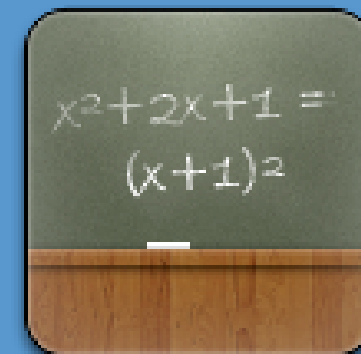




Programmation



Algorithmie

Module 1 Partie 2

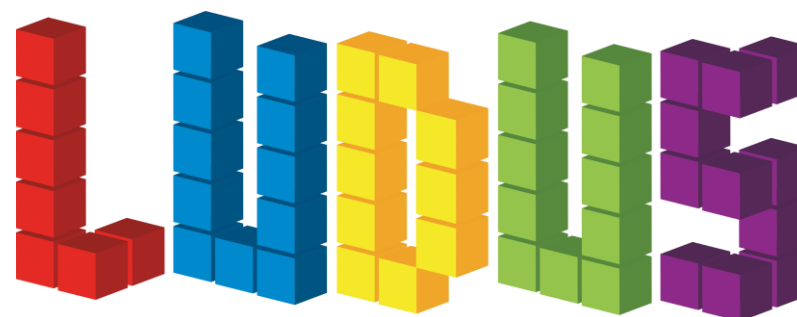


Cursus

1

Level

1



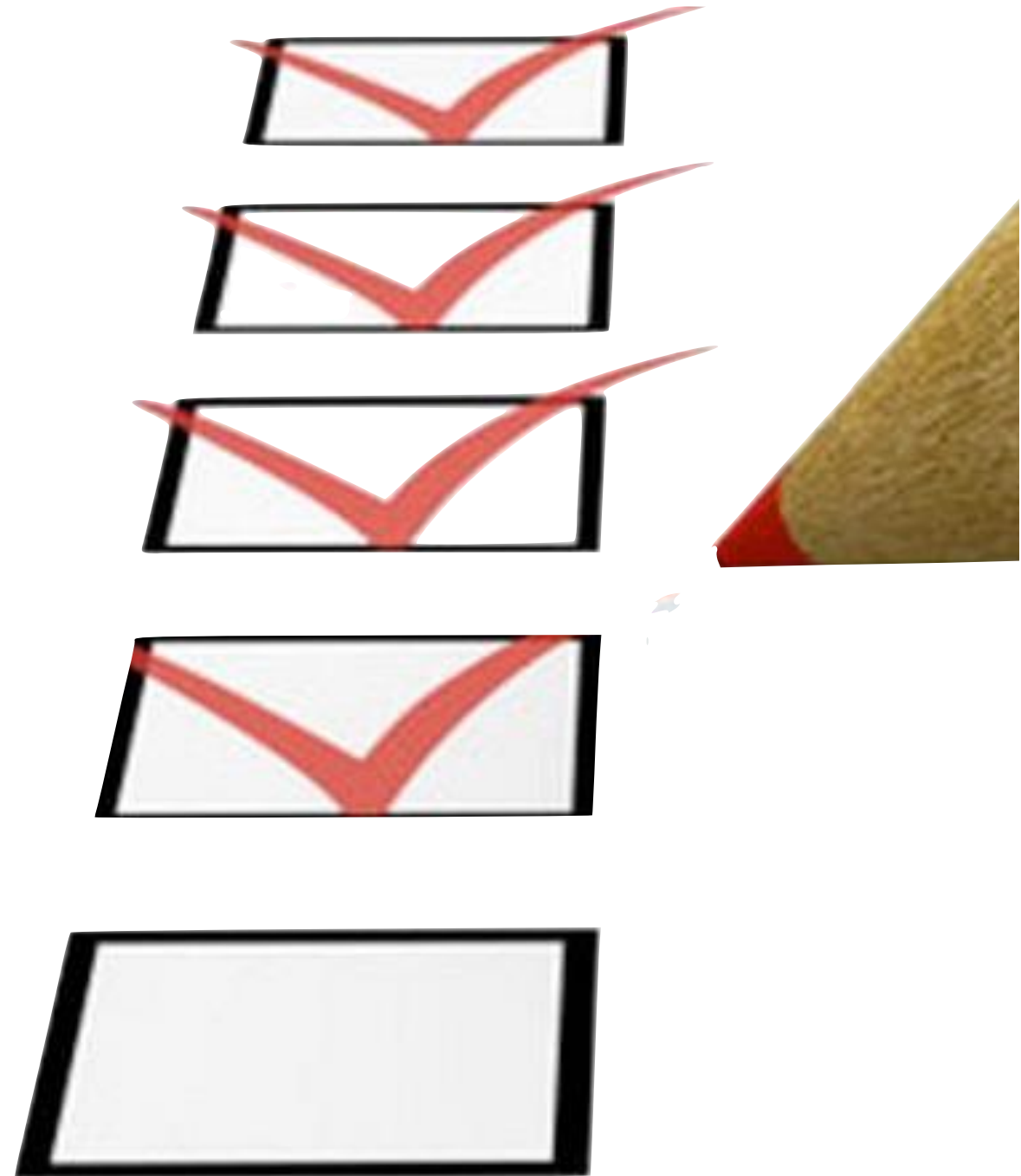
ACADÉMIE





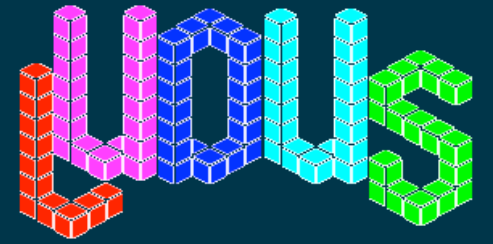
Objectifs

- Aborder la programmation.
Présentation d'un langage universel.
- Organiser un programme.
- Utiliser des données élémentaires.
- Structurer les étapes de résolution d'un problème.





Sommaire



- ✳️ Présentation
- ✳️ Structure de données.
- ✳️ Structure algorithmique.
- ✳️ Structure de programme.

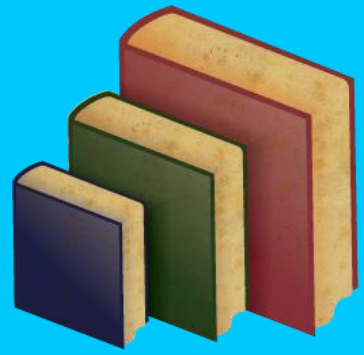




Références Internet

Introduction à l'algorithmique et à la programmation

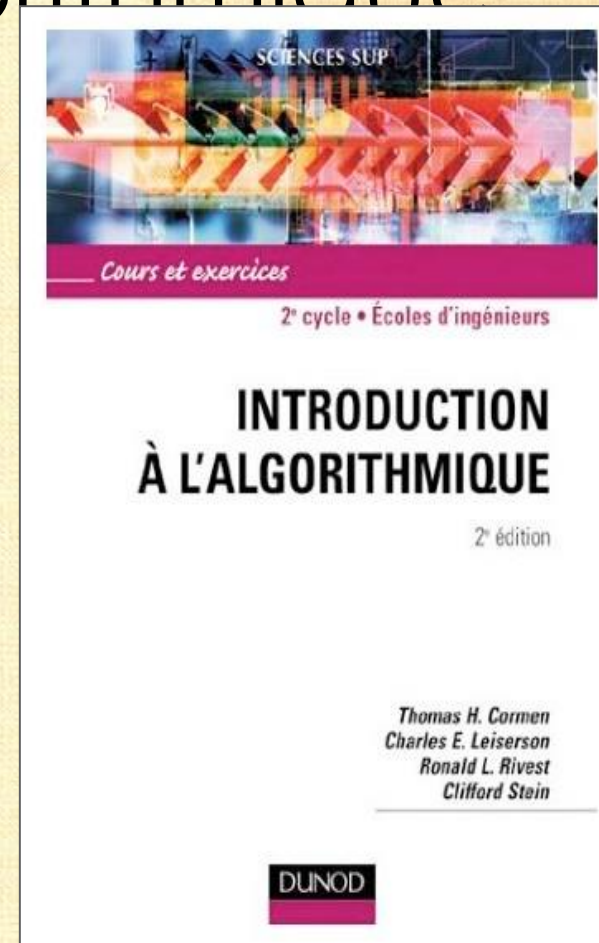
<http://perso.citi.insa-lyon.fr/afraboul/imsi/algo-imsi-4.pdf>

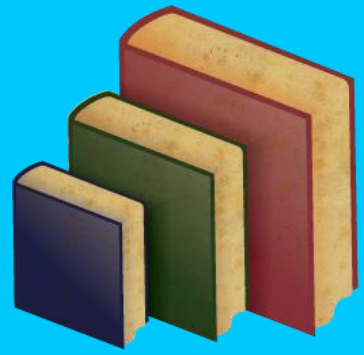


Références Bibliographiques

Introduction à l'algorithmique :

Cours et exercices
de Cormen, Leiserson, Rivest

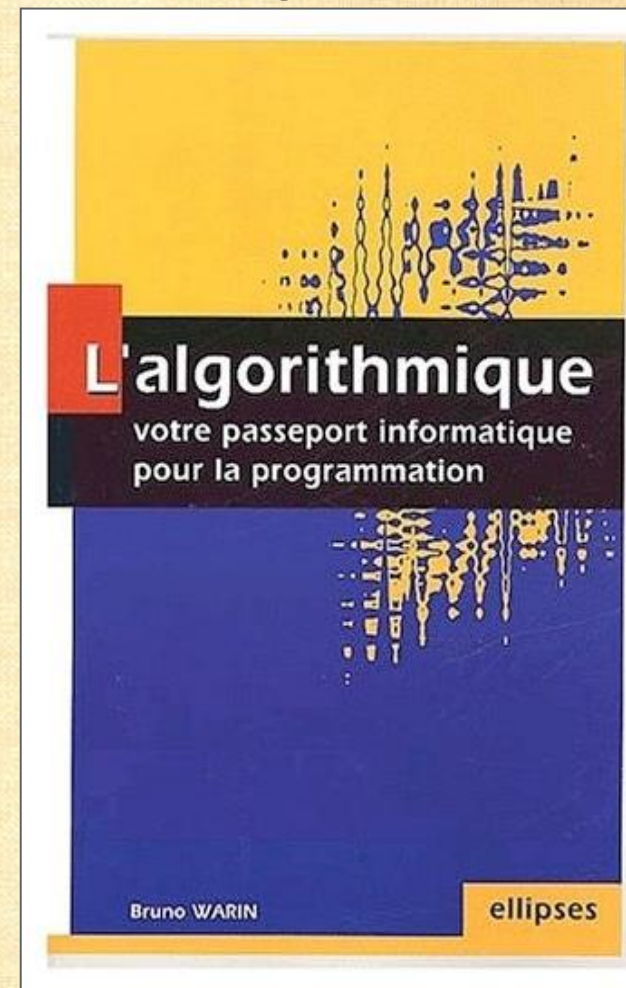




Références Bibliographiques

L'algorithmique :

Votre passeport informatique pour la programmation
de Bruno Warin





Liens pédagogiques



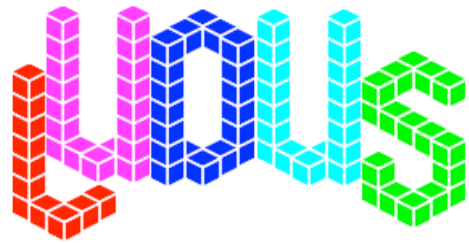
Pré-requis:

Historique et modèle de Von Neumann



Nécessaire pour:

Réaliser tout programme.



