Les opérateurs numériques et les opérateurs de comparaison

Table des matières

I. Opérateur numérique en algorithmique

Contexte

L'algorithmique et les mathématiques sont des concepts dans la littérature scientifique. L'ère des Babyloniens, au III^{ème} millénaire av. J.-C., témoigne déjà de l'existence de la première utilisation des algorithmes dans la résolution des équations. Par définition, l'algorithmique est l'étude des règles et techniques pour la conception des algorithmes, en d'autres mots, c'est une science des algorithmes. Quant à l'algorithme, il se définit comme l'enchaînement des opérations pour résoudre un problème précis. Il doit son nom à son créateur, le mathématicien Al-Khawarizmi, qui a exercé ses talents entre l'an 813 et l'an 833. Il est important de savoir que les langages mathématiques et algorithmiques partagent beaucoup de points communs.

Bien que les ordinateurs soient des machines performantes pour résoudre des problèmes avec rapidité, pour traiter des calculs à grande échelle, les travaux de la littérature scientifique convergent vers une possibilité de trouver un terrain d'entente entre les mathématiques et l'informatique. Au cours de l'histoire, les mathématiciens ont fait des efforts pour trouver une solution afin d'optimiser les temps dans la résolution de problèmes nécessitant d'immenses calculs. C'est alors qu'une extension du langage mathématique a vu le jour vers 1843 et témoigne de l'évolution de la langue algorithmique. En 1843, la mathématicienne Ada Lovelace, spécialisée en sciences informatiques, a fait la première implémentation des algorithmes dans des machines informatiques. C'est de là que commence la traduction des langages mathématiques en langage informatique ou plutôt en langage algorithmique. Parlant alors de mathématiques, les opérations en font les pionniers de cette science. Pour implémenter les opérations dans des programmes informatiques, on doit les traduire en langage compréhensible pour un ordinateur. En algorithmique, les opérations mathématiques sont les opérateurs arithmétiques et les opérateurs de comparaison.

Les opérateurs peuvent varier en fonction de la langue de programmation informatique utilisée. D'ailleurs il est important de faire la différence entre langages algorithmiques et langages de programmation. Ceci dit, dans ce contenu nous allons nous focaliser sur le langage algorithmique.

Définition **Opérateurs en algorithmique**

Les opérateurs sont des symboles qui permettent d'exécuter des opérations, ou font office d'affectation ou de comparaison dans un programme informatique ou simplement algorithmique.

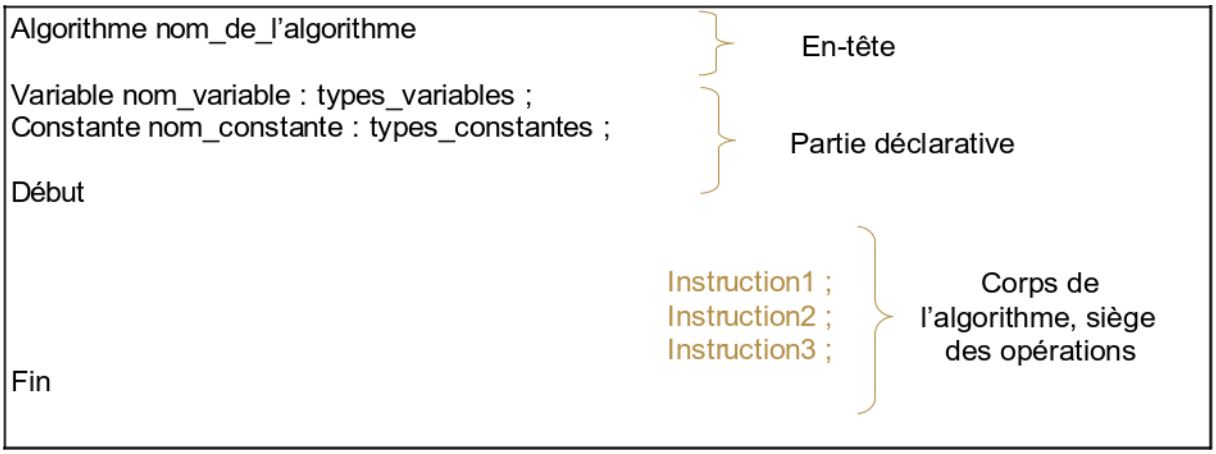
Dans le récit algorithmique on compte 5 catégories de ces types d'opérateurs :

- Opérateurs de calcul ou arithmétique
- Opérateurs de comparaison
- Opérateurs de concaténation
- Opérateurs logiques
- Opérateurs d'affectation

