L’Algorithmique

>>> Comment "informatiser" une idée avant de la programmer.

Description :

Ce cour à pour but de donner les définitions des concepts d’algorithme, de programme et de tous les éléments concernant ce langage (les types variables et leurs mises en œuvre, …). Ces éléments seront accompagnés d’exemples permettant d’en voir la programmation.

L’Algorithmique

>>> Comment "informatiser" une idée avant de la programmer.

Sommaire :

I) Définitions générales de l’algorithmique
1) Algorithme
2) Programme
II) Les commentaires
1) Utilisation
III) Les types et leurs déclarations algorithmiques
1) Le Booléen
2) L’entier
3) Le Flottant
4) Le Caractère
IV) Les variables
1) Définition
2) Affectation
V) Les constantes
1) Définition
2) Déclaration
VI) Les instructions
1) Définition
2) Utilisation
VII) les conditions
1) Définitions
2) Utilisation
VIII) Les opérations
1) Définition
2) Opérations arithmétiques
3) Opérations relationnel
4) Opérations booléennes
IX) Les structures de contrôle
1) Sélection simple et complète
2) Les boucles "TantQue"
3) Boucle Pour ... FinPour
Conclusion

I) Définitions générales de l’algorithmique

1) Algorithme :

– Un algorithme est un moyen d’atteindre un but en répétant un nombre fini de fois un nombre fini d’instructions, par exemple pour le nombre de cases d’un tableau, on peut faire une boucle dans laquelle on incrémente une variable.
Dans la programmation, la difficulté principale est d’écrire complètement et sans ambiguïté un processus complexe, d’où la nécessité d’utiliser un langage formalisé pour la description des algorithmes.

2) Programme :

– Un programme est la traduction d'un algorithme en un langage compatible ou interprétable par un ordinateur. Il est souvent écrit en plusieurs parties, dont une qui pilote les autres qu'on appelle le programme principal (un peu comme la fonction main() en C).

II) Les commentaires

1) Utilisation

– Un programme est écrit pour un humain, il est donc très important d'en faciliter la lecture, en utilisant, d'une part les indentations et, d'autre part, les commentaires.
– Les indentations ne sont autres que les alinéas qu'on laisse volontairement en début de ligne. Ils permettent de faire ressortir le code contenu dans des structures de contrôle, telles que les itérations ou les conditions.
– Les commentaires permettent d'éclaircir certains passages de codes. Ainsi, si l'on n'a pas touché à son travail depuis un certain temps ou si c'est un projet dont on a récupéré les sources, il est très souvent agréable d'avoir des commentaires pour comprendre ce que le développeur à voulu faire.
– En algorithmique, on commente son code en utilisant soit le soulignage soit le double "slash", comme ceci :

Voici un premier commentaire

// et en voici un second

III) Les types et leur déclaration algorithmique

1) Le Booléen

– C'est le type le plus simple il ne prend que deux valeurs : vrai ou faux.
– Sa déclaration se fait de la manière suivante : *a, b* : booléen

2) l'Entier

– Son ensemble de définition est : Z (comme en mathématique)
– Déclaration algorithmique : *n1, n2* : entier

3) Le Flottant

– Son ensemble de définition est : F (comme en math)
– Déclaration algorithmique : *x1, x2* : flottant

4) Le Caractère

- Son ensemble de définition est : la table ASCII
- Déclaration algorithmique : *c1, c2* : caractère
- La table ASCII est codée sur 7 bits et permet donc d'avoir tous les caractères usuels (États-uniens) soit 128 positions.

IV) Les variables

1) Définition

- Une variable est un nom donné à une valeur. La variable est définie par son caractère changeant, c'est-à-dire qu'elle pourra prendre une multitude de valeurs.
- La déclaration d'une variable se fait en lui donnant un nom et un type, par exemple pour utiliser une variable de type entier on écrira :

température : entier

2) Affectation

- L'affectation désigne l'opération qui permet d'attribuer une valeur à une variable. En algorithmique, l'affectation d'une variable se fait comme ceci :

température <- 20

- On attribut la valeur 20 à la variable qui porte le nom *température*.

V) Les constantes

>

1) Définition

- La constante est en quelque sorte une variable qui ne varie pas dans le temps, elle doit donc être initialisée dès sa déclaration.

2) Déclaration

- Une constante est déclarée et initialisée comme suit :

seuilTempérature <- 25 : entier

Ici seuilTempérature est une constante de type entier dont la valeur est 25.

VI) Les instructions en algorithmique

1) Définition

– Une instruction est une expression du langage qui peut être soit simple soit composée, on parle alors d'un bloc d'instructions.
En algorithmique, on écrit une instruction simple par ligne et les blocs d'instructions sont délimités par un début et une fin.