Algorithmie

Partie N°2

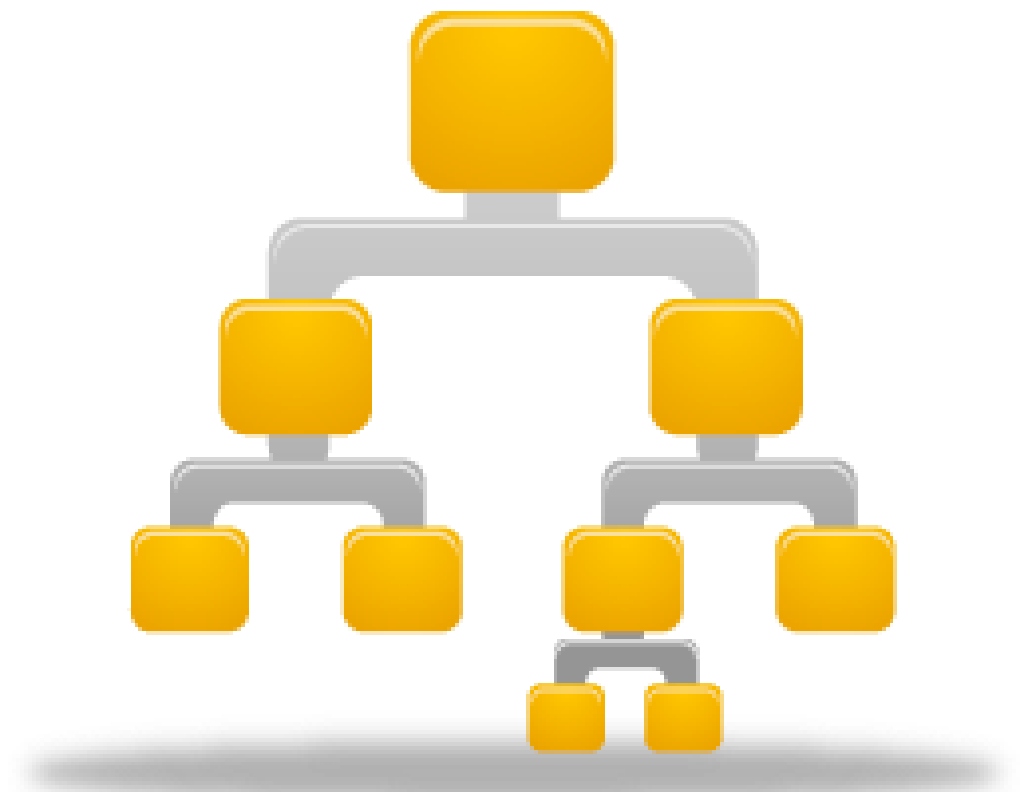
Structure de données



Algorithmie

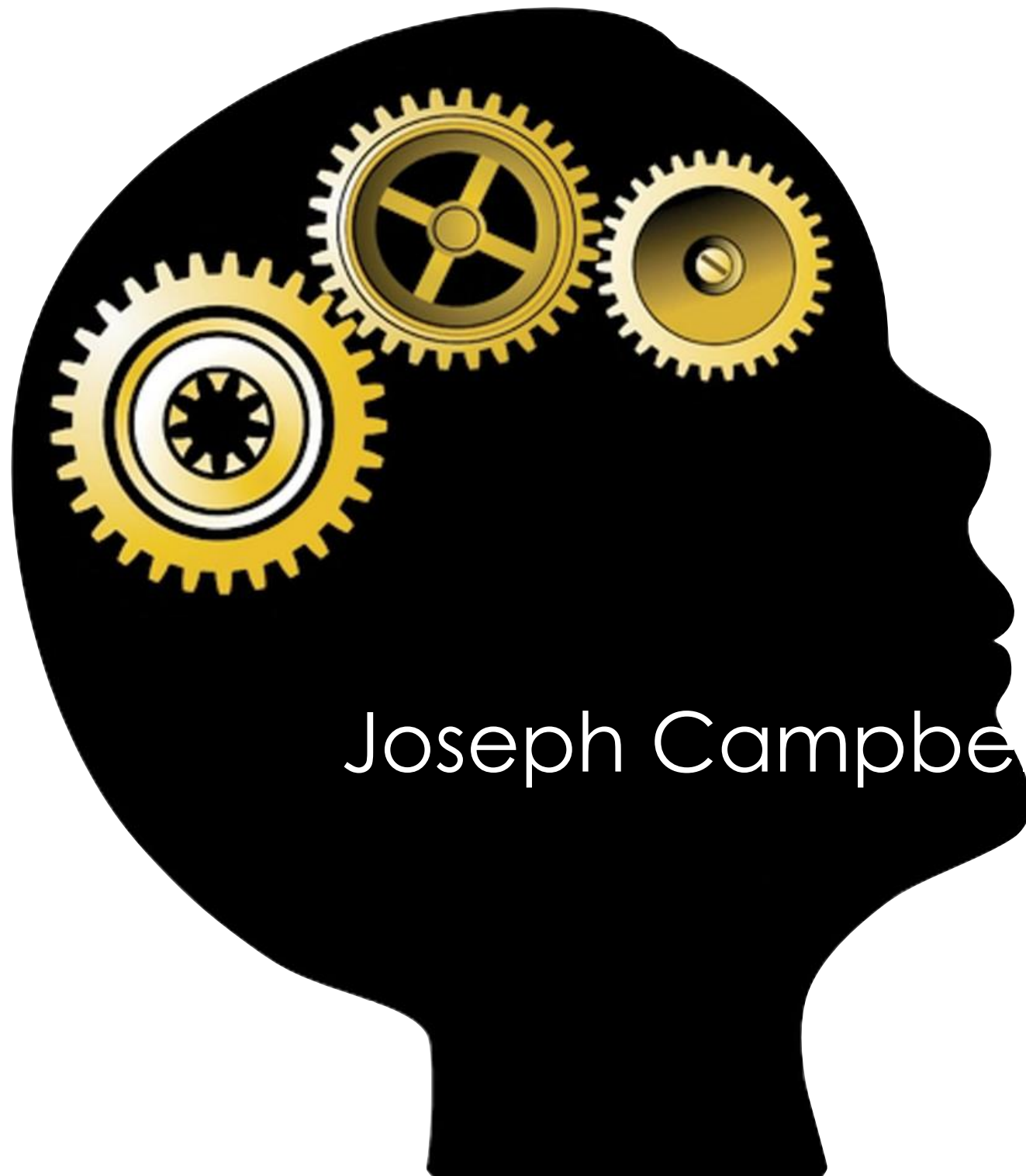
Structure de données

- Présentation
- Les 5 structures de données élémentaires
- Opérateurs et symboles associés
- Opérations élémentaires
- Les structures de données avancées





Citation



Joseph Campbe

«Les ordinateurs sont
comme les dieux de
l'Ancien Testament :
avec beaucoup de
règles, et sans pitié.»

Algorithmie > Structure de données

Notion de variables (Rappel)

Les types élémentaires possibles sont:

- ✱ Les entiers
- ✱ Les réels
- ✱ Les caractères
- ✱ Les chaînes de caractères
- ✱ Les booléens

Algorithmie > Structure de données

Les entiers

ENTIER

 nombres entiers

 négatifs

 positifs

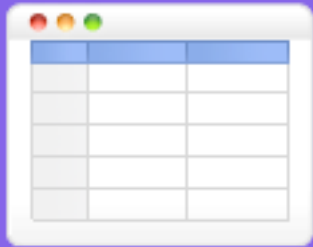
 nuls

Exemples :

 33

 - 13

 [pas de limite de taille]



Entier

Opérations possibles

Addition
Soustraction
Multiplication
Division
Exposant
Modulo
Comparaisons

Opérateurs associés, symboles, mot clés correspondants

+

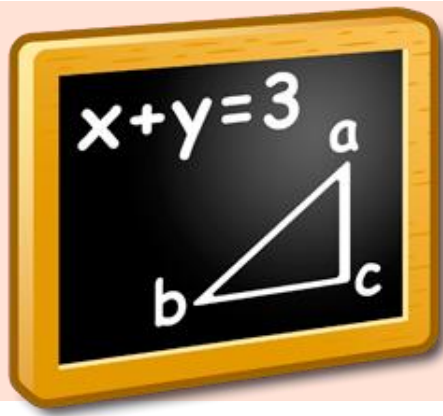
-

DIV (Pour la division entière)

^

MOD (Pour le reste d'une division)

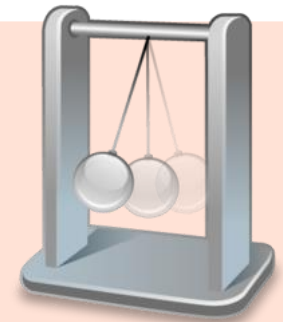
< <= > >= = <>



Exercice



Ecrire un algorithme qui demande à un utilisateur un reel et affiche sa partie entière



2 min



solo



Live

**ALGORITHME : reel2entier**

```
//ENTREE : un réel saisi par l'utilisateur
```

```
//SORTIE : la partie entière
```

```
VAR :      reel2ent      : ENTIER
      nb1                : REEL
```

LIRE nb1

```
reel2ent ← nb1 DIV 1
```

ECRIRE reel2ent

FIN

Algorithmie > Structure de données

Les réels

REEL

 nombres entiers

 fractionnés

 négatifs

 positifs

 nuls

Exemples :


 3,26

 -10,5

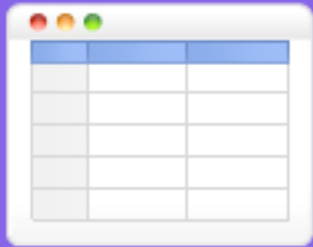
 [pas de limite de taille]



Conseil



En écriture algorithmique on utilise la virgule et non le point pour noter la partie décimale.



Réél

Opérations possibles

Addition
Soustraction
Multiplication
Division
Exposant
Comparaisons

Opérateurs associés, symboles, mot clés correspondants

+
-

/
^
< <= > >= = <>

Algorithmie > Structure de données

Les caractères

CARACTERE (ou CAR)

caractère unique

Exemples :

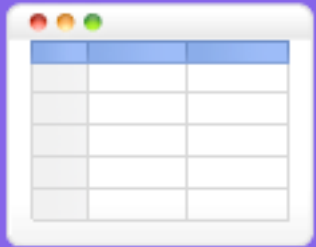
'F'

'j'

'!'

[référence : table des codes ASCII, Unicode]

apostrophe simple



Caractère

Opérations possibles

**Opérateurs associés, symboles,
mot clés correspondants**

Comparaisons
Extraction

< <= > >= = <>

CAR1 ← EXTRACTION (chaine, position)

Algorithmie > Structure de données

Les chaînes

CHAINE

caractère unique

suite de caractères

Exemples :

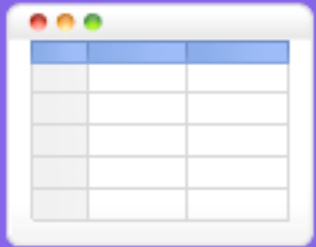
"Bonjour"

"64"

"Super Mario World"

[référence : table des codes ASCII, Unicode]

apostrophe double



Chaîne

Opérations possibles

Comparaisons
Extraction
Longueur
Concaténation

Opérateurs associés, symboles, mot clés correspondants

< <= > >= = <>

CH1 ← EXTRACTION (CH2, position, longueur)

ent ← LONGUEUR (ch)

ch ← CONCATENER (ch1, ch2)

ou ch ← ch1 + ch2



Conseil

Dans une fonction de chaîne de caractères, un caractère peut être vue comme une chaîne de longueur 1.

Ainsi

`chr ← ch2 + '.'`

est valide.

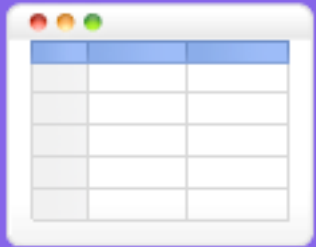
Algorithmie > Structure de données

Les booléens

BOOLEEN

vrai

faux



Booléen

Opérations possibles

Opérateurs associés, symboles, mot clés correspondants

Comparaisons

= <>

Négation

NON

Conjonction

ET

Disjonction

OU



Solution



ALGORITHME : testbool

//Exo purement virtuel

VAR : a,b,c,d,e,f

: BOOLEEN

DEBUT

a ← Vrai

b ← Faux

c ← a ou b

d ← a et b

e ← non c ou non d ou a et b

f ← non a ou non b et a ou non a

Ecrire a,b,c,d,e,f

FIN

Algorithmie > Structure de données

Autres structure de données

Structures linéaires :

- Tableaux

- Piles

- Files

- Listes

Structures non linéaires :

- Enregistrements

- Graphes

- Arbres

Algorithmie > Structure de données

Quelle structure de données ?

