Cours algorithme

Introduction

Un algorithme est par définition une suite finie d'instruction permettant de résoudre un problème.

Un programme informatique est un ensemble de commande donnée à la machine écrite dans un langage spécifique pour effectuer une série d'opération déterminer afin d'obtenir un résultat.

Il faut noter qu'un algorithme est un ensemble ordonné d'opérations précises qui doivent être suivie dans l'ordre. Des lors que l'algorithme est établi, le programme associé est une suite d'instruction permettant à un système informatique d'exécuter la tâche demandée. Un programme informatique est écrit dans un langage de programmation compréhensible par un ordinateur. Un langage de programmation est un ensemble de règle de vocabulaire et de règles grammaticales compréhensibles par un ordinateur. Un programme décrit un algorithme dans un langage de programmation donné.

$$Problème \rightarrow Algorithme \rightarrow Programme \rightarrow Résultat$$

Exemple :

```
Algorithme : division
Variable
(a,b) :réel
C : réel
Début
Ecrire (‘‘Entrer un nombre de 1 à 10’’)
Lire (a)
Ecrire (‘‘Entrer un nombre de 1 à 10’’)
Lire(b)
Si b=a alors écrire (‘‘ERREUR’’)
Sinon c=a/b
Ecrire (c)
Fin.
```

Les variables

Elles ont pour but de stocker l'information en mémoire centrale durant l'exécution d'un programme. Une variable possède 4 propriétés :

- Nom
- Adresse
- Un type
- Une valeur

La première chose à faire avant de pouvoir utiliser une variable est de lui donner un nom et une adresse et de lui coller un type. La déclaration des variables se fait tout au début de l'algorithme avant même les instructions. Le nom de la variable obéi à un ensemble de règle changeant en fonction du langage. Toutefois, une règle absolue est qu'un nom de variable peut comporter des lettres et des chiffres mais qu'il exclut la plupart des signes de ponctuation en particulier les espaces. Lorsqu'on crée une variable il est nécessaire de déclarer le type. Il y a trois grandes catégories de type de variable :

- Les types numérique (réel, entier …)
- Les types alphanumérique (les caractères, chaine de caractères, …)
- Les types Booléen.

Instruction d'affectation

Une fois la variable déclarée, il nous faut lui affecter une valeur. Cette opération consiste à lui attribuer une valeur conformément au type qui a été déclarer. En algorithmique, l'instruction d'affectation se note avec le signe (←).

Exemple : $toto \leftarrow 3$ cela attribut la valeur 3 à la variable numérique toto. Par contre, si toto a été déclaré comme un type booléen, ce ci provoquera une erreur. L'on peut également attribuer à une variable la variable d'une autre variable.

Les instructions de lecture et d'écriture

Lorsque l'algorithme rencontre l'instruction lire, son exécution s'arrête attendant la frappe d'une valeur au clavier. « Lire » est une autre manière d'affecté une valeur à une variable. Avec l'instruction d'affectation (←) c'est le programmeur qui choisi à l'avance quel doit être cette valeur. Avec l'instruction « lire » il laisse le chois à l'utilisateur. Pour écrire quelque chose à l'écran, l'on utilise l'instruction écrire.

Exercice

Ecris un algorithme qui demande un nombre à l'utilisateur puis qui calcule et écrit le carré de ce nombre.

Réponse

```
Algorithme: Carré
variable
a,b : réel
Début
1-ecrire ("entrer un nombre")
2-lire (a)
3-b reçoi a*a
4- ecrire ("Le carré",a,"est",b)
Fin
```

Les structures conditionnelles

Une condition est une expression qui peut prendre l'une des deux valeurs suivantes (vrai ou faux). Une condition est obtenue généralement en combinat un certain nombre d'opérateur de comparaison. Un test est une instruction qui permet d'effectuer un traitement différent selon qu'une condition est vérifier ou non. La forme la plus simple est la suivante $\begin{cases} Si\ (condition) alors\ (instruction) \\ fin\ si \end{cases}$

La forme complète $\begin{cases} si\ condition\ alors\ instruction1 \\ sinon\ instruction\ 2 \\ fin\ si \end{cases}$ si la condition est vraie, l'algorithme effectuera les instructions 1 puis passera aux instructions après ‘‘Fin si’’. Si la condition est fausse, l'algorithme effectuera les instruction2 puis passera aux instructions après fin si.

Exercice

Ecrit un algorithme qui demande deux nombres à l'utilisateur et l'informe si leur produit est négatif ou positif. On ne traite pas le cas ou le produit est nul, on ne doit pas calculer le produit des deux nombres.

Règles de logique

Dans une condition composée employant à la fois des opérateurs ‘‘ET’’ et des opérateurs ‘‘OU’’, la présence de parenthèse possède une influence sur le résultat tout comme dans le cas s'une expression numérique comportant des multiplications et des additions. Toute structure de test recevant une condition composée faisant intervenir l'opérateur ‘‘ET’’ peut être exprimer de manière équivalente avec un opérateur OU et réciproquement.

$$\begin{cases} Si\ A\ ET\ B\ alors\ instruction\ 1 \\ Sinon\ instruction\ 2 \\ Fin\ si \end{cases} \leftrightarrow \begin{cases} Si\ (AouB) alors\ instruction\ 2 \\ Sinon\ instruction\ 1 \\ Fin\ si \end{cases}$$

Exemple :

$$\begin{cases} Si\ il\ fait\ chaud\ ET\ il\ ne\ pleut\ pas; ouvrir\ a\ fenêtre \\ Sinon\ fermer\ la\ fenêtre \\ Fin\ si \end{cases} \leftrightarrow \begin{cases} S'ilne\ fait\ pas\ chaud\ ou\ il\ pleut\ alors\ fermer\ la\ fenêtre \\ Si\ non\ ouvrir\ la\ fenêtre \\ Fin\ si \end{cases}$$

Les boucles

La boucle Pour (For) est utilisé lorsque le nombre d'itération (nombre de fois que l boucle sera exécuter) est connu d'avance. La syntaxe est donnée ainsi :

Pour i allant de début à fin faire instructions
fin pour

Exercice :

Ecrit un algorithme qui affiche les dix premiers entiers à partir de 0

Réponse :

```
Algorithme: premier entier
variable
i: réel
Debut
      pour i allant de 0 à 9 faire
            Ecrire (i)
      Fin pour
Fin
```