Il est à noter que les 3 opérateurs suivants sont des prérequis nécessaires pour faciliter la compréhension du cours. Des éléments de ces opérateurs seront évoqués souvent dans le contenu lorsqu'on va parler des opérateurs de comparaisons et arithmétiques :

- Opérateurs de concaténation
- Opérateurs logiques
- Opérateurs d'affectation

On a parlé récemment de correspondance de langage mathématique et de langage algorithmique ou informatique. L'implémentation algorithmique a en effet une légère particularité si on parle de syntaxe. Voilà pourquoi il est important de faire un rappel sur le corps d'un algorithme et sur les opérateurs.



Les variables tant en mathématiques qu'en informatique jouent un rôle important. Toutefois, la définition en informatique se décale légèrement de la définition en mathématiques. Les variables sont des entités, des noms que l'on associe à des valeurs, dont la valeur peut changer au cours des instructions dans un algorithme ou programme informatique.

Les types de variables regroupent les catégories suivantes :

- Entier : regroupent les entiers naturels : 1, 2, 15, 100, - 5, - 4, etc.
- Réel : ensemble des nombres réels : 12,2 ; - 5,2 ; etc.
- Chaine de caractère : correspond aux types de mots ou à une phrase.

  Exemples : « *résultats* », « *somme* », etc.
- Caractère : regroupe les types alphanumériques : lettres et symboles, ou autres.

  Exemples : « *D4* », « *V1* », etc.
- Booléen : seulement deux valeurs possibles « *vrai* » ou « *faux* ».

Voilà donc les types de variables que les opérateurs vont opérer et chaque type d'opérateur correspond à des variables spécifiques. Néanmoins, les opérateurs numériques, ou arithmétiques plus précisément, vont concerner les types de variables entier, réel et caractère, tandis que les opérateurs de comparaison vont tout englober.

A. Opérateurs arithmétiques

Fondamental

En ce qui concerne les opérateurs arithmétiques, il en existe au total 7 dans le langage algorithmique, dont certains vont être légèrement similaires à des opérateurs mathématiques que l'on a l'habitude de rencontrer en mathématiques.

+	Addition
-	Soustraction
*****	Multiplication
div	Division entière
mod	Reste de la division entière

^	Exponentiation ou puissance
/	Division

Tableau des opérateurs arithmétique

Remarque

Il est à noter que pour l'affectation des variables en opération algorithmique, on utilise l'opérateur d'affectation symbolisé par une flèche « ← ».

Par exemple si on veut affecter la valeur 4 dans A alors on a l'instruction suivante : *A ← 4*.

Addition et soustraction

Pour l'opérateur addition, le résultat obtenu sera la somme des deux variables réelles ou entières. Prenons par exemple l'instruction : *A ← 3 + 4*. La nouvelle valeur que la variable va prendre sera 7.

Pour la soustraction, le résultat obtenu sera la soustraction des deux variables. Bien évidemment on parle des valeurs numériques. D'autant plus qu'ils ont tous les deux le même ordre de priorité dans une opération. Par exemple, *C ← 43 + 5 - 3*, cela renvoie comme valeur 45.

Attention

Il ne faut pas confondre le « + » dans un opérateur de concaténation et le « + » pour un opérateur arithmétique. De plus, la concaténation agit sur des variables de type caractère ou chaîne de caractère, et fusionne les deux pour donner une phrase ou un mot généralement. Voyons cela avec un exemple :

Exemple

A ← matheB ← matiqueC ← A + B

La chaine de caractère « *mathématique* » sera affectée dans « *C* »

Multiplication

La multiplication n'est pas à confondre avec le symbole de multiplication en mathématique. En algorithmique ce dernier est symbolisé par « * » (un astérisque). Par exemple, si on veut calculer *D = 5 x 80* en syntaxe algorithmique cela donne : *D ← 5 * 80.* Ce qui renvoie la valeur 400.

Division

Il existe trois types de divisions comme opérateur arithmétique. La division entière, le reste de la division et la division. Le symbole « / » représente la division entre deux réels ou entiers. Le symbole « *div* » renvoie la division entière. Puis « *mod* » calcule le reste de la division de deux nombres.

Exemple

Instructions	Valeurs de A
A ←15 / 4	3,75
A ←15div4	3
A ←15mod4	3 : reste de la division de 15 par 4

La puissance ou l'exponentiation

La puissance est symbolisée par « ^ ».