"Validez"

FAUX

1024

'o'

9,81

ENTIER

CHAINE

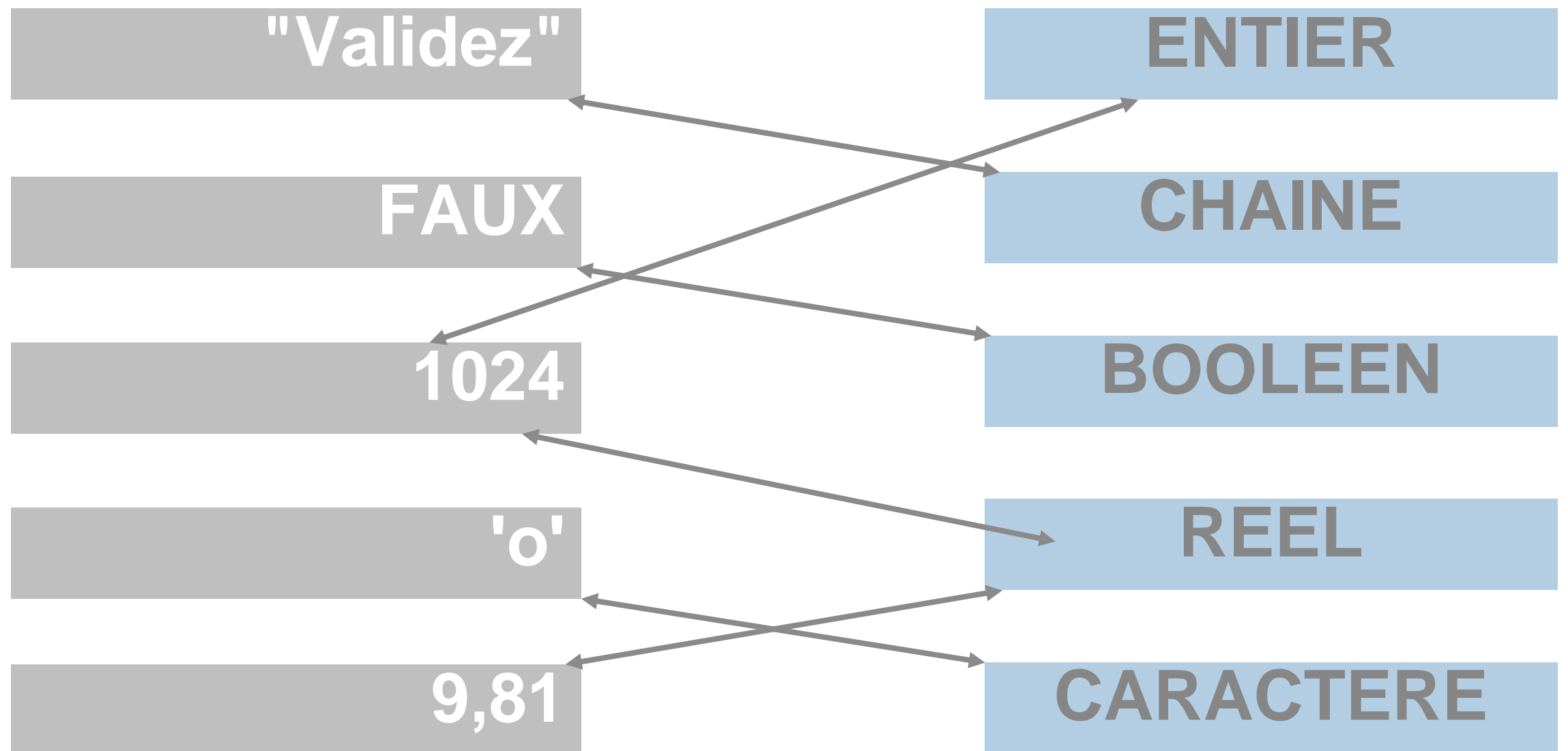
BOOLEEN

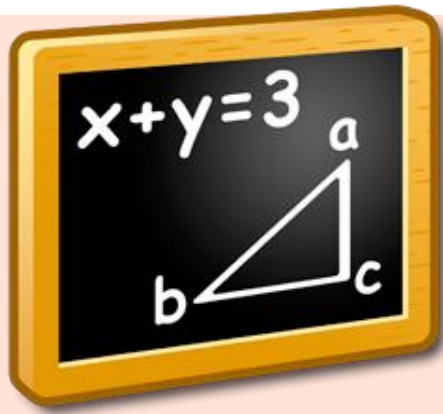
REEL

CARACTERE

Algorithmie > Structure de données

Quelle structure de données ?

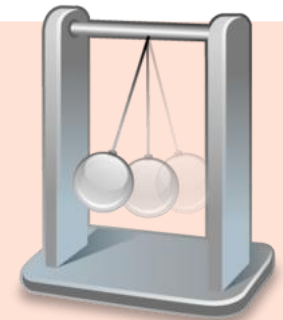




Exercice



Ecrire un algorithme qui demande à un utilisateur son nom et son prénom et qui affiche ses initiales.



5 min



solo



Live



Solution



ALGORITHME : Initiales

//BUT : cet algorithme extrait les initiales d'une personne

//ENTREE : Le nom et le prénom saisis par l'utilisateur

//SORTIE : les initiales

VAR : nom, prenom, initiales: **CHAINE**

DEBUT

 ECRIRE "Veuillez entrer votre nom"

 LIRE nom

 ECRIRE "Veuillez entrer votre prénom"

 LIRE prenom

 initiales ← EXTRACTION(nom,1,1) + EXTRACTION(prenom,1,1)

 ECRIRE initiales

FIN



Solution



ALGORITHME : Initiales

//BUT : cet algorithme extrait les initiales d'une personne

//ENTREE : Le nom et le prénom saisis par l'utilisateur

//SORTIE : les initiales

VAR : nom, prenom, initiales: **CHaine**

DEBUT

 Ecrire "Veuillez entrer votre nom"

 Lire nom

 Ecrire "Veuillez entrer votre prénom"

 Lire prenom

 initiales ← EXTRACTION(nom+' ',1,1) + EXTRACTION(prenom+' ',1,1)

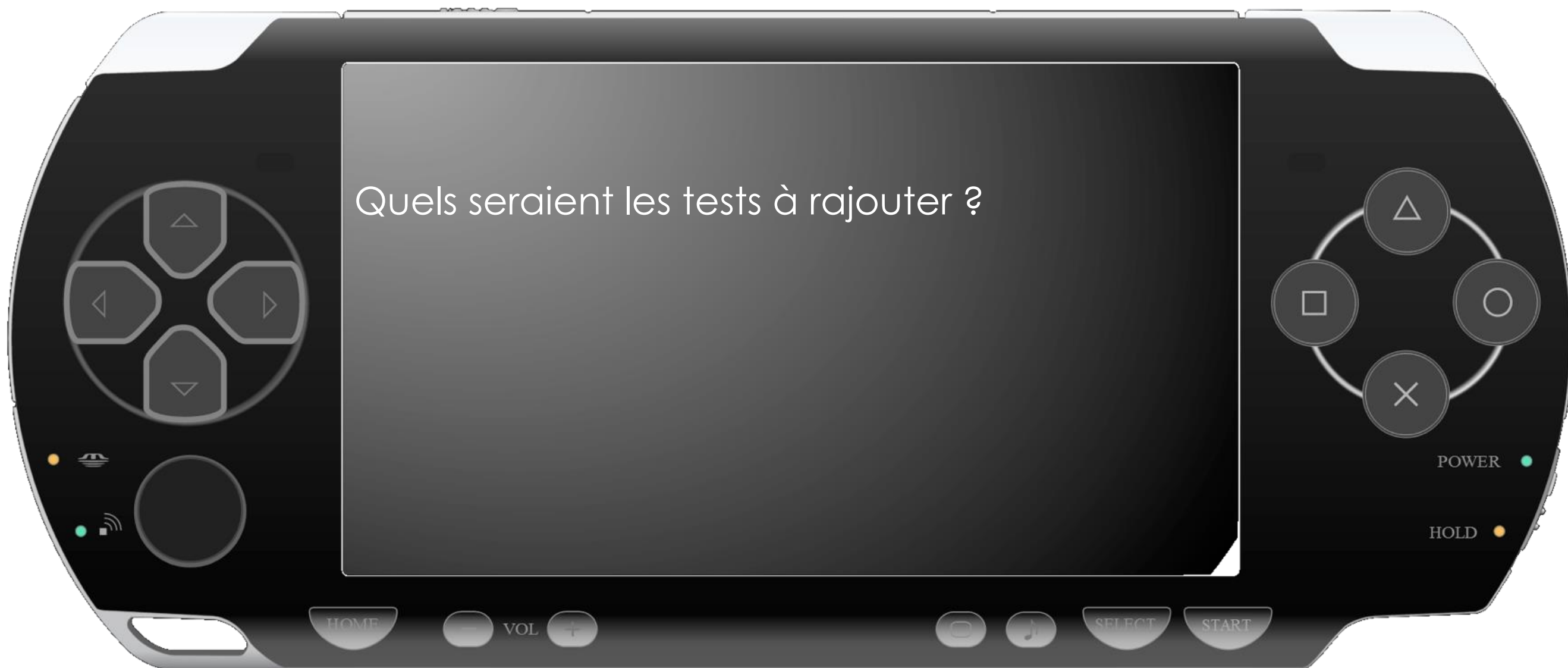
 Ecrire initiales

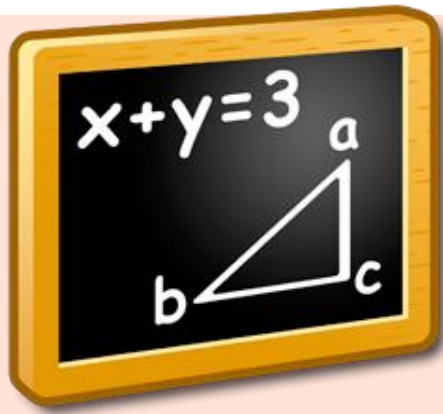
FIN



Question

Quels seraient les tests à rajouter ?





Exercice

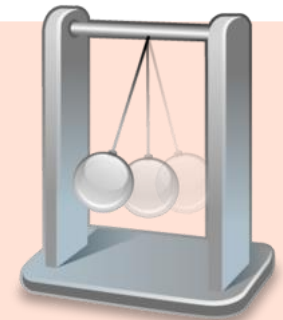


Ecrire un algorithme qui demande à un utilisateur son nom et son prénom et qui affiche son adresse mail Ludus:

Jean

Martin

=> j.martin@ludus-academie.com



5 min



solo



Live



Solution



ALGORITHME : Initiales

//BUT : cet algorithme extrait les initiales d'une personne

//ENTREE : Le nom et le prénom saisis par l'utilisateur

//SORTIE : les initiales

VAR : nom, prenom, mail: **CHaine**

DEBUT

 ECRIRE "Veuillez entrer votre nom"

 LIRE nom

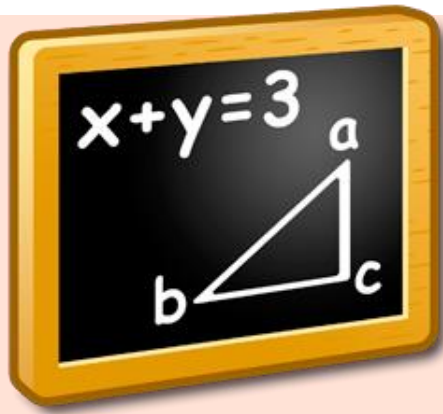
 ECRIRE "Veuillez entrer votre prénom"

 LIRE prenom

 mail ← EXTRACTION(prenom+' ',1,1) + '.' +nom + '@ludus-academie.com'

 ECRIRE mail

FIN

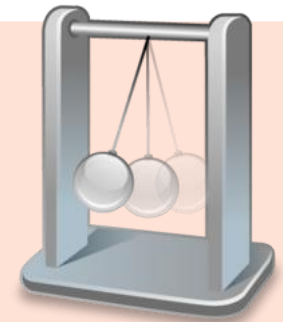


Exercice



Ecrire un algorithme qui demande à un utilisateur un mot de trois lettres et affiche son miroir.

ex: ABC devient CBA



5 min



solo



Live



Solution



ALGORITHME : Miroir

//BUT : cet algorithme inverse un mot de trois lettres

//ENTREE : Le mot de trois lettres saisi par l'utilisateur

//SORTIE : le mot miroir

VAR : mot, motmiroir: **CHaine**

DEBUT

 ECRIRE "Veuillez entrer votre mot"

 LIRE mot

 motmiroir ← EXTRACTION(mot,3,1) + EXTRACTION(mot,2,1) + EXTRACTION(mot,1,1)

