithme d'Euclide pour calculer un pgcd; on la présente dans une boîte à traits horizontaux:

Algorithme 2.3: PGCD(a, b)

Commentaire: Calcul du pgcd par l'algorithme d'Euclide

si a divise b
retourne (a)
sinon
retourne (pgcd(a,b-a))

On l'a obtenu en entrant le code suivant:

```
\begin{pseudocode}[ruled]{pgcd}{a,b}
\COMMENT{Calcul du pgcd par l'algorithme d'Euclide}\\
\SI \mbox{ a divise b}\\
\RETOURNE{a}
\SINON \\
\RETOURNE{\mbox{pgcd(a,b-a)}}
\end{pseudocode}
```

3 L'environnement "pseudocode"

3.1 Appel

Pour entrer dans l'environnement *pseudocode*, on fournit le nom de l'algorithme et la liste de ses variables. On peut aussi donner entre crochets, le style de la boîte choisie. Voir ci-dessus pour des exemples.

3.1.1 boîtes

Les styles de boîtes possibles sont les suivants:

shadowbox	ombrée
doublebox	bord dédoublé
ovalbox	coins arrondis
Ovalbox	ronds plus grands
framebox	rectangle
plain	pas de bords
ruled	traits horizontaux
display	sans titre

3.2 Commentaires

Un commentaire (externe à l'algorithme) est donné par

```
\COMMENT{commentaire}
```

où le commentaire est donné entre accolades. Les commentaires sont en mode texte. Donc dans un commentaire, il faut retourner dans l'environnement *math* si on veut entrer des formules mathématiques.

3.3 Variables

On peut déclarer des variables avec un mot-clé parmi les suivants: BOOL, ENT, REEL, CHAINE, TABLEAU, LOCAL, GLOBAL, EXTERNE, représentant respectivement les booléens, les entiers, les réels, les chaînes de caractères, les tableaux, les variables locales, globales et externes. Chaque type de variable est bien entendu précédé de son backslash comme toutes les instructions de *pseudocode*.

3.4 Affectation

L'affectation est écrite

```
\PREND
```

et représentée par une flèche vers la gauche.
Ainsi,

```
n \PREND n+1
```

donne $n \leftarrow n + 1$. Le texte dans l'environnement *pseudocode* est par défaut lui-même placé dans l'environnement *math*. Donc il est parfois nécessaire d'utiliser *mbox* pour revenir provisoirement au mode texte, et le saut à la ligne se fait avec le *backslash* dédoublé.

3.5 Entrées et sorties

L'entrée d'une variable se code

```
\ENTRER{variable}
```

et sa sortie

```
\AFF{variable}
```

Le renvoi d'une variable par une procédure s'écrit

```
\RETOURNE{variable}
```

3.6 blocs

un bloc commence par

```
\DEBUT
```

et se termine par

```
\FIN
```

Tout ce qui est entre les deux balises est indenté et regroupé derrière une accolade. On peut aussi mettre des sous-blocs dans les blocs.
Par exemple,

```
\DEBUT
ligne 1\\
ligne 2\\
ligne 3
\FIN
```

produit l'effet suivant:
$\begin{cases} ligne1 \\ ligne2 \\ ligne3 \end{cases}$
On remarque que le double *backslash* n'est pas nécessaire à la fin, et que les espaces ont disparu à l'affichage: Il faut mettre les lignes de texte dans des *mbox* pour éviter cela.

4 Tests et boucles

4.1 Tests

Un test commence par

```
\SI test
\ALORS instruction
\SINONSI instruction
\SINON instruction
```

Il n'y a pas de limite au nombre de *SINONSI* et les instructions peuvent être remplacées par des blocs. L'effet obtenu est le suivant:

si *test*
  alors *instruction*
  sinon, si *instruction*
  alors *instruction*

4.1.1 Logique

Les opérations logiques sont représentées par les mots-clé VRAI, FAUX, NON, ET et OU (précédés de leur antislash).

4.2 Boucles

4.2.1 À nombre prédéfini d'exécutions

Pour des boucles où l'indice i va de 0 à 10 par exemple on entre

```
\POUR i=0 \JUSQUE 10 \FAIRE
```

ou mieux encore

```
\POUR i \PREND 0 \JUSQUE 10 \FAIRE
```

Lorsque l'indice est élément d'un ensemble on utilise *POURTOUT*:

```
\POURTOUT x \in \mathbb{N} \FAIRE
```

4.2.2 À condition de sortie

Une boucle *tant que* s'écrit

```
\TANTQUE test \FAIRE instruction
```

Une boucle *jusqu'à* s'écrit

```
\REPETE instruction \JUSQUA test de sortie
```

4.3 Sous-programmes

On peut vouloir décrire une procédure avant le programme principal. Toutefois dans ce cas on est plus dans la programmation que dans l'algorithmique:

```
\begin{pseudocode}{Programme}{arguments}
\PROCEDURE{utilitaire}{variables}
corps
\FINPROCEDURE
\MAIN
corps\\
\APPEL{utilitaire}{variables}\\
corps\\
\ENDMAIN
\end{pseudocode}
```

Algorithme 4.2: PROGRAMME(*arguments*)

procédure UTILITAIRE(*variables*)
corps

principal
corps
UTILITAIRE(*variables*)
corps

La sortie d'un programme est codée par

```
\EXIT
```

4.4 Numérotation

On peut numéroter des lignes par

```
\STMTNUM{2cm}{repère}
```

avec réglage de la distance entre l'instruction et le numéro de ligne.