Exemple

Voici un exemple d'algorithme pour mieux comprendre :

```
Algorithme exemple_puissance
Variables : x, cub : reel ;
Début
 Écrire('donner un réel x') ;
 Lire(x) ;
 cub=x^3 ;
 Ecrire('la valeur cubique de x est :', cub)
Fin
```

L'algorithme calcule la valeur cubique du réel que vous choisissez.

B. Ordre de priorité des opérateurs arithmétique

Les opérateurs, dans le langage algorithmique, sont une forme d'extension des opérateurs numérique en syntaxe mathématique. Cela veut dire que ces opérateurs arithmétiques n'échappent pas aux règles de priorités dans les enchaînements des opérations. Cette règle en algorithmique nous permettra de définir l'ordre d'exécution des opérations.

Dans une suite d'opérations algorithmiques, les parenthèses ont la priorité la plus forte, suivies des opérateurs multiplications et divisions. C'est après que les opérateurs additions et soustractions prennent place. Le tableau suivant présente de façon synthétique le résumé de ces informations :

Priorité 1	Parenthèse « *()* »
Priorité 2	« * », « / », « *div* », « *mod* » , « ^ »
Priorité 3	« + », « - »

Prenons les références des autres langages de programmation pour avoir un aperçu des syntaxes des opérateurs arithmétiques :

En langage Java

Sur Java, la syntaxe des opérateurs est la même à l'exception de l'opérateur de la division euclidienne ou « *mod* » en algorithme. En java il devient : « % ».

Exemple

```
Public class operateur_test{
Public static void main{
 Var a=9 ;
Var b=4 ;
opp=a+b ;# Addition
System.out.println(‘’a+b : ‘’ + opp)
opp=a-b ;# Soustraction
System.out.println(‘’a+b : ‘’ + opp)
opp=a*b ;#multiplication
System.out.println(‘’a*b : ‘’ + opp)
opp=a/b ;#division
System.out.println(‘’a/b : ‘’ + opp)
opp=a%b ; #reste de la division de a avec b ou mod
System.out.println(‘’a mod b : ‘’ + opp)
}
}
```

Langage C

Les syntaxes des opérateurs sont les mêmes en langage C qu'en Java. Le programme suivant vous aidera à avoir une idée des opérateurs en langage C.

```
Int main()
{
Int a= 9, b=6, opp ;
opp=a + b ;  // addition
printf(‘’a+b :  %d\n ‘’ , opp) ;
opp=a - b ; // soustraction
printf(‘’a-b :  %d\n ‘’ , opp) ;
opp=a*b ; // multiplication
printf(‘’a*b :  %d\n ‘’ , opp) ;
opp=a%b ; // reste de la division
printf(‘’a mod b :  %d\n ‘’ , opp) ;
Return 0 ;
}
```

Exercice : Quiz

[solution n°1 p.17]

Question 1

L'addition et la soustraction sont prioritaires dans une opération algorithmique.

- ○ Vrai
- ○ Faux

Question 2

On peut comparer un caractère avec une valeur numérique.

- ○ Vrai
- ○ Faux

Question 3

L'instruction suivante est correcte : *X ← (2 + 7 x 3) mod 4*.

- Vrai
- Faux

Question 4

L'instruction « *X ← imple + menter* » renvoie :

- La somme des caractères de « *imple* » avec « *menter* »
- Le mot « *implementer* » sera affecté aux variables X

Question 5

L'instruction suivante n'a pas d'erreur : $f \leftarrow 5^2 / 6 * 3$.

- Vrai
- Faux

III. Opérateur de comparaison

A. Comparaison des valeurs numériques

Dans les opérations en algorithmique, les nombres font partie des variables qui entrent en jeu dans l'exécution des enchaînements d'opérations. La syntaxe algorithmique n'échappe alors pas à la comparaison comme dans les mathématiques. D'autant plus que certaines opérations demandent une condition afin d'être exécutées et le résultat dépendra de ces conditions. Mais alors, quels sont les opérateurs de comparaison en algorithmique ? Le tableau suivant résume les différents opérateurs de comparaison en langage algorithmique.

>=	Supérieur ou égal
<=	Inférieur ou égal
>	Strictement supérieur
<	Strictement inférieur
==	Comparaison d'égalité
<> ou !=	Différent de

Pour établir une condition, le langage algorithmique possède des structures de conditions notées « *tant que* » et « *si* ». Ces commandes ou syntaxes font appel à des opérateurs de comparaison afin de conditionner les instructions des contenants.