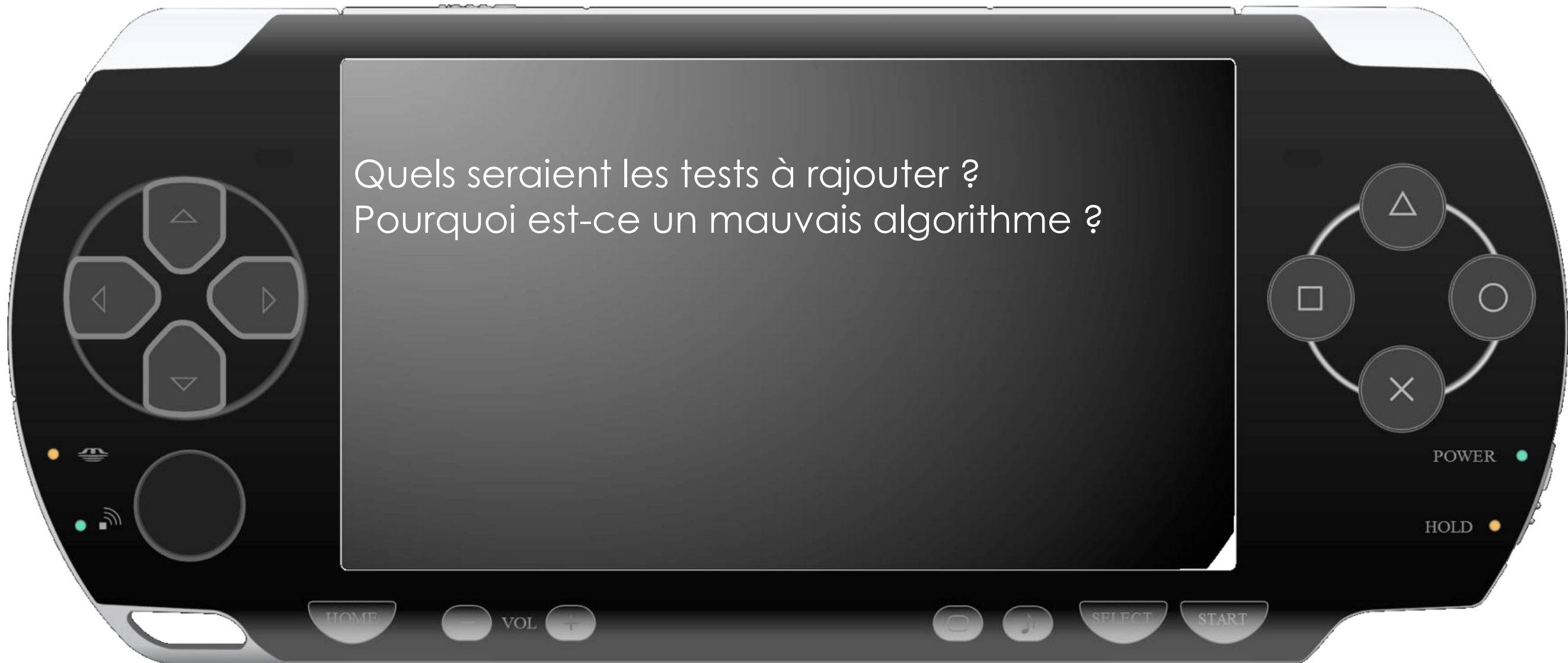
 ECRIRE motmiroir

FIN



Question

Quels seraient les tests à rajouter ?
Pourquoi est-ce un mauvais algorithme ?





ALGORITHME : Mirroir

//BUT : cet algorithme inverse un mot de trois lettres

//ENTREE : Le mot de trois lettres saisi par l'utilisateur

//SORTIE : le mot miroir

VAR : mot, motmiroir: **CHaine**

DEBUT

ECRIRE "Veuillez entrer votre mot"

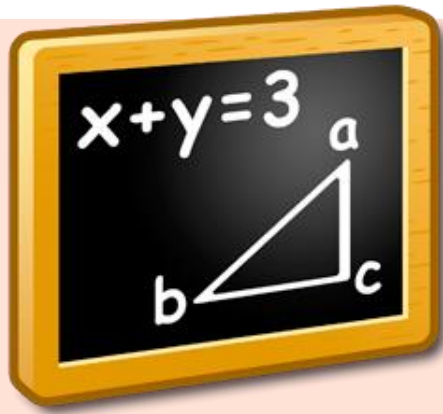
LIRE mot

mot ← EXTRACTION(mot + " _ _ ",1,3)

motmiroir ← EXTRACTION(mot,3,1) + EXTRACTION(mot,2,1) + EXTRACTION(mot,1,1)

ECRIRE motmiroir

FIN



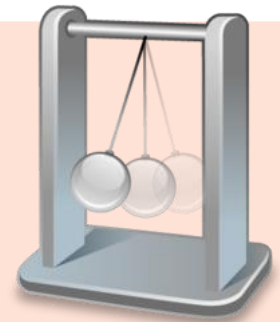
Exercice



Ecrire un algorithme qui demande à un utilisateur un mot de cinq lettres et qui affiche si c'est un palindrome.

ex: « RADAR » affiche VRAI

« Mario » affiche FAUX



10 min



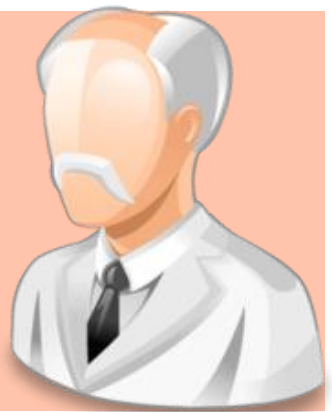
solo



Live



Solution



ALGORITHME : Palin

//BUT : cet algorithme vérifie si un mot est un palindrome

//ENTREE : Le mot saisi par l'utilisateur

//SORTIE : Vrai si c'est un palindrome, faux sinon

VAR : mot, motmiroir: **CHaine**

DEBUT

ECRIRE "Veuillez entrer votre mot"

LIRE mot

mot ← EXTRACTION(mot + "_____",1,5)

motmiroir ← EXTRACTION(mot,5,1) + EXTRACTION(mot,4,1) + EXTRACTION(mot,3,1)
+ EXTRACTION(mot,2,1) + EXTRACTION(mot,1,1)

ECRIRE motmiroir = mot

FIN



Solution



ALGORITHME : Palin

//BUT : cet algorithme vérifie si un mot est un palindrome

//ENTREE : Le mot saisi par l'utilisateur

//SORTIE : Vrai si c'est un palindrome, faux sinon

VAR : mot, motmiroir: **CHaine**

motegal : **BOOLEEN**

DEBUT

ECRIRE "Veuillez entrer votre mot"

LIRE mot

mot ← EXTRACTION(mot , " _ _ _ _ _ ",1,5)

motmiroir ← EXTRACTION(mot,5,1) + EXTRACTION(mot,4,1) + EXTRACTION(mot,3,1)
+ EXTRACTION(mot,2,1) + EXTRACTION(mot,1,1)

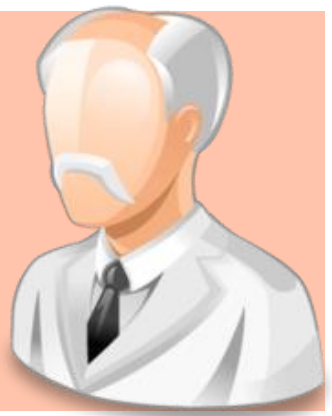
motegal ← (motmiroir = mot)

ECRIRE motegal

FIN



Solution



ALGORITHME : Palin

//BUT : cet algorithme vérifie si un mot est un palindrome

//ENTREE : Le mot saisi par l'utilisateur

//SORTIE : Vrai si c'est un palindrome, faux sinon

VAR : mot: **CHaine**

 estpalin: **BOOLEEN**

DEBUT

 Ecrire "Veuillez entrer votre mot"

 lire mot

 mot ← EXTRACTION(mot + " _ _ _ _ _",1,5)

 estpalin ← EXTRACTION(mot,5,1) = EXTRACTION(mot,1,1) et EXTRACTION(mot,4,1)
 = EXTRACTION(mot,2,1)

 Ecrire estpalin

FIN



Solution



ALGORITHME : Palin

//BUT : cet algorithme vérifie si un mot est un palindrome

//ENTREE : Le mot saisi par l'utilisateur

//SORTIE : Vrai si c'est un palindrome, faux sinon

VAR : mot: **CHaine**

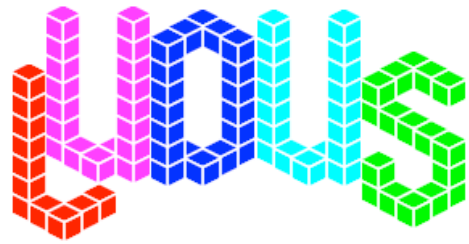
DEBUT

 ECRIRE "Veuillez entrer votre mot"

 LIRE mot

 ECRIRE EXTRACTION(mot + "_____",5,1) = EXTRACTION(mot + "_",1,1) et
EXTRACTION(mot + "_____",4,1) = EXTRACTION(mot+ "__",2,1)

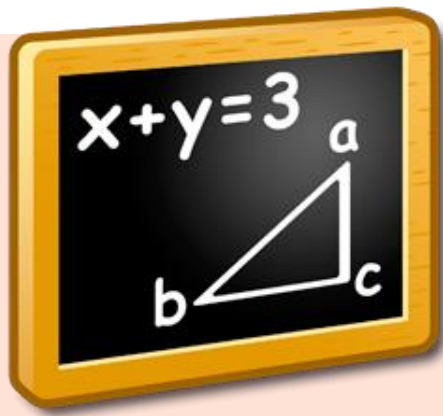
FIN



Algorithmie

Partie N°3

Structure algorithmique



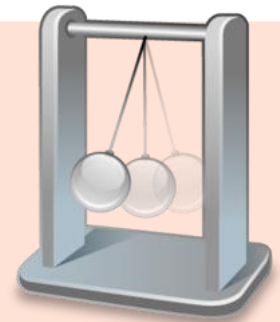
Exercice



Ecrire un algorithme qui demande à un utilisateur trois entiers et qui affiche si ils sont bien dans l'ordre croissant.

ex: « 1, 7, 33 » affiche VRAI

« 4, 19, 8 » affiche FAUX



10 min



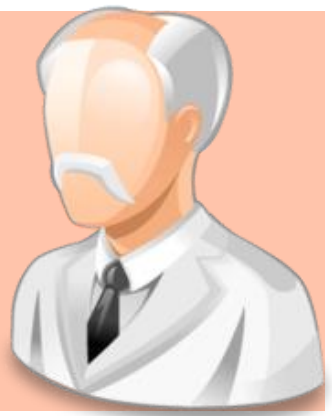
solo



Live



Solution



ALGORITHME : croissant

//BUT : cet algorithme vérifie si 3 entiers sont croissants

//ENTREE : Trois entiers saisis par l'utilisateur

//SORTIE : Vrai si ils sont dans l'ordre, faux sinon

VAR : n1, n2, n3 : **ENTIER**

DEBUT

 ECRIRE "Veuillez entrer un nombre"

 LIRE n1

 ECRIRE "Veuillez entrer un nombre"

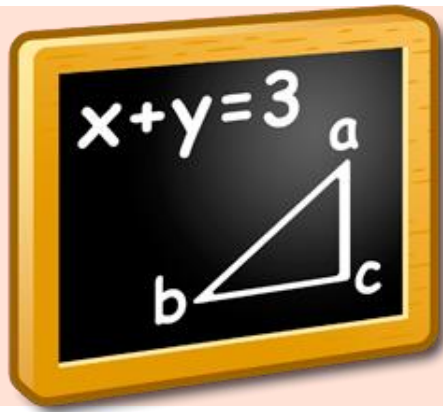
 LIRE n2

 ECRIRE "Veuillez entrer un nombre"

 LIRE n3

 ECRIRE (n1 < n2) ET (n2 < n3)

FIN

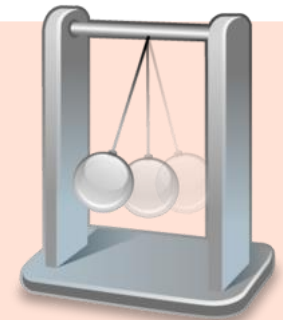


Exercice



Ecrire un algorithme qui demande à un utilisateur Le prix HT d'un produit, la quantité et qui indique si l'utilisateur à le droit à une réduction (La réduction est obtenue si le total TTC est supérieur à 200,00€, le taux de TVA est à 10%)

ex: « 100; 10 » affiche VRAI
« 4,20; 5 » affiche FAUX



10 min



solo



Live



Solution



ALGORITHME : redus

//BUT : cet algorithme vérifie si une reduction est possible

//ENTREE : Le prix HT et la quantité saisi par l'utilisateur

//SORTIE : Vrai si c'est une réduction est possible, faux sinon

VAR : quantite: **ENTIER**
 prix, total : **REEL**

CONST : TVA \leftarrow 1,1 : **REEL**

DEBUT

 ECRIRE "Veuillez entrer le prix HT"

