



Bases – Variables, Entrées / Sorties -
Opérateurs

Objectifs

- Apprendre à représenter des données en mémoire (variable)
- Découvrir les principaux opérateurs arithmétiques
- Découvrir les principaux opérateurs logiques
- Découvrir les entrées / sorties

Algorithme

- Exprimé sous forme de pseudocode
 - Langage pour rédiger des algorithmes
 - Proche des langages de programmation
 - Doit être non ambigu
 - Se lit de haut en bas
 - Est une séquence d'instructions
 - Les instructions sont séparées par « ; »
- Peut être commenté en préfixant le texte par « `//` »
 - Exemple : `// ceci est un commentaire`

Variable – Définition

Une variable est utilisée pour **stocker** une donnée / **valeur** qui sera manipulée dans un algorithme.

Une variable a minimalement comme caractéristiques un **nom** et une **valeur**.

Dans notre cas nous aurons aussi un **type**.

Variable – Noms

- Un nom de variables débute par une minuscule. Chaque première lettre de mots qui suivent est écrit en majuscule
 - Exemple : **moyenneSalaire, resultatAdmission**
- Le nom d'une variable doit avoir un sens :
 - **x** n'est pas un nom qui a du sens
 - **moyenneSalaire, resultatAdmission**

Variable – Types

- Types des données (= ensemble des valeurs possibles) :
 - booléen : VRAI ou FAUX
 - entier : -42, 0, 42, 97
 - réel : -42.0, 0.0, 3.1415, 42.0
 - caractère : 'a'
 - chaîne : "bonjour"



Variable – Déclaration

- Avant d'être utilisée, une variable doit être : déclarée et initialisée
- Pour déclarer et l'initialiser une variable, on va utiliser la notation suivante :

`<type> <nomVariable> = <valeurInitialisation>`

- Exemple :

`entier somme = 0;`

`réel moyenne = 0.0;`

`booléen estAdmis = VRAI;`

Opérations - booléen

- Trois opérations sur les booléens : et, ou, non

et	FAUX	VRAI
FAUX	FAUX	FAUX
VRAI	FAUX	VRAI

ou	FAUX	VRAI
FAUX	FAUX	VRAI
VRAI	VRAI	VRAI

non	FAUX	VRAI
	VRAI	FAUX

- Exemple :

booléen **b1** = FAUX;

booléen **b2** = b1 et VRAI; // => FAUX

booléen **b3** = (b1 ou VRAI) et non b2; // => VRAI

entier – Opérateurs

Opérateur	Définition	Exemple
+	Addition	30 + 12 // 42
-	Soustraction	-20 - 22 // -42
*	Multiplication	21 * 2 // 42
/	Division	23 / 2 // 11 (Quotient)
%	Modulo – reste division entière	23 % 2 // 1 (Reste)

réel – Opérateurs

Opérateur	Définition	Exemple
+	Addition	30.12 + 12.30 // 42.42
-	Soustraction	-20.11 - 22.31 // -42.42
*	Multiplication	21.0 * 2.2 // 46.2
/	Division	23.0 / 2.0 // 11.5

⇒ Pas de modulo

⇒ Si on mélange un entier et un réel, le résultat est réel

chaine – Opérateurs

Opérateur	Définition	Exemple
[]	Obtenir le caractère de la chaine à la position donnée. Attention, débute à 0	"Bonjour"[0] // 'B' "Bonjour"[1] // 'o'
.Longueur	Obtenir la longueur d'une chaine	"Bonjour".Longueur // 7 chaine texteSaluer = "Bonjour" texteSaluer.Longueur // 7
+	Concaténation	"Bonjour " + "tout le monde !" // "Bonjour tout le monde !"