La syntaxe « *tant que* » signifie plus simplement « *tant que la condition définie est encore respectée, ou vraie, si on fait allusion aux variables booléennes, les opérations doivent encore s'enchaîner* ». Cette syntaxe utilise toujours une initialisation et après parcourt une liste de variables jusqu'à ce que la condition ne soit plus respectée ou soit fausse. Au contraire, la syntaxe « *si* » affecte seulement une ou des conditions sur une des instructions.

Exemple

```
Algorithme exemple 1
 Variables n : entier ;
 n=0 ;
 Début
 Tant que n<=10
 Faire Somme=n+1 ;
 n=n+1 ;
 Fin tant que
 Fin
```

Cet algorithme calcule la somme des 10 premiers termes des nombres entiers naturels. On voit ici que l'opérateur conditionnel « <= » conditionne la valeur de n et agit sur l'ensemble des instructions.

```
Algorithme exemple 2
Variables a, b : entier ;
Début
 Si a!=0 alors x=-b\a ;
Fin siEcrire ('la solution de l'équation est est :', x)
Fin
```

Cet algorithme donne la solution de l'équation $ax + b = 0$ pour « a » différent de 0. En langage mathématique on a les instructions suivantes :

$$ax + b = 0, a \neq 0 \, x = -\frac{b}{a}$$

Les lignes de codes suivantes vous montreront les opérateurs de comparaison dans le langage C :

```
1  Int(main)
2  {
3  Int a = 8, b = 8
4  If ( a >b)
5  printf(‘’ a est supérieur à b\n ‘’) ;
6  else
7  if (a<=b)
8  printf(‘’ a est inférieur ou égal à b\n ‘’) ;
9  else
10 if (a>=b)
11                      printf(‘’ a est supérieur ou égal  à b\n ‘’) ;
12 else
13                     if(a !=b)
14 printf(‘’ a est différent de  b\n ‘’) ;
```

```
15 else
16 if(a==b)
17 printf(‘’ la  valeur  de a est égal à b\n ‘’) ;
18 return 0 ;
19 }
```

B. Comparaison des chaînes de caractères

Relations d'ordre des chaînes de caractère

Pour rappel, une chaîne est une suite de lettres ou bien de chiffres ou bien de caractères spéciaux. En faisant appel à la lexicographie, il est possible d'établir une relation d'ordre pour comparer les chaînes de caractère. En algorithmique, nous utilisons l'opérateur de comparaison pour comparer les chaînes de caractère.

Définition

L'ordre lexicographique est un concept dérivé de l'ordre alphabétique. Ce concept est utilisé pour comparer des chaînes et des caractères. Par définition notons A et B deux chaînes quelconques. On dit que A est lexicographiquement inférieur à B si et seulement si :

- A(i), avec i allant de 1 jusqu'à p,
- B(j), avec j allant de 1 jusqu'à q,
- i et j sont des caractères composant les chaînes A et B.

On a : p < q et A(i) = B(j) pour 1 < i < p et 1 < j < q sinon pour le plus petit i tel que A(i) est différent de B(i), A(i) < B(i).

Cela signifie que pour deux chaînes, on compare caractère par caractère. Si les premiers caractères comparés sont identiques on ne s'arrête pas tant qu'on n'a pas encore deux caractères différents et l'ordre des deux chaînes suivra l'ordre de ces deux caractères.

Exemple

- Si A = 223A et B = 223P alors A < B.
- Si A = baobab et B = banane alors A > B. Il suffit de comparer la lettre n et o (en position 3 de chacun des mots). La comparaison ne peut pas se faire sur les deux premiers caractères car ils sont identiques.
- Si A = baobab et B = salle alors A < B. Il suffit de comparer la $1^{ère}$ lettre.