 LIRE prix

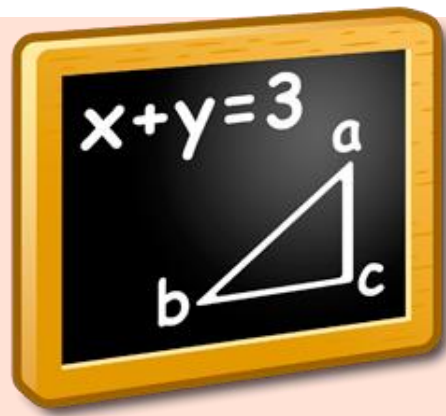
 ECRIRE "Veuillez entrer la quantité"

 LIRE quantite

 total \leftarrow quantite * prix * TVA

 ECRIRE total \geq 200

FIN

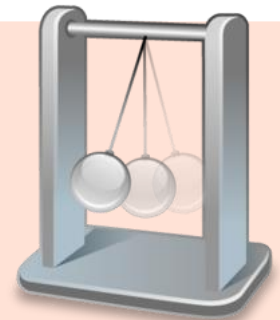


Exercice



Ecrire un algorithme qui demande à un utilisateur le prix d'un loyer annuel hors charge, le salaire mensuel et vérifie que le taux d'endettement ne dépasse pas $x\%$ (x est une constante actuellement à 33%).

ex: « 14000; 3000 » affiche FAUX
« 6000; 1000 » affiche VRAI



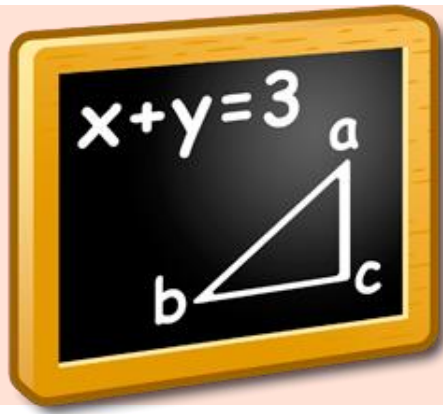
10 min



solo



Live

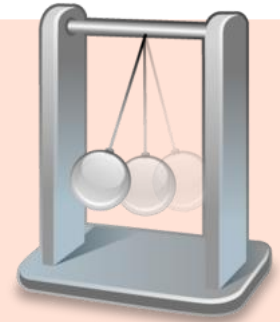


Exercice



Vous voulez calculer le prix de la reprise d'une voiture d'occasion première main de moins de 5 ans. Vous demander le prix neuf au client et son année d'achat. On retire 20% du prix du neuf puis 10% du prix d'achat par année d'ancienneté. Afficher le prix de reprise.

ex: « 20000; 2017 » affiche 10 000
« 50 000; 2019 » affiche 35 000



10 min



solo



Live



Algorithmie

Structure algorithmique

- Présentation
- Rupture de séquence
- Les instructions conditionnelles
- Les instructions répétitives





Citation



Dave Small

« Un langage de programmation est une convention pour donner des ordres à un ordinateur. Ce n'est pas censé être obscur, bizarre et plein de pièges subtils. Ca, ce sont les caractéristiques de la magie. »



Définition

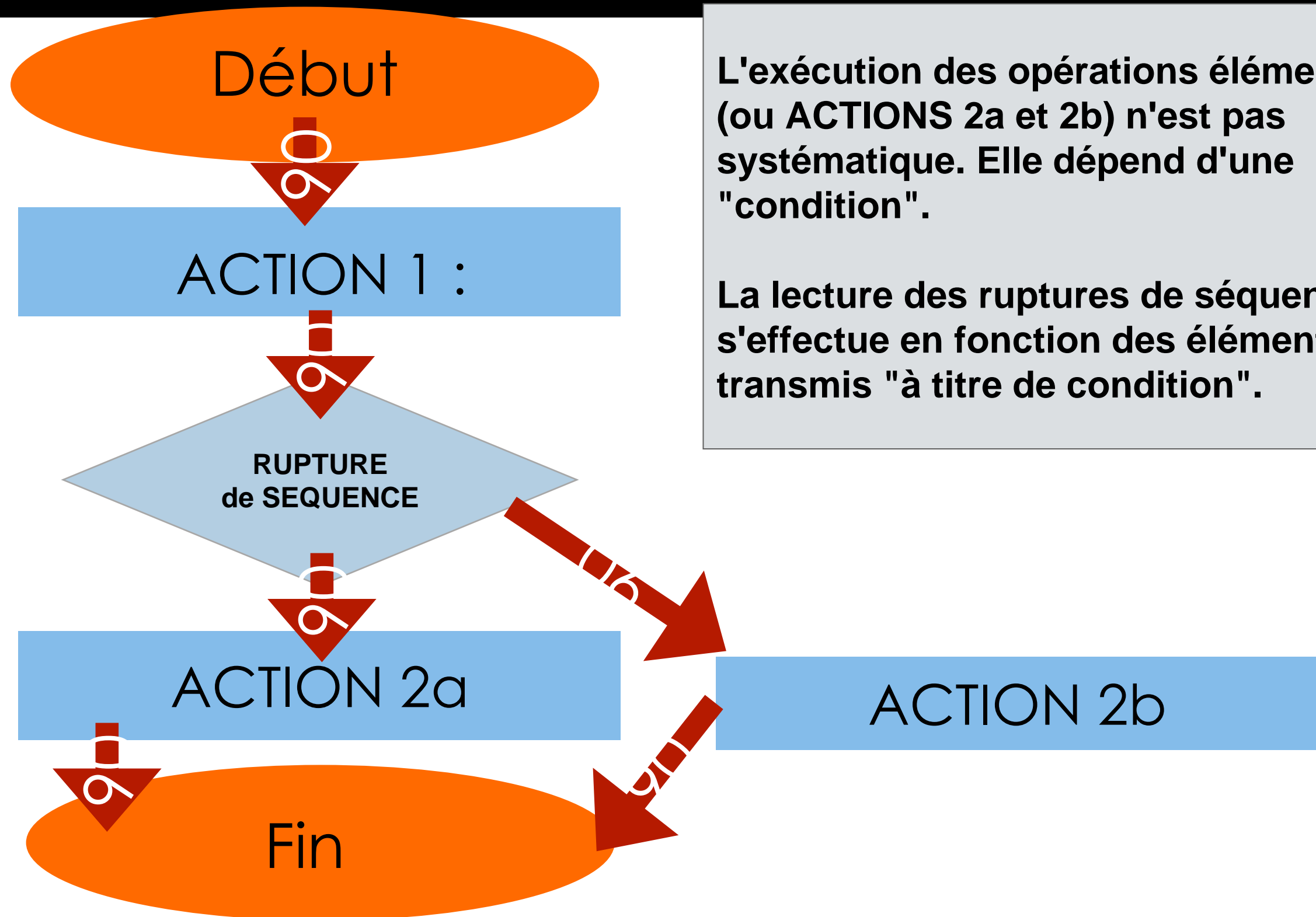
Structure Algorithmique:

La structure algorithmique concerne la concaténation et les imbrications de séquences et ruptures de séquences.

Elle forme ainsi une partie d'un programme.

Algorithmie> Présentation

Notion de rupture de séquences



L'exécution des opérations élémentaires (ou ACTIONS 2a et 2b) n'est pas systématique. Elle dépend d'une "condition".

La lecture des ruptures de séquences s'effectue en fonction des éléments transmis "à titre de condition".

Algorithmie > Structure algorithmique

Catégories

Séquences

Ruptures de séquences :

Rupture conditionnelle (et non répétitive):

SI

CAS

Rupture répétitive :

POUR

TANTQUE

REPETER

Algorithmie > Structure algorithmique

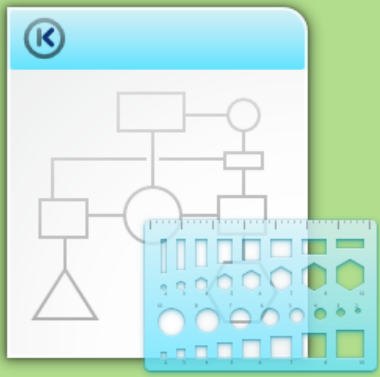
Ruptures conditionnelles

SI..ALORS..FINSI

SI..ALORS..SINON..FINSI

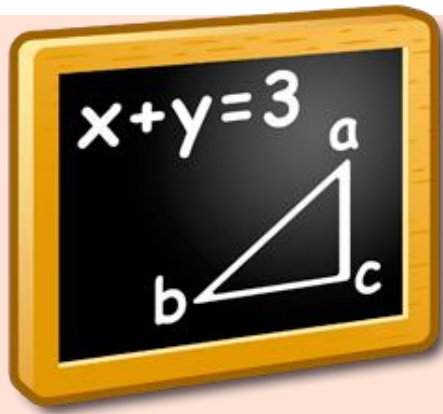
CAS..PARMI..

CAS..PARMI..PARDEFAUT



SI

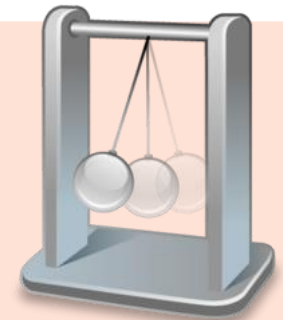
```
SI <condition> ALORS  
  <instructions>  
[SINON  
  <instructions>  
]  
FINSI
```

Exercice



Ecrire un algorithme qui donne le maximum entre deux nombres saisis par l'utilisateur.



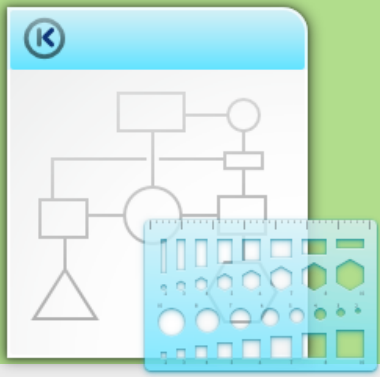
5 min



solo



Live



Algorithme

ALGORITHME : max_deux_nombres

//BUT : cherche la valeur max. parmi 2 valeurs saisies

//ENTREE : deux réels saisis par l'utilisateur

//SORTIE : le maximum des deux valeurs

VAR :

a, b, max **: REEL**

DEBUT

ECRIRE "Veuillez entrer deux nombres"

LIRE a, b

SI (a >= b)

ALORS

max ← a

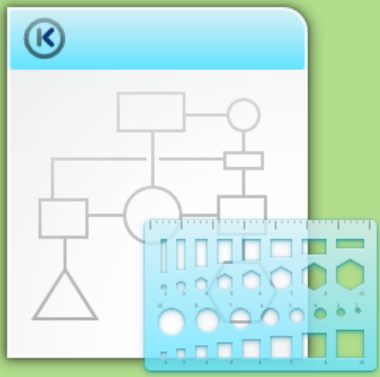
SINON

max ← b

FINSI

ECRIRE "Le Maximum entre " + a + " et " + b + " est: " + max

FIN



Algorithme

ALGORITHME : max_deux_nombres

//BUT : cherche la valeur max. parmi 2 valeurs saisies

//ENTREE : deux réels saisis par l'utilisateur

//SORTIE : le maximum des deux valeurs

VAR :

a, b, max **: REEL**

DEBUT

ECRIRE "Veuillez entrer deux nombres"

LIRE a, b

max \leftarrow b

SI (a \geq b)

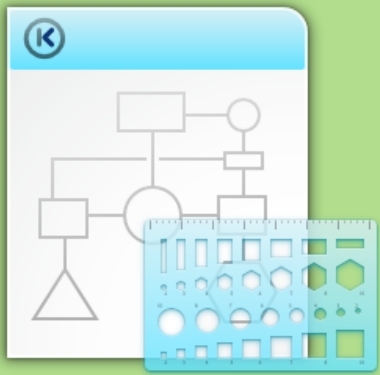
ALORS

max \leftarrow a

FINSI

ECRIRE "Le Maximum entre " + a + " et " + b + " est: " + max

FIN



Algorithme

ALGORITHME : max_deux_nombres

//BUT : cherche la valeur max. parmi 2 valeurs saisies

//ENTREE : deux réels saisis par l'utilisateur

//SORTIE : le maximum des deux valeurs

VAR :

a, b **:REEL**

DEBUT

ECRIRE "Veuillez entrer deux nombres"

LIRE a, b

SI (a >= b)

ALORS

ECRIRE "Le Maximum entre " + a + " et " + b + " est: " + a

SINON

ECRIRE "Le Maximum entre " + a + " et " + b + " est: " + b

FINSI

FIN



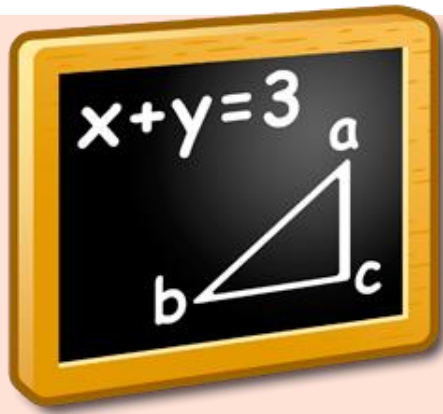
Conseil

Le code de l'algorithme se lit très bien car il a été correctement «indenté». Vous devez vous astreindre à rendre le code lisible.

