

Résumé de la syntaxe algoscript avec la traduction du langage algorithmique

Dans la suite, les instructions algorithmique en bleu sont traduites en Javascript en rouge.

Syntaxe générale

Commentaires

Pour placer du texte en commentaire en javascript, on a deux possibilités:

- bloc de commentaire: texte placé entre les symboles `/*` et `*/`. Attention, on ne peut pas emboîter ces blocs.
- commentaire en ligne: texte placé après les symboles `//`.

En-tête

Il n'y a pas en javascript d'équivalent pour la déclaration : Algorithme *nom_algorithme*. On commence directement le programme. Cependant, on impose dans ce cours de traduire l'en-tête en commentaires :

Algorithme triTableau devient `// Programme triTableau`

Les déclarations de variables se traduisent ainsi

Variables

`i : entier // variable de boucle`

`res : entier // compteur`

devient :

`var i; // variable de boucle`

`var res; // compteur`

ou encore, de façon plus compacte :

`var i, res; // i est une variable de boucle, res un compteur`

Notez que le type n'est pas précisé (voir ci-dessous le typage dynamique). La notion de portée de variables, qui conditionne l'endroit du programme où elles doivent être déclarées, sera vue en temps utile.

Instructions et blocs

Pour l'instant, on appellera instruction une ligne de l'algorithme, correctement traduite en javascript. Toutes les instructions doivent être suivies d'un point virgule : la ligne d'algorithme `x←y+4;` devient `x=y+4;`

On peut vouloir réunir plusieurs instructions en une seule en créant un bloc, qui est délimité par des accolades. Dans ce cas, le contenu du bloc doit être indenté.

Par exemple,

`x←y+4;`

`z←x*8;`

peut devenir

`x=y+4;`

`z=x*8;`

ou bien

{

`x=y+4;`

`z=x*8;`

}

Le calcul effectué sera le même ici, la seule différence est que la première version contient deux instructions (deux lignes avec deux points-virgules) alors que la deuxième d'en contient qu'une (un bloc, qui contient lui-même deux instructions).

Les opérations de base

En javascript, il y a trois types de base: les nombres (numbers), les booléens (boolean) et les chaînes de caractères (string). Il y a aussi un type "general" objet (object). Pour obtenir le type d'une expression, on peut utiliser la fonction `typeof(<expression>)` (ex: `typeof(5+2)`, `typeof('bonjour')`,....).

Le typage des variables est dit dynamique (une variable est forcément du même type que ce qu'elle contient et ce type peut changer).

Voici les différentes opérations de bases en fonction des différents types.

Opérations d'affectation (tous les types)

On affecte une valeur, par exemple 4, à une variable, par exemple x, ainsi:

`x ← 4;` devient `x=4;`

Le type de la variable x devient celui de la valeur affectée, ici entier. On parle de *typage dynamique*.

Opérations de tests (tous les types)

- égalité: `x=4` devient `(x==4)`
- comparaison stricte: `x<4` devient `(x<4)`
- comparaison "ou égal": `x≤4` devient `(x<=4)`
- différent: `(x !=4)`

Opérations sur les nombres

- addition: 5+3 devient 5+3
- multiplication: 5*3 devient 5*3
- division entière: 5 div 3 ou [5/3] devient Math.floor(5/3)
- reste de la division entière: 5 mod 3 devient 5 % 3
- De nombreuses fonctions mathématiques sont accessibles via la bibliothèque Math comme par exemple Math.cos(4), Math.abs(-3), Math.floor(2.54), Math.ceil(2.54),...

Opérations sur les booléens

- et logique: vrai et vrai devient true && true
- ou logique: vrai ou vrai devient true || true
- non logique: non vrai devient ! true

Opérations sur les chaînes de caractères

La concaténation de chaînes de caractères s'écrit '+'. Par exemple, le résultat de l'expression 'bon'+'jour' est la chaîne de caractère 'bonjour'.

La plupart des fonctions utilisées en langage algorithmique sont également définie dans le langage Algoscript, comme Longueur(ch), CaractereEn(ch,i),...