2) Principales instructions utilisées en algorithmique

On en compte une vingtaine, les voici (par ordre alphabétique et francisés) :

- Afficher
- Dans
- DébutDéclarations
- DébutFonction
- DébutProcédure
- DébutPogramme
- Faire
- FinDéclarations
- FinFonction
- FinPour
- FinProcédure
- FinProgramme
- FinSi
- Retourner
- Saisir
- Si
- Sinon
- TantQue

– A celles-ci, il faut bien sûr ajouter les fonctions mathématiques usuelles comme sin(x), par exemple, ou Alea(n), fonction qui génère un entier au hasard entre 0 et n-1.

Nous retrouverons ces instructions dans des exemples tout au long de ce cours.

VII) Les conditions

1) Définition

– Une condition est une expression écrite entre parenthèses dont la valeur de résultat est booléenne. Elle permet ainsi de prendre une décision élémentaire suivant son résultat : vrai ou faux.
– Dans un langage quel qu'il soit, les conditions sont mises en œuvre au moyen de structures de contrôles.

2) Utilisation

Pour observer l'utilisation des conditions, voir la partie concernant les structures de contrôles.

VIII) Les opérateurs

1) Les opérateurs arithmétiques

- addition (**+**) : a+b
- Soustraction (**-**) : a-b
- opposé (**-**) : -a
- produit (*****) : a*b
- division (**/**) : a/b (attention à la division entière et surtout par 0)
- modulo (**%**) : a%b

2) Les opérateurs relationnels

- égal : **==**
- différent : **?**
- inférieur : **<**
- supérieur : **>**
- inférieur ou égal : **=**
- supérieur ou égal : **=**

3) Les opérateurs booléens

- négation : **NON(a)**
- Ou : a **OU** b
- et : a **ET** b

IX) Les structures de contrôles

1) Sélection simple et complète

- Notation de la sélection simple :

```
    Si (condition)
      bloc_instructions
    FinSi
```

- Notation de la sélection complète :

```
    Si (condition}
      bloc_instructions
```

```
    Sinon
      bloc-instructions
    FinSi
```

– Exemple de sélection complète : Calcul des résultats d'une équation du second degré ($ax^2+bx+c=0$)

```
DébutDéclaration
  a <- 1.0, b <- -1.0, c<- -1.0 : flottant
  delta <- (b*b) - 4 * (a*c), r : flottant
FinDéclaration

Si (delta < 0)
  Afficher("Pas de solution réelle")
Sinon
  Si (delta == 0)
    Afficher("Une solution double : ", -b/(2*a))
  Sinon
    r <- Racine(delta)
    Afficher("Deux solutions réelles : ", (-b-r)/(2*a), " et ", (-b+r)/(2*a))
  FinSi
FinSi
```

2) Les boucles "TantQue"

– Notation de la boucle "Faire ... TantQue"

```
Faire
  bloc_instructions
TantQue(condition)
```

— Exemple :

```
faire
  Afficher("Appuyer sur '5'")
  Saisir(n)      // la fonction "Saisir()" permet de récupérer la saisie sur le clavier
TantQue(n?5)
```

Commentaire de l'exemple : *L'ordinateur exécutera cette boucle tant que "l'interface chaise/clavier" n'aura pas saisie le chiffre 5.*

– Notation de la boucle "TantQue ... FinTantQue"

TantQue (condition)
bloc_instructions
FinTantQue

— Exemple :
L'exemple réalise un compteur diviseur par 2 de n (n étant le nombre à tester)

```
DébutProgramme
    DébutDéclaration
        n <- 32 : entier
        cpt <- 0 : entier
    FinDéclaration

    TantQue(n%2 == 0)
        n <- n/2
        cpt <- cpt + 1
    FinTantQue
```

```
    Afficher("32 est ", cpt, " fois divisible par 2")
FinProgramme
```

3) Boucle Pour ... FinPour

- Notation de la structure "Pour ... FinPour"

```
Pour cpt Dans [a..b] ParPasDe n
  bloc_instructions
FinPour
```

— Exemple :
Dans cet exemple, nous allons voir la mise en œuvre du calcul factorielle d'un entier n

```
DébutProgramme
    DébutDéclaration
        n <- 5, fact <- 1, cpt : entier
    FinDéclaration

    Pour cpt Dans [1..n] ParPasDe 1
        fact <- fact*cpt
    FinPour

    Afficher("5! = ", fact)
FinProgramme
```

Dans cet exemple, la boule Pour ... FinPour permet de calculer le factorielle de 5 en incrémentant cpt par pas de 1, c'est à dire qu'on réalise le calcul suivant : 1*2*3*4*5. Enfin on affiche le résultat de l'opération.

Conclusion

Dans cette Article, nous avons pu voir (ou revoir) les bases et les fondements du langage Algorithmique. Un second article complétera celui-ci, pour présenter le fonctionnement des fonctions et des procédures. Ces deux types de structure montreront comment rendre ce langage plus performant.