De même vous devez nommer vos variables de façon claire et précise.

$z \leftarrow x * y * \text{const1}$
n'évoque rien.

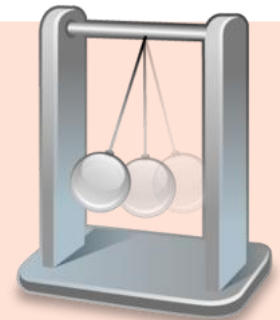
$\text{PrixTTC} \leftarrow \text{PrixHT} * \text{Qte} * \text{Taux-TVA}$
ne demande aucun effort de compréhension.



Exercice



Ecrire un algorithme qui calcule la valeur absolue d'un entier.



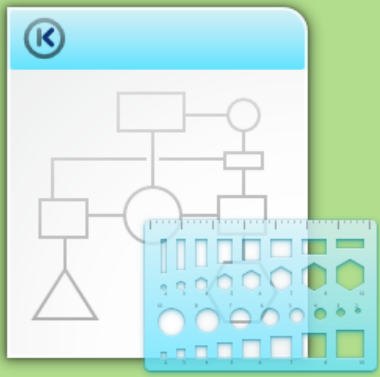
5 min



solo



Live



Algorithme

ALGORITHME: valeur_absolue

//BUT : calcule la val.abs. d'un entier saisi par l'util.

//ENTREE : un entier relatif saisi par l'utilisateur

//SORTIE : la valeur absolue de l'entier saisi

VAR:

x **:ENTIER**

DEBUT

LIRE x //affectation d'une valeur par l'util.

SI (x < 0) //si cette valeur est négative

ALORS //alors cette condition est vraie

 x ← x*-1 //et la variable se trouve réaffectée

FINSI

ECRIRE "La val.abs. du nombre saisi vaut : "+x

FIN



```
//BUT : calcule la val.abs. d'un entier saisi par l'util.
//ENTREE : un entier relatif saisi par l'utilisateur
//SORTIE : la valeur absolue de l'entier saisi
```

```
x, abs : ENTIER
```

```

LIRE x          //affectation d'une valeur par l'util.
SI( x < 0 )    //si cette valeur est négative
ALORS          //alors cette condition est vraie
    abs ← x*-1
SINON
    abs ← x
FINSI
Ecrire "La val.abs. de " + x + "vaut : " + abs

```

FIN



```
//BUT : calcule la val.abs. d'un entier saisi par l'util.  
//ENTREE : un entier relatif saisi par l'utilisateur  
//SORTIE : la valeur absolue de l'entier saisi
```

```
x, abs : ENTIER
```

```
LIRE x //affectation d'une valeur par l'util.
```

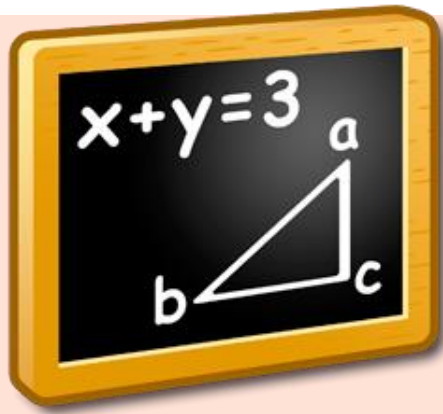
```
SI( x < 0 ) //si cette valeur est négative
```

ALORS //alors cette condition est vraie

$$\text{abs} \leftarrow x^* - 1$$

```
ECRIRE "La val.abs. de " + x + "vaut : " + abs
```

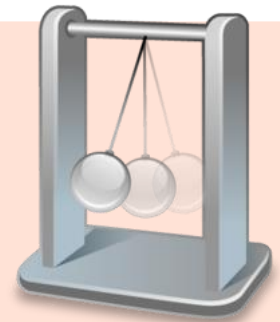
FIN



Exercice



Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on laisse de côté le cas où le nombre vaut zéro).



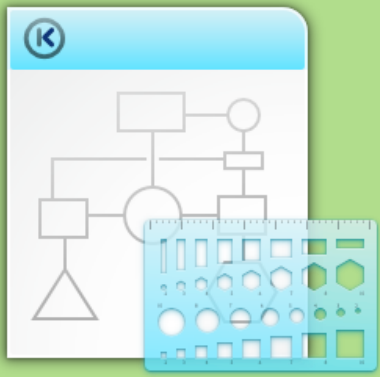
10 min



solo



Live



Algorithme

ALGORITHME : exo-1

VAR :

n : REEL

DEBUT

ECRIRE " Entrez un nombre "

LIRE n

SI (n > 0)

ALORS //alors cette condition est vraie

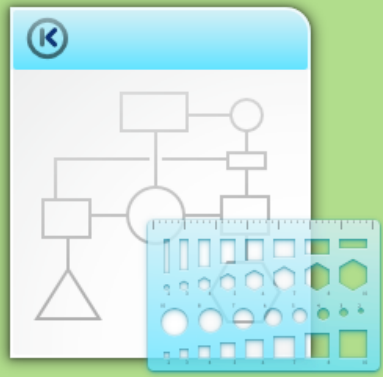
ECRIRE "Ce nombre est positif"

SINON

ECRIRE "Ce nombre est négatif"

FINSI

FIN



Algorithme

ALGORITHME: exo-1

VAR:

n **:REEL**

mot **:CHAINE**

DEBUT

ECRIRE " Entrez un nombre "

LIRE n

SI (n > 0)

ALORS //alors cette condition est vraie

 mot ← "positif"

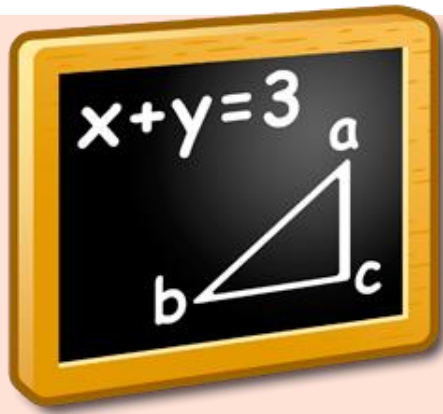
SINON

 mot ← "négatif"

FINSI

ECRIRE "Le nombre " + n + " est " + mot

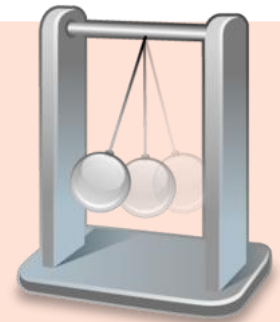
FIN



Exercice



Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul).



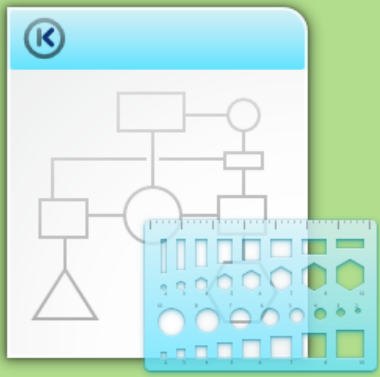
10 min



solo



Live



Algorithme

ALGORITHME : exo-2

VAR :

m, n	: REEL
mot	: CHAINE

DEBUT

ECRIRE " Entrez deux nombres "

LIRE m, n

SI $m * n > 0$

ALORS

mot \leftarrow "positif"

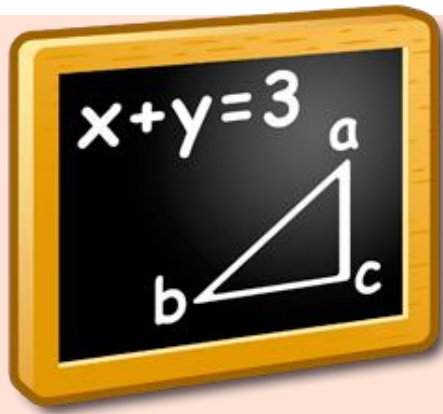
SINON

mot \leftarrow "négatif"

FINSI

ECRIRE "Le produit de " + m + " et " + n + " est " + mot

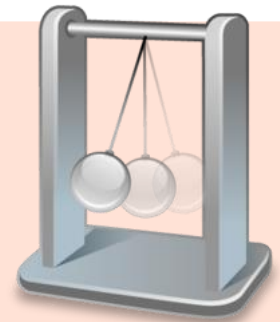
FIN



Exercice



Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul). Attention toutefois : on ne doit pas calculer le produit des deux nombres.



10 min



solo



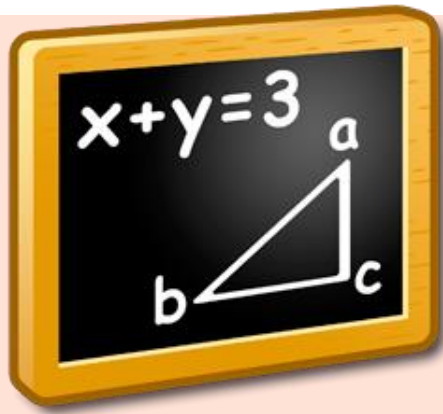
Live



Algorithme

: ENTIER

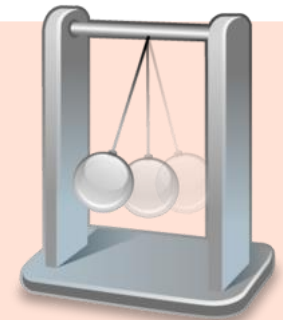
FIN



Exercice



Ecrire un algorithme qui demande trois noms à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre alphabétique.



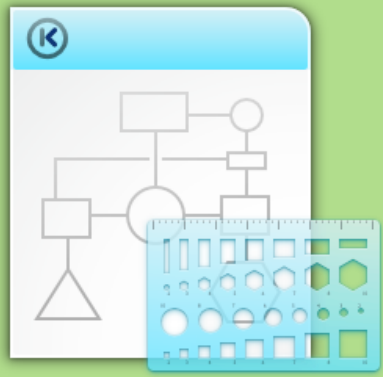
5 min



solo



Live



Algorithme

ALGORITHME : exo-4

VAR :

m1, m2, m3 : CHAINE

DEBUT

ECRIRE " Entrez trois noms "

LIRE m1, m2, m3

SI ((m1<=m2) et (m2<=m3))

ALORS

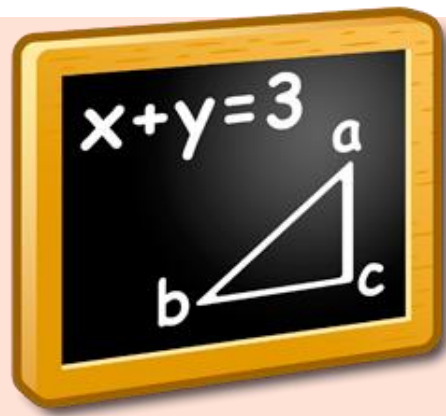
ECRIRE "Ces noms sont classés"

SINON

ECRIRE "Ces noms ne sont pas classés"

FINSI

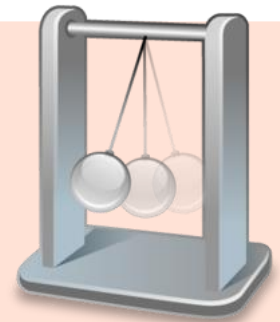
FIN



Exercice



Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif, négatif ou nul.