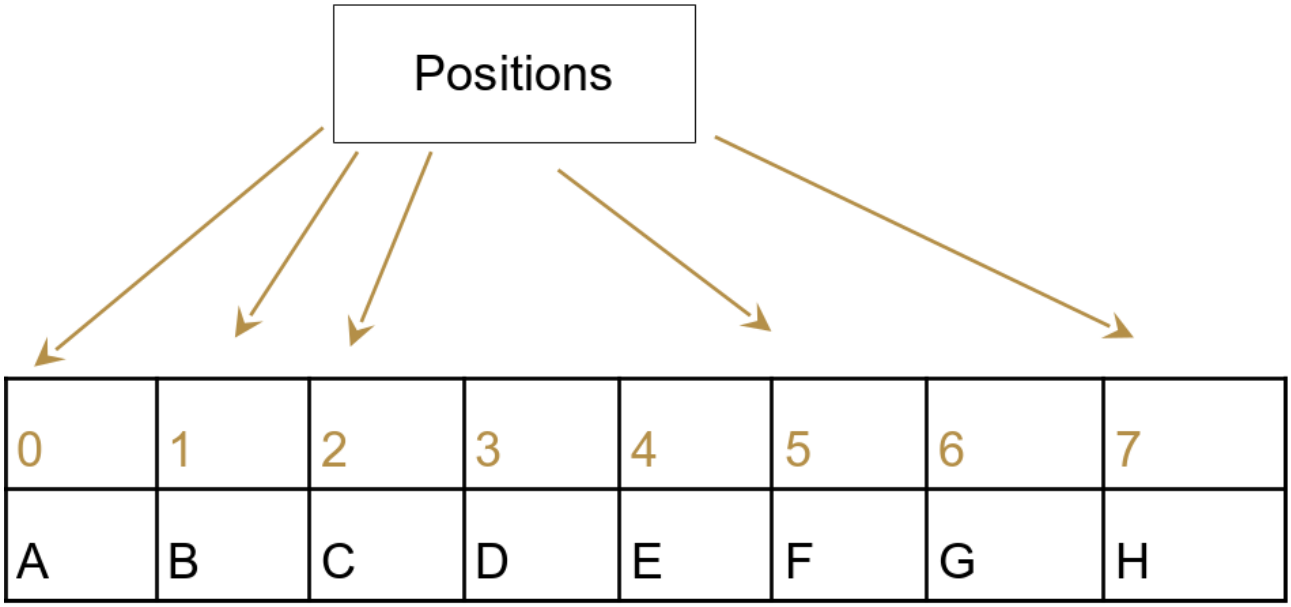
Remarque

Pour les caractères en majuscule et minuscule. Le plus grand sera considéré comme celui qui est minuscule. Plus généralement la comparaison en algorithmique fait toujours référence au code ASCII (*American Standard Code for Interchange Information*).

Il est également possible de mettre des conditions sur des chaînes de caractère. Cette dernière renvoie des booléens « *vrai* » ou « *faux* ». Si la condition qui a été posée est vraie alors la valeur renvoyée est « *vraie* » et inversement pour la valeur logique fausse.

Il est possible d'accéder à un caractère d'une chaîne : il est important de connaître certaines commandes qui nous permettront de manipuler les chaînes de caractères. Il existe alors la syntaxe « *[]* » qui permet d'accéder à un caractère spécifique dans une chaine de caractère.

Il est également possible de calculer la longueur d'une chaîne. Cela s'obtient par la commande : « *longueur(0)* ».

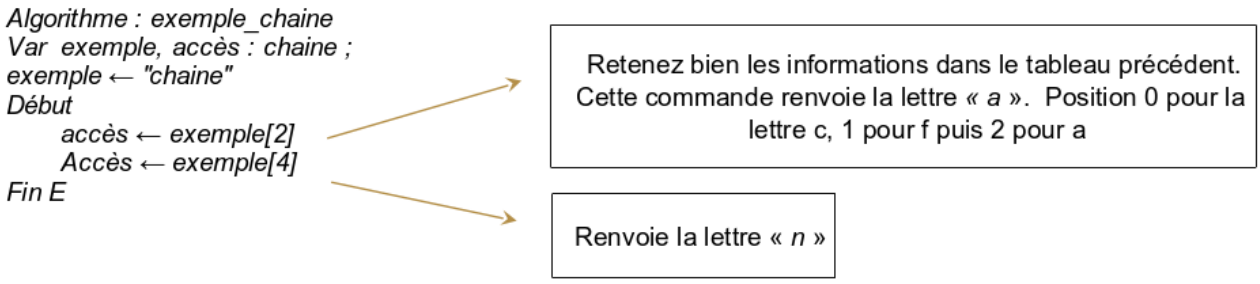


Prenons un exemple pour mieux comprendre : prenez le temps de bien assimiler ce tableau.