Conversion

De	Vers	Exemple
entier	réel	<code>réel sommeNotes = 0</code> <code>// 0 est un entier ici</code>
tous	chaine	<code>réel pi = 3.1415</code> <code>chaine valeurEnChaine = pi.VersChaine()</code>

⇒ D'autres conversions sont possibles nous les verrons plus tard

Comparaison

Comparaison	Définition	Exemple
<	Inférieure à	<code>23 < 7 // FAUX</code> <code>42 < 56 // VRAI</code> <code>55 < 55 // FAUX</code> <code>"Allo" < "Bonjour" // VRAI</code> <code>"4" < "100" // FAUX</code>
<=	Inférieure ou égale à	<code>23 <= 7 // FAUX</code> <code>42 <= 56 // VRAI</code> <code>55 <= 55 // VRAI</code> <code>"Allo" <= "Bonjour" // VRAI</code>

⇒ Les opérateurs de comparaisons ont pour résultat un booléen, donc la valeur VRAI ou FAUX

⇒ Les comparaisons fonctionnent seulement entre les données du même type

Comparaison

Comparaison	Définition	Exemple
>	Supérieure à	23 > 7 // VRAI 42 > 56 // FAUX 55 > 55 // FAUX "Allo" > "Bonjour" // FAUX
>=	Supérieure ou égale à	23 >= 7 // VRAI 42 >= 56 // FAUX 55 >= 55 // VRAI "Allo" >= "Bonjour" // FAUX

Comparaison

Comparaison	Définition	Exemple
==	Égale à	23 == 7 // FAUX 42 == 56 // FAUX 55 == 55 // VRAI "Allo" == "Bonjour" // FAUX
!=	Différente de	23 != 7 // VRAI 42 != 56 // VRAI 55 != 55 // FAUX "Allo" != "Bonjour" // VRAI

Entrées / sorties

- Les entrées / sorties sont réalisées avec les fonctions suivantes :
 - Pour lire une donnée sur le clavier de l'utilisateur :
 - Utiliser : « **<nomVariable>** = lire() »
 - Exemples :
 - `booléen possedeUnPermis = lire();`
 - `réel soldeBancaire = lire();`
 - Pour afficher des données à l'écran de l'utilisateur :
 - Utiliser : « écrire(<valeur>) » ou « écrireNL([<valeur>]) » (NL = Nouvelle ligne : ajoute un retour de chariot à la fin de l'affichage)
 - Exemples :
 - `écrireNL("Bonjour tout le monde");`
 - `écrireNL("Votre solde en banque : " + soldeBancaire.VersChaine());`

Générer un nombre aléatoire

- Si vous avez besoin de faire appel au hasard, c'est-à-dire de demander à la machine de choisir un nombre (lancer un dé) :
 - Utilisez : `générerNombre(<valeurMinIncluse>, <valeurMaxExcluse>)`
 - Exemple :
 - `entier nombreAleatoire = générerNombre(1, 6)`
 - `// après l'exécution de la ligne précédente, nombreAleatoire contiendra une valeur comprise entre les valeurs 1 et 5 incluses.`

Exemple complet

- Soit le problème suivant :
 - Écrire un algorithme qui lit deux réels et qui affiche le résultat de leur multiplication
- Démarche :
 - Quel est le problème à résoudre ?
 - Quelles sont mes données d'entrées ?
 - Deux réels : besoin de deux variables, une pour chaque opérande
 - Quel est le traitement ?
 - Deux lectures de réels : la lecture directe est un peu brute, nous allons aussi afficher un message à l'utilisateur de notre algorithme
 - Une multiplication de ces deux derniers : besoin d'une autre variable pour le résultat
 - Affichage du résultat

Exemple complet

```
réel operande1 = 0.0;  
réel operande2 = 0.0;  
réel resultatMultiplication = 0.0;
```

} Déclaration de 3 variables :

- une pour stocker chaque opérande
- une dernière pour stocker le résultat

```
écrire("Entrez un premier réel : ");  
operande1 = lire();  
écrire("Entrez un second réel : ");  
operande2 = lire();  
écrireNL();
```

} Pour chaque valeur à saisir (lire),

- affichage d'un message à l'utilisateur
- lecture de la valeur et affectation à la variable adéquat

```
resultatMultiplication = operande1 * operande2;
```

} Affectation du calcul à la variable
resultatMultiplication

```
écrire("La multiplication de " + operande1.VersChaine());  
écrire(" et de " + operande2.VersChaine() + " est : ");  
écrire(resultatMultiplication);
```

} Affichage du résultat accompagné d'un message