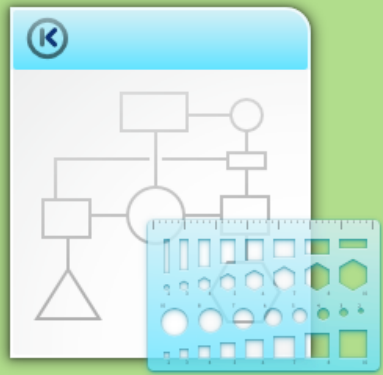
5 min



solo



Live



Algorithme

ALGORITHME: exo-1

VAR:

n :REEL

DEBUT

ECRIRE " Entrez un nombre "

LIRE n

SI (n > 0) **ALORS**

ECRIRE "Ce nombre est positif"

SINON

SI (n < 0) **ALORS**

ECRIRE "Ce nombre est négatif"

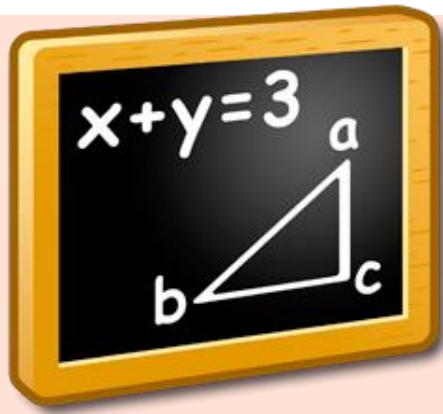
SINON

ECRIRE "Ce nombre est nul "

FINSI

FINSI

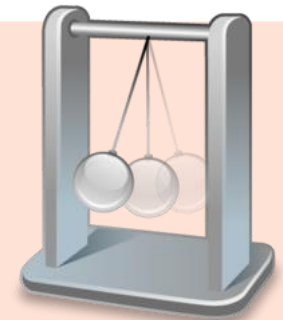
FIN



Exercice



Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si le produit est négatif ou positif (on inclut cette fois le traitement du cas où le produit peut être nul). Attention toutefois, on ne doit pas calculer le produit !



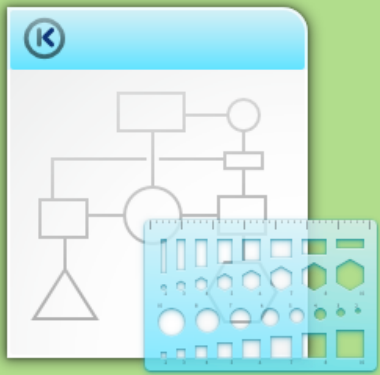
5 min



solo



Live



Algorithme

ALGORITHME : exo-3

VAR :

m, n : **ENTIER**

DEBUT

ECRIRE " Entrez deux nombres "

LIRE m, n

SI ((m > 0 ET n > 0) OU (m < 0 ET n < 0)) **ALORS**

ECRIRE "Le produit est positif"

SINON

SI (m = 0 OU n = 0) **ALORS**

ECRIRE "Le produit est nul"

SINON

ECRIRE "Le produit est négatif"

FINSI

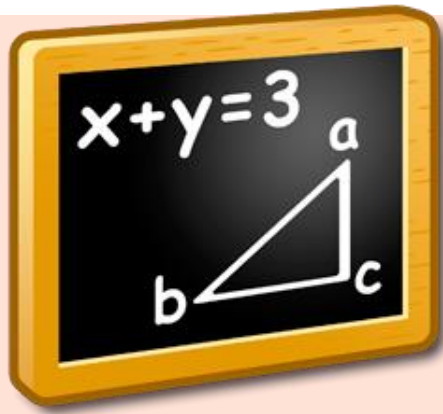
FINSI

FIN



Conseil

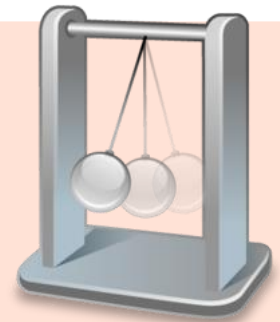
On a ici des structures IMBRIQUEES.
L'art de la programmation est de
combiner et d'imbriquer
intelligemment ces structures.



Exercice



Ecrire un algorithme qui demande le numéro d'un mois de l'année et affiche le nom du mois.



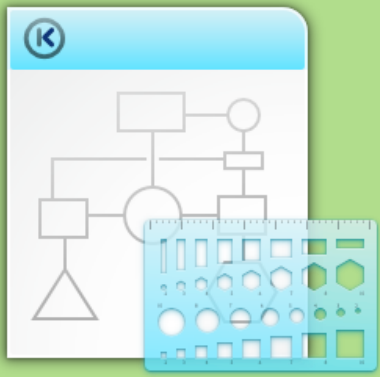
15 min



solo



Live



Algorithme

ALGORITHME: mois1

VAR:

 mois **: ENTIER**

 mois_chaine **: CHAINE**

DEBUT

 LIRE mois

SI(mois = 1)

ALORS

 mois_chaine ← "janvier "

FINSI

SI(mois = 2)

ALORS

 mois_chaine ← "février "

FINSI

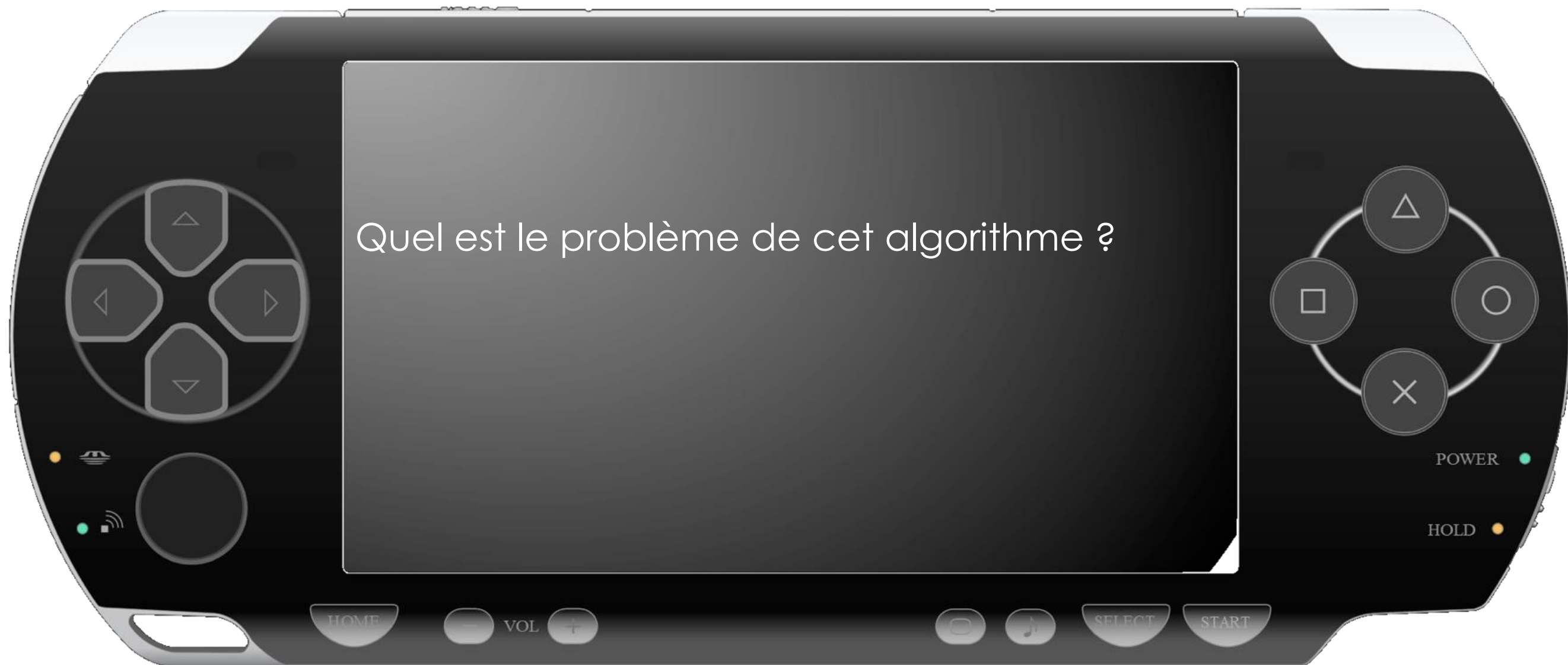
 ...

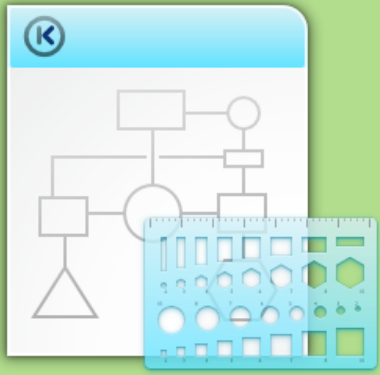
ECRIRE "Le mois est : " + mois_chaine

FIN



Question





Algorithme

ALGORITHME : mois2

VAR :

 mois **: ENTIER**

 mois_chaine **: CHAINE**

DEBUT

 LIRE mois

SI (mois = 1)

ALORS

 mois_chaine ← "janvier "

FINSI

SI (mois = 2)

ALORS

 mois_chaine ← "février "

FINSI

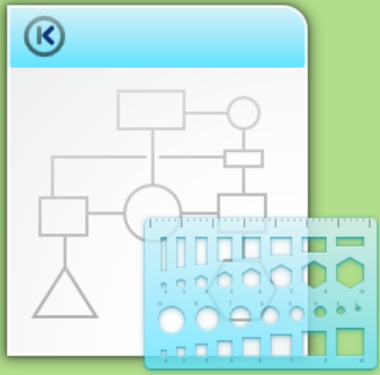
 ...

SI mois >= 1 et mois <= 12 **ALORS**

 Ecrire "Le mois est : " + mois_chaine

FIN SI

FIN



Algorithme

ALGORITHME : mois3

VAR :

 mois **: ENTIER**

 mois_chaine **: CHAINE**

DEBUT

 LIRE mois

 mois_chaine ← ""

SI (mois = 1)

ALORS

 mois_chaine ← "janvier "

FINSI

SI (mois = 2)

ALORS

 mois_chaine ← "février "

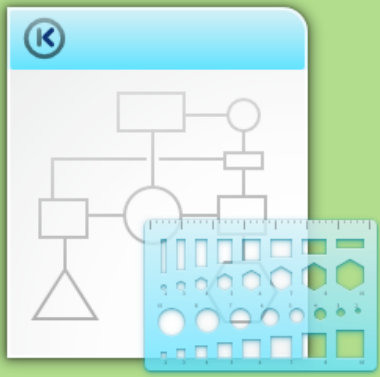
FINSI

 ...

SI mois_chaine <> "" **ALORS**

 Ecrire "Le mois est : " + mois_chaine **FINSI**

FIN



Algorithme

ALGORITHME : mois4

VAR :

 mois **: ENTIER**

 mois_chaine **: CHAINE**

DEBUT

 LIRE mois

SI (mois = 1) **ALORS**

 mois_chaine ← "janvier "

SINON SI (mois = 2)

ALORS

 mois_chaine ← "février "

SINON SI (mois = 3)

ALORS

 mois_chaine ← "mars "

SINON

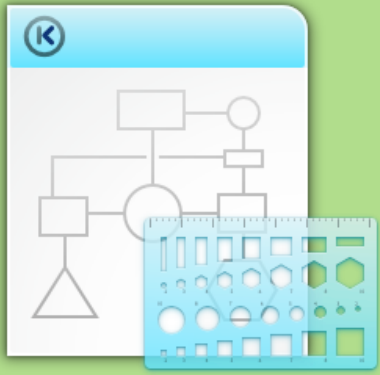
 ...

FINSI FINSI FINSI

FINSI

 Ecrire "Le mois est : " + mois_chaine

FIN



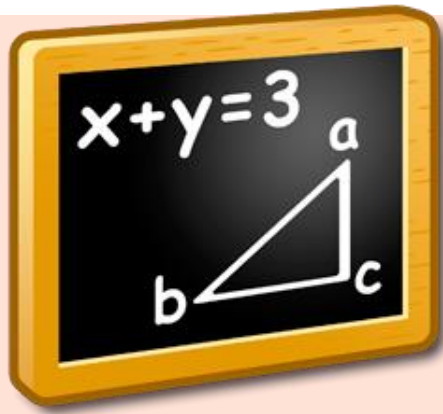
CAS PARM

CAS <variable> PARM

{CAS {value[,]}: <instructions>}

[PARDEFAULT: <instructions>]

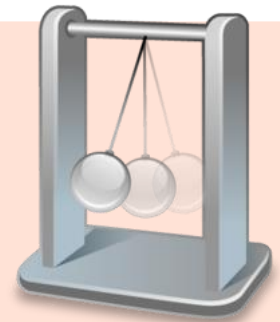
FINCASPARI



Exercice



Ecrire un algorithme qui demande le numéro d'un mois de l'année et affiche le nom du mois en utilisant la structure cas parmi.