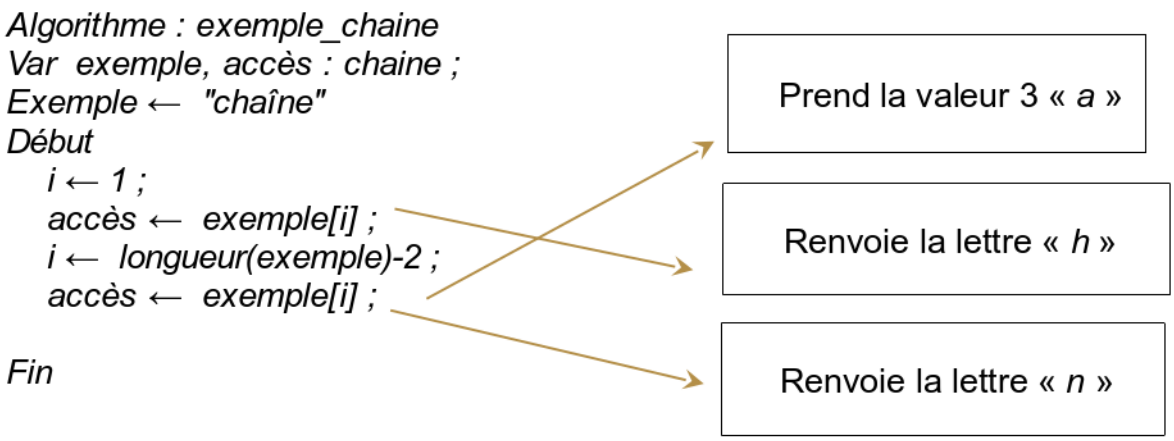
0	1	2	3	4	5
C	h	a	i	n	e

Dans un tableau, on part de l'index 0 et on incrémente de 1 à chaque fois. Donc dans ce tableau, l'index 0 correspond à la lettre c, l'index 1 à la lettre h, le 2 à la lettre a… Les index se notent entre [].

Soit l'algorithme suivant :



Un autre exemple en utilisant la commande « *longueur()* » :



On peut expliquer ici que la valeur de i est initialement 1.

Ensuite la commande `accès ← exemple[i]` renvoie la lettre « *h* », car comme i=1 c'est l'index 1 qui est renvoyé.

La valeur de i change, car elle a été affectée par la valeur de la longueur de « *chaîne* » donc 6 moins 2, ce qui donne 4.

Et donc « *accès* » prend la lettre « *n* » en dernier, car elle est en index 4.

Par défaut, les caractères de la chaîne « *chaîne* » sont étiquetés par des numéros : 0 pour la première lettre, 1 pour la deuxième et ainsi de suite. On peut constater ici que la variable i contient le chiffre 1 initialement. La commande *exemple[i]* consiste à accéder à la lettre étiquetée par la position « *1* » de la chaîne « *chaîne* » qui est alors la lettre « *h* » à compter de la position « *0* » de la lettre « *c* ». La variable i va maintenant changer de valeur. La valeur qui va être affectée à « *i* » sera la longueur de la chaîne « *chaîne* » soustrait par deux. N'oubliez pas, la longueur d'une chaîne est le nombre de caractères qui la compose, ici pour « *chaîne* » la longueur est donc 6 car elle est composée de six caractères. Attention à ne pas confondre strictement avec les étiquettes des positions des caractères dans une chaîne pour le décompte. « *i* » va donc prendre comme valeur 4. Si vous regardez le tableau ci-dessus, la lettre n est étiquetée par le numéro 4. Ce qui veut dire finalement que « *n* » va être affecté dans la variable « *accès* ».

Voilà donc deux exemples permettant de comprendre les deux commandes « *longueur()* » et la commande « *[]* ». Ces notions sont essentielles pour la comparaison des chaînes de caractères.

En résumé, les opérateurs relationnels opèrent également dans les caractères ainsi que dans les chaînes de caractères. Le fond de la comparaison réside sur l'ordre lexicographique des caractères à comparer en question.

Complément

Efface()	Efface un caractère dans une chaîne
Longueur()	La syntaxe donne la longueur de la chaîne
C[j]	Accède au caractère de position j dans une chaîne

Exercice : Quiz

[solution n°2 p.17]

Question 1

Soit l'algorithme suivant :

```
Algorithme quizz
```

```
Variable n : entier
Début
n ← 0
Ecrire('Ecrire un nombre entre 0 et 2')
Tant que n<0 ou n>2
Lire(n) ;
Si n<0 ou n>2 alors
Ecrire('Nombre non compris entre 0 et 2, donnez un autre nombre') ;
Finsi
Fintantque
Fin
```

L'algorithme consiste à :

- Déterminer si le nombre n introduit est supérieur à 2 ou inférieur à 0
- Demander à l'utilisateur un nombre n jusqu'à ce que le nombre choisi soit la bonne réponse

Question 2

La commande C[j], « *C* » étant une chaîne, renvoie la longueur de la chaîne.