Les structures de contrôle

Conditionnelles

Nous ne donnons ici que la traduction d'une alternative.

```
if (condition) {  
    si (condition)           instruction_alors  
        Alors instruction_alors } else {  
    Sinon instruction_sinon   instruction_sinon  
    fin si                   }  
  
par exemple          // nb: dans ce cours, on met systématiquement toutes les accolades ce qui  
si (x<40)            permet de vérifier  
    Alors Ecrire(x+' est  
    petit');                // que le code est bien indenté, une aide au débogage.  
    if (x<40) {  
        Ecrire(x+' est petit');  
    } else {  
        Ecrire(x+' est grand');  
    }  
fin si
```

Boucle pour

```
var i; // si besoin  
for(i=1; i<=10; i=i+1) {  
    Ecrire(i);  
}  
pour i allant de 1 à 10 faire  
    Ecrire(i);  
fin pour  
// là encore, on met systématiquement les accolades.  
// On n'utilise pas non plus de raccourcis du type "i++".  
// Ne pas oublier de définir la variable de boucle.
```

Boucle Tant que

```
x<-1;           var x=1;  
tant que (x<20) faire   while (x<20) {  
    x<- x*2;         x=x*2;  
fin tant que       }  
}
```

Boucle répéter ... tant que

```
x<-1;           var x=1;  
répéter          do {  
    x<-x*2;         x=x*2;  
Tant que (x<20) } while (x<20);
```

Les algorithmes

[Algorithme toto](#)

[Variables](#)

message : chaîne de caractère;

// Programme toto

var message;

message=Saisie();

Ecrire(message);

[Début](#)

message <- Saisie();

Ecrire(message);

[Fin](#)

Les fonctions

Fonction truc(x : réel, n : entier) : réel;

```
function truc(x,n) {
```

Variables

i : entier;

```
var i,res;
```

res : réel;

```
res=1;
```

Début

```
for(i=1; i<=n; i=i+1) {
```

res <- 1;

```
res=res*x;
```

pour i allant de 1 à n faire

```
}
```

res <- res * x;

```
}
```

fin pour

```
return res;
```

retourner res;

```
}
```

Fin

Un exemple complet (définition d'une fonction utilisée dans un algorithme)

On définit si possible la/les fonctions avant de les utiliser dans un algorithme

Fonction truc(x : réel, n : entier) : réel;

Variables

i : entier;

```
function truc(x,n) {
```

res : réel;

```
var i,res;
```

Début

```
res=1;
```

res <- 1;

```
for(i=1; i<=n; i=i+1) {
```

pour i allant de 1 à n faire

```
res=res*x;
```

res <- res * x;

```
}
```

fin pour

```
return res;
```

retourner res;

```
}
```

Fin

Algorithme utiliser truc

```
// Programme utiliser truc
```

Variables

```
var nombre
```

nombre : réel;

```
nombre=Saisie();
```

```
Ecrire(truc(nombre,2));
```

Début

nombre <- Saisie();

Ecrire(truc(nombre,2));

Fin

Les tableaux en javascript

Les tableaux en javascript sont tous dynamiques. Ils se déclarent par `var T=[];` ou encore `var T=[5, 12, 23];` pour créer un tableau prérempli.

Les éléments du tableau sont accessible en lecture et en écriture en utilisant son indice par exemple `Ecrire(T[5]);` ou `T[5]=25.3;`

Il est souvent utile de connaître le nombre d'éléments d'un tableau. Ceci se fait grâce à la fonction `Taille(T)` (ou encore `T.length` en javascript pur).

Pour créer un tableau à deux dimensions, il faut nécessairement utiliser un tableau à une dimension. Il y a deux solutions:

1. Faire un calcul savant sur les indices. Pour mémoriser un tableau de n lignes et m colonnes, il faudra créer un tableau de $n*m$ cases. L'élément situé en ligne x et colonne y sera stocké dans la case `[x*m+y]` dans le tableau.
2. Noter qu'on peut mettre ce qu'on veut dans une case de tableau, y compris un tableau. Pour mémoriser un tableau de n lignes et m colonnes, il faudra créer un tableau de n cases donc chaque case contiendra un tableau de m cases. L'élément situé en ligne x et colonne y sera stocké dans la case `[x][y]` dans le tableau.

Conseils aux étudiants

1) Il n'y a pas de secret, il faut beaucoup s'entraîner avant de traduire rapidement et sans erreur un code algorithmique. Pour limiter les bugs, il faut absolument indenter convenablement son code (utiliser l'outil d'indentation automatique), vérifier que les parenthèses et accolades sont fermées convenablement. Là encore, l'interface détecte les parenthèses qui se correspondent ce qui peut être d'une grande aide.

2) Pour toutes les fonctions particulières à l'environnement AlgoScript, notamment les sorties graphiques et sonores, n'hésitez pas à vous référer au mémento qui se trouve directement intégré dans l'interface.

Modifié le: dimanche 1 décembre 2013, 00:26