Algorithmique et programmation procédurale - TD No 2

Fonctions et procédures - Corrigé

Exercice 1. Écrire une procédure qui permet de trouver la résolution d'une équation bicarrée ($ax^4 + bx^2 + c = 0$).

```
Procédure EquationBicarre(a : Réel, b : Réel, c : Réel)
   Variables delta, x1, x2 : Réel
   Si a = 0 Alors
       Ecrire "Attention, a doit être non nul"
   Sinon
       delta ← b*b-4*a*c
       Si delta < 0 Alors
           Ecrire "Delta négatif, il n'y a pas de solution"
       Sinon
           x1 ← (-b-sqrt(delta))/2a
           x2 ← (-b+sqrt(delta))/2a
           Si x1 >= 0 Alors
               Ecrire "Il y a 4 racines :",-sqrt(x1),sqrt(x1),-sqrt(x2),sqrt(x2)
           Sinon
               Si x2 >= 0 Alors
                   Ecrire "Il y a 2 racines :",-sqrt(x2),sqrt(x2)
               Sinon
                   Ecrire "Il n'y a pas de solution"
               FinSi
           FinSi
       FinSi
  FinSi
FinProcédure
```

Exercice 2. Écrire une fonction booléenne qui permet de tester si un entier donné est premier ou non.

```
Fonction NombrePremier(nb : Entier) : Booléen
       Variables d : Entier, premier : Booléen
       premier ← vrai
       d ← 2
       TantQue ((premier = vrai) et (d <= sqrt(nb)))
              Si (nb mod d = 0) Alors
                     premier ← faux
              FinSi
              d ← d + 1;
       FinTantQue
       retourner premier
FinFonction
```

Autre version :

```
Fonction NombrePremier(nb : Entier) : Booléen
	Variables d : Entier
	Pour d ← 2 à sqrt(nb)
		Si (nb mod d = 0) Alors
			retourner faux
		FinSi
	FinPour
	retourner vrai
FinFonction
```

Exercice 3. Écrire un programme qui permet d'afficher tous les nombres premiers inférieurs à un entier donné.

```
Programme Premiers
Variables n,i : Entier
Début
	Ecrire « Entrez un entier positif »
	Lire n
	Pour i ← 1 à n
		Si NombrePremier(i) Alors
			Ecrire i
		FinSi
	FinPour
Fin
```

Exercice 4. Écrire une fonction qui teste si un nombre est parfait ou non. Un **nombre parfait** est un nombre naturel non nul qui est égal à la somme de ses diviseurs stricts (par exemple : le premier nombre parfait est 6, car 1, 2, et 3 sont les diviseurs stricts de 6 et 1 + 2 + 3 = 6).

```
Fonction NombreParfait(nb : Entier) : Booléen
Variables s,i : Entier
	s ← 0
	Pour i ← 1 à nb-1
		Si (nb mod i = 0) Alors
			s ← s + i
		FinSi
	FinPour
	retourner (nb = s)
FinFonction
```

Exercice 5. Cryptographie 1

Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc. Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui la code selon ce principe, puis affiche le résultat à l'écran.

```
Programme Cryp1
Variables Bla, Cod, Alpha, Let : Caractère
Variables i, Pos : Entier
Début
	Ecrire "Entrez la phrase à coder : "
	Lire Bla
	Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
	Cod ← ""
	Pour i ← 0 à longueur(Bla)-1
		Let ← extrait(Bla, i, 1)
		Si Let ≠ "Z" Alors
			Pos ← trouve(Alpha, Let)
			Cod ← Cod & extrait(Alpha, Pos + 1, 1)
		Sinon
			Cod ← Cod & "A"
		FinSi
	FinPour
	Ecrire "La phrase codée est : ", Cod
Fin
```

Exercice 6. Cryptographie 2 - *le chiffre de César*

Une amélioration du principe précédent consiste à opérer avec un décalage non de 1, mais d'un nombre quelconque de lettres. Ainsi, par exemple, si l'on choisit un décalage de 12, les A deviennent des M, les B des N, etc.

Réalisez un algorithme sur le même principe que le précédent, mais qui demande en plus quel est le décalage à utiliser. Votre sens proverbial de l'élégance vous interdira bien sûr une série de vingt-six "Si...Alors"

```
Programme Cryp2
Variables Bla, Cod, Alpha: Caractère, i, Pos, Décal : Entier
Début
	Ecrire "Entrez le décalage à appliquer : "
	Lire Décal
	Ecrire "Entrez la phrase à coder : "
	Lire Bla
	Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
	Cod ← ""
	Pour i ← 0 à longueur(Bla)-1
		Pos ← trouve(Alpha, extrait(Bla, i, 1))
		Cod ← Cod & extrait(Alpha, (Pos + Décal) mod 26, 1)
	FinPour
	Ecrire "La phrase codée est : ", Cod
Fin
```