- Vrai
- Faux

Question 3

Les deux symboles « = » et « == » sont deux opérateurs de même usage.

- Vrai
- Faux

Question 4

Le symbole « >= » est un opérateur relationnel qui signifie :

- Supérieur ou égal
- Strictement supérieur
- Égalité

Question 5

Les parenthèses sont de priorité absolue dans une instruction.

- Vrai
- Faux

V. Essentiel

Concernant les opérateurs arithmétiques, l'addition et la soustraction ont les mêmes syntaxes en mathématiques et en informatique. Il faut savoir que le « + », lorsqu'il s'agit d'une variable chaîne ou chaîne de caractère, fusionne les caractères ou les chaînes de caractères. Il s'agit dans ce cas d'un opérateur de concaténation. La multiplication se différencie de la syntaxe mathématique, en algorithmique elle est symbolisée par « * ». La division se catégorise en trois types. La division entière « *div* », la division décimale « / », la division euclidienne ou bien le reste de la division de deux entiers « *mod* ».

Concernant les opérateurs de comparaison, la syntaxe de l'inégalité stricte est de même en mathématique qu'en informatique. Pour la différence (deux nombres différents), elle est symbolisée par « *!=* ». L'opérateur de comparaison d'égalité est symbolisé par une concaténation de deux signes d'égalité, c'est-à-dire « == ». À ne pas confondre avec « = » qui est un opérateur d'affectation.

VI. Auto-évaluation

A. Exercice

Question 1

[solution n°3 p.19]

Traduisez en langage algorithmique l'instruction suivante, puis affectez à x la valeur de la division entière de 5 par 4 et écrivez les lignes d'instruction permettant de calculer la valeur de f.

$$f = \frac{2x}{(1-x)^3} + 3$$

Question 2

[solution n°4 p.19]

Écrire un algorithme qui demande un nombre qui calcule l'inverse de ce nombre.

Question 3

[solution n°5 p.19]

Traduire les phrases suivantes en langage algorithmique :

Phrase 1 : « *Si n est plus grand que p, alors faire la division euclidienne de n par p, par contre, si n est égal à p, afficher comme résultat « opération impossible »* ».

Phrase 2 : « *Tant que p est différent de q, faire la division entière de p par q ; si p est pair, faire la soustraction de p et 2 ; si p est impair, faire l'addition de p et 2* ».

B. Test

Exercice 1 : Quiz

[solution n°6 p.20]

Question 1

L'algorithmique est :

- O L'ensemble des procédures pour résoudre un problème précis
- O L'ensemble des règles et techniques pour la conception d'un algorithme

Question 2

<, > et != sont des :

- O Opérateurs relationnels
- O Opérateurs logiques
- O Opérateurs arithmétiques

Question 3

« ← » et « = » sont des opérateurs d'affectation de variables.

- O Vrai
- O Faux

Question 4

Le symbole « + », lorsqu'il s'agit de chaîne de caractères, renvoie à la somme des longueurs des chaînes.

- O Vrai
- O Faux

Question 5

L'opérateur « *div* » est un opérateur :

- O Arithmétique qui renvoie la division exacte
- O Relationnel
- O Arithmétique qui renvoie la division entière

Solutions des exercices

Exercice p. 7 Solution n°1

Question 1

L'addition et la soustraction sont prioritaires dans une opération algorithmique.

- ○ Vrai
- ◉ Faux

Q En règle de priorité la soustraction et l'addition sont en dernières places.

Question 2

On peut comparer un caractère avec une valeur numérique.

- ○ Vrai
- ◉ Faux

Q Il est de coutume de voir des nombres affectés à des variables. Cependant la comparaison entre un caractère et une valeur numérique n'est pas possible.

Question 3

L'instruction suivante est correcte : *X ← (2 + 7 x 3) mod 4*.

- ○ Vrai
- ◉ Faux

Q L'opérateur de multiplication « *x* » n'est pas valide dans un langage algorithmique.

Question 4

L'instruction « *X ← imple + menter* » renvoie :

- ○ La somme des caractères de « *imple* » avec « *menter* »
- ◉ Le mot « *implementer* » sera affecté aux variables X

Q Il y a une implémentation aux variables X et le « + » indique ici qu'un opérateur de concaténation est dans cette instruction.

Question 5

L'instruction suivante n'a pas d'erreur : $f \leftarrow 5^2 / 6 * 3$.

