L'algorithmique

I2 – Fiche 1

I. Les règles d'écriture

Elément	Règle	Exemple
Variable, nom de programme, fonction, procédure	Lettres et chiffres, pas d'espaces. Commence par une lettre minuscule, puis majuscule au début de chaque mot.	nomDuProgramme
Type de variable	Commence par une majuscule.	Entier
Constantes	Lettres majuscules et chiffres, pas d'espaces.	CONSTANTE
Chaine de caractères	Texte entre "	"chaine de caractères"
Caractère	Caractère entre '	'a'

II. Les types

Type	Définition	Exemple
Naturel	$\mathbb{N}$	2
Entier	$\mathbb{Z}$	-5
Réel	$\mathbb{R}$	-5.2
Caractère	Un seul caractère	'c'
Chaine de caractères	Suite de caractères	"chaine de caractères"

III. Les opérations

Type des opérandes	Opérateurs disponibles	Type résultat
Booléen	non, et, ou, ouExclusif, =, ≠	Booléen
Entier, Naturel	+, -, *, div, mod	Entier, Naturel
Réel	+, -, *, /	Réel
Caractère	succ, pred	Enuméré
Naturel	car	Caractère
Caractère	ord	Naturel
Entier, Naturel, Réel, Caractère	=, ≠, <, ≤, >, ≥	Booléen
Chaîne de Caractères	+	Chaîne
	=, ≠	Booléen

Note : Il peut y avoir un transtypage des entiers et des naturels vers les réels afin de réaliser des opérations entre ces différents types.

IV. Les instructions d'entrée/sortie

Instruction	Fonction	Syntaxe	Exemple
Affectation	Donner une valeur à une variable	var ← valeur	toto ← "toto"
Ecrire	Afficher la valeur d'une (ou plusieurs) variables	ecrire(var1, var2, …)	ecrire(toto)
Lire	Demander à l'utilisateur la valeur à mettre dans une (ou plusieurs) variables	lire(var1, var2, …)	lire(toto)

V. Tests et boucles

Nom	Syntaxe

Si	**si** *expression booléenne* **alors**
	  // instructions si l'expression est vraie
	sinon
	  // instructions si l'expression est fausse
	finsi

Cas où	**cas où** variable **vaut**
	  valeur1 : // instructions si variable = valeur1
	  valeur2 : // instructions si variable = valeur2
	  ...
	  autre : // instructions si rien n'a été fait
	fincas

Tant que	**tant que** *expression booleenne* **faire** // instructions exécutées tant que l'expression est vraie **fintantque**
Pour	**pour** varIncrement ← valeurDebut **jusqu'à** valeurFin **faire** // instructions / à chaque passage, on ajoute 1 à varIncrement **finpour**
Répéter	**répéter** // instructions exécutées tant que l'expression est fausse **jusqu'à ce que** *expression booleenne*

VI. Le programme de « base »

```
Nom : leNomDuProgramme
Rôle : Description du rôle du programme
Entrée : entree1, entree2 : Type, entree3 : Type, …
Sortie : sortie1, sortie2 : Type, sortie3 : Type, …
Entrée/Sortie : es1, es2 : Type, es3 : Type, …
Déclaration : var1, var2 : Type, var3 : Type, …

debut
   // corps du programme
fin
```

VII. Fonctions et procédures

1. Différences et signatures

La fonction prend des valeurs en entrée et renvoie une valeur en sortie. La procédure prend des valeurs (E) et des variables (E/S) en entrée et renvoie des variables (S) en sortie. On choisit généralement la procédure quand on veut plusieurs sorties, ou que l'on veut modifier l'état de variables (qui seront E/S) sans renvoyer de valeur en sortie.

```
fonction nomDeLaFonction (entree1 : Type, entree2 : Type, …) : TypeSortie

procedure nomDeLaFonction (E varEntrees ; S varSorties ; E/S varES)
```

2. Code de « base »

```
fonction/procédure nom (variables E/S) [: Type] // signature
      Déclaration : var1, var2 : Type, var3 : Type, …
debut
      // instructions
fin
```