- ○ Vrai
- ◉ Faux

Q Si vous regardez dans cette instruction, il a une erreur, la puissance devrait être écrite avec le symbole « ^ » en algorithmique.

Exercice p. 12 Solution n°2

Question 1

Soit l'algorithme suivant :

```
Algorithme quizz
Variable n : entier
Début
n ← 0
Ecrire('Ecrire un nombre entre 0 et 2')
Tant que n<0 ou n>2
Lire(n) ;
Si n<0 ou n>2 alors
Ecrire('Nombre non compris entre 0 et 2, donnez un autre nombre') ;
Finsi
Fintantque
Fin
```

L'algorithme consiste à :

- ○ Déterminer si le nombre n introduit est supérieur à 2 ou inférieur à 0
- ◉ Demander à l'utilisateur un nombre n jusqu'à ce que le nombre choisi soit la bonne réponse

La boucle « *tant que* » émet une condition sur le nombre choisi et affiche comme résultat que le nombre ne convient pas, c'est-à-dire n'est pas compris entre 0 et 2.

Question 2

La commande C[j], « *C* » étant une chaîne, renvoie la longueur de la chaîne.

- ○ Vrai
- ◉ Faux

La commande « *longueur(C)* » sert à déterminer la longueur d'une chaîne de caractères.

Question 3

Les deux symboles « = » et « == » sont deux opérateurs de même usage.

- ○ Vrai
- ◉ Faux

Il faut faire très attention, car « = » est l'opérateur d'affectation tandis que « == » est un opérateur de comparaison d'égalité.

Question 4

Le symbole « >= » est un opérateur relationnel qui signifie :

- ◉ Supérieur ou égal
- ○ Strictement supérieur
- ○ Égalité

« >= » est l'opérateur relationnel qui signifie supérieur ou égal.

Question 5

Les parenthèses sont de priorité absolue dans une instruction.

- (•) Vrai
- ( ) Faux

Cette règle n'échappe pas aux instructions algorithmiques et suit les mêmes règles qu'en langage mathématique.

p. 14 **Solution n°3**

En écriture algorithmique, faisant appel aux opérateurs arithmétiques on a :

$f = (2 * x) / (1 - x)^{\wedge 3} + 3$

L'opérateur de la division entière est le « *div* ». On a alors :

$x \leftarrow 5 \text{ div } 4$

$f \leftarrow (2 * x) / (1 - x)^3 + 3$

p. 14 **Solution n°4**

```
Algorithme inverse
Variables x, inv : reel;
Début
 Écrire('donner un réel');
 Lire(x) ;
 inv ← 1/x ;
 Écrire( l'inverse de ',x,'est',inv);
fin
```

p. 14 **Solution n°5**

Phrase 1 : La traduction en pseudo-code est la suivante :

Si n>p alors

D=n mod p ;

Si n== p alors

Ecrire (''*opération impossible*'')

Phrase 2 :

Tant que p !=q faire

D=p div q

Si p mod 2=0 alors

D=p-2 ;

Si p mod 2 !=0 alors

D=p+2

Exercice p. 14 Solution n°6

Question 1

L'algorithmique est :

- ○ L'ensemble des procédures pour résoudre un problème précis
- ◉ L'ensemble des règles et techniques pour la conception d'un algorithme

Il y a une grande différence entre algorithme et algorithmique. Les algorithmes sont des ensembles de procédures figurant des solutions pour un problème. L'algorithmique est l'étude des règles pour le concevoir.

Question 2

<, > et != sont des :

- ◉ Opérateurs relationnels
- ○ Opérateurs logiques
- ○ Opérateurs arithmétiques

Ces symboles sont des opérateurs relationnels pour comparer une différence.

Question 3

« ← » et « = » sont des opérateurs d'affectation de variables.

- ◉ Vrai
- ○ Faux

En effet, ce sont des opérateurs dans la catégorie d'affectation de variable, à ne pas confondre avec « == ».

Question 4

Le symbole « + », lorsqu'il s'agit de chaîne de caractères, renvoie à la somme des longueurs des chaînes.

- ○ Vrai
- ◉ Faux

L'opérateur « + », lorsqu'il s'agit de chaîne de caractères, est un opérateur de concaténation et non la somme de la longueur des chaînes.

Question 5

L'opérateur « *div* » est un opérateur :

- ○ Arithmétique qui renvoie la division exacte
- ○ Relationnel
- ◉ Arithmétique qui renvoie la division entière

Le « *div* » fait partie des opérateurs arithmétiques de division. Le résultat est toujours la partie entière de la division.