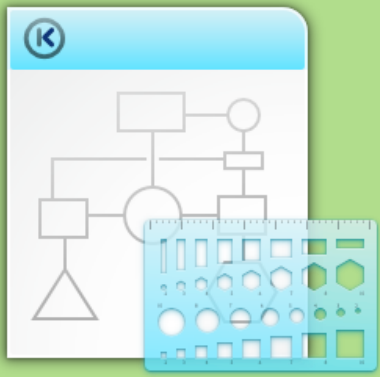
10 min



solo



Live



Algorithme

ALGORITHME: nom_mois2

//BUT : affiche le nom du mois en fonction du numéro

//ENTREE : un entier saisi par l'utilisateur

//SORTIE : le nom du mois correspondant au chiffre saisi

VAR:

mois **:ENTIER**

DEBUT

LIRE mois

CAS mois **PARMI:**

CAS 1: ECRIRE " Janvier "

CAS 2: ECRIRE " Février "

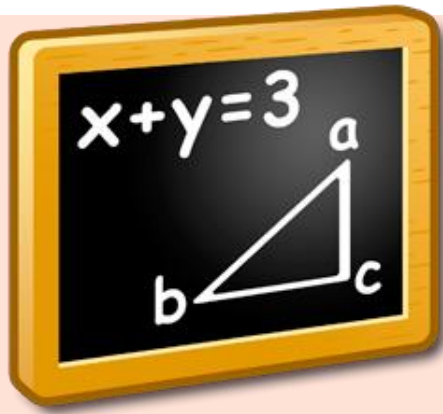
 ...

CAS 12: ECRIRE " Décembre "

PARDEFAUT: ECRIRE " Erreur de saisie ! "

FINCASPARMI

FIN



Exercice

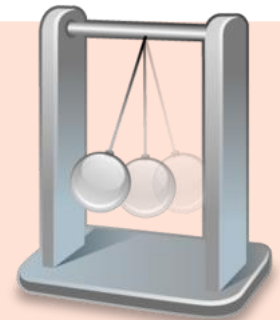


Adapter l'algorithme qui transforme le numéro du mois en nombre de jour.

Sachant que depuis l'[ajustement du calendrier grégorien](#), sont bissextiles les années :

- soit divisibles par 4 mais non divisibles par 100 ;
- soit divisibles par 400.

Afiner le calcul pour le mois de février.



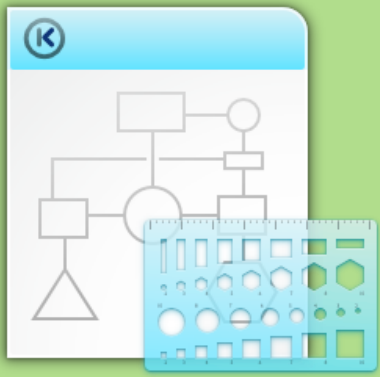
15 min



solo



Live



Algorithme

ALGORITHME : nom_mois

//BUT : affiche le nom du mois en fonction du numéro

//ENTREE : un entier saisi par l'utilisateur

//SORTIE : le nom du mois correspondant au chiffre saisi

VAR :

mois, annee **: ENTIER**

DEBUT

LIRE mois, annee

CAS (mois) PARMIS :

CAS : 1, 3, 5, 7, 8, 10, 12 ECRIRE (" 31")

CAS : 2 **SI** (ANNEE MOD 4=0 ET ANNEE MOD 100<>0) ou
 (ANNEE MOD 400=0) **ALORS**
 ECRIRE "29"

SINON

 ECRIRE "28"

FINSI

PARDEFAUT : ECRIRE "30"

FINCASPARMI

FIN

Algorithmie > Structure algorithmique

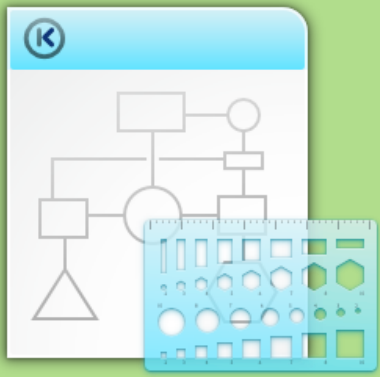
Ruptures de répétition

POUR..FAIRE..FINPOUR

TANTQUE..FAIRE..FINTANTQUE

REPETER..JUSQU'A..

REPETER..TANTQUE..

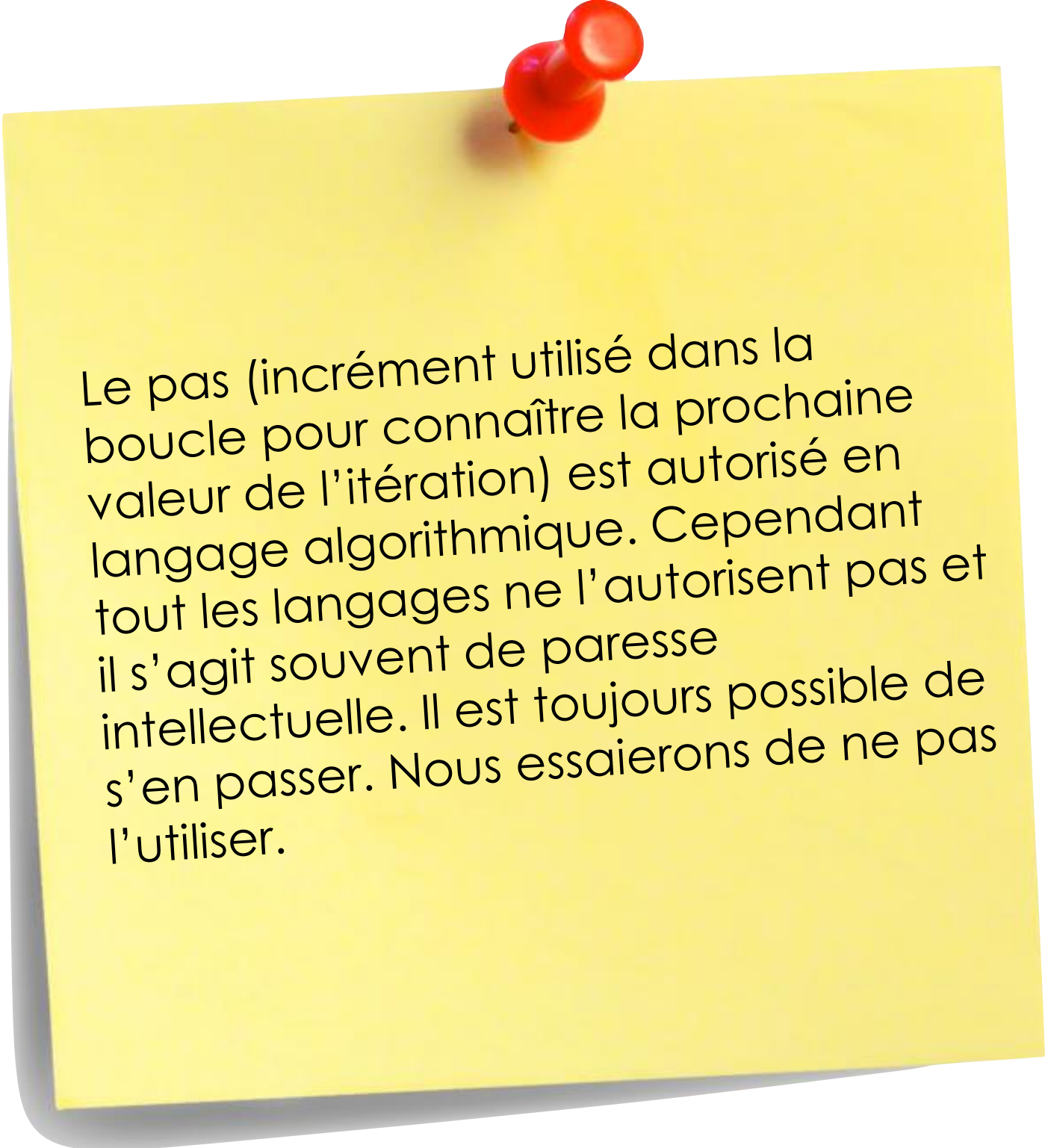


POUR

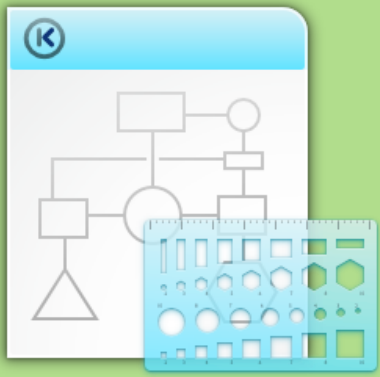
POUR <variable entière> DE <variable ou
constante entière> A <variable ou constante
entière> [PAS <variable ou constante entière>]
FAIRE
 [<instructions>]
FINPOUR



Conseil



Le pas (incrément utilisé dans la boucle pour connaître la prochaine valeur de l'itération) est autorisé en langage algorithmique. Cependant tout les langages ne l'autorisent pas et il s'agit souvent de paresse intellectuelle. Il est toujours possible de s'en passer. Nous essaierons de ne pas l'utiliser.



Algorithme

ALGORITHME : compteur_rebours1

//BUT : affiche la valeur d'un compteur à rebours

//ENTREE : un entier saisi par l'utilisateur

//SORTIE : la valeur du compteur à rebours

VAR :

nombre, compteur : **ENTIER**

DEBUT

LIRE nombre

POUR compteur **DE** nombre **A** 1 pas - 1

FAIRE

 Ecrire "Valeur du compteur à rebours:" + compteur

FINPOUR

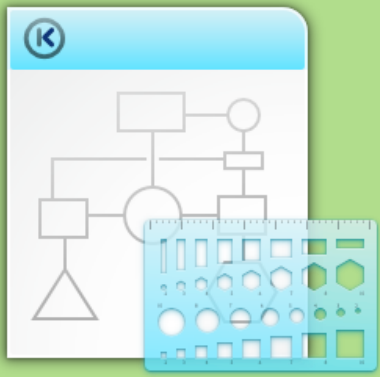
FIN



Conseil



Revient à écrire



Algorithme

ALGORITHME : compteur_rebours2

//BUT : affiche la valeur d'un compteur à rebours

//ENTREE : un entier saisi par l'utilisateur

//SORTIE : la valeur du compteur à rebours

VAR :

nombre, compteur : **ENTIER**

DEBUT

LIRE nombre

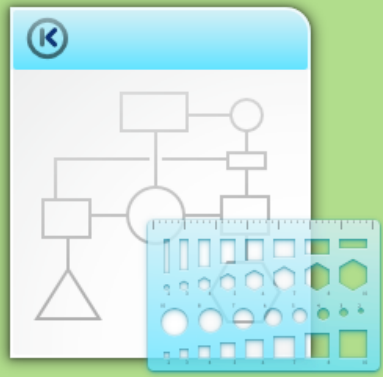
POUR compteur **DE** 0 **A** nombre -1

FAIRE

 Ecrire "Valeur du compteur à rebours: " + nombre -compteur

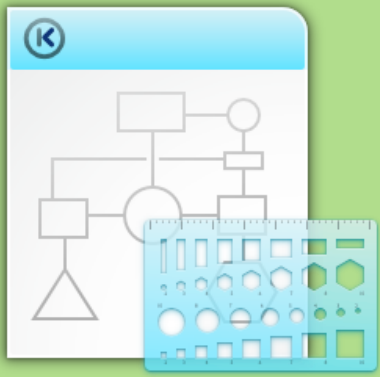
FINPOUR

FIN



TANTQUE

**TANTQUE <variable booléenne> FAIRE
{<instructions>
FINTANTQUE**



Algorithme

ALGORITHME: compteur_rebours3

//BUT : affiche la valeur d'un compteur à rebours

//ENTREE : un entier saisi par l'utilisateur

//SORTIE : la valeur du compteur à rebours

VAR:

nombre **:ENTIER**

DEBUT

LIRE(nombre)

TANTQUE (nombre > 0)

FAIRE

 ECRIRE("Valeur du compteur à rebours: " + nombre)

 nombre ← nombre - 1

FINTANTQUE

FIN



Instruction répétitive « Tantque »

Instruction répétitive :

TANTQUE..FAIRE..FINTANTQUE

```
ALGORITHME: compteur_rebours
```

```
//BUT : affiche la valeur d'un compteur à rebours
```

```
//ENTREE : un entier saisi par l'utilisateur
```

```
//SORTIE : la valeur du compteur à rebours
```

```
VAR:
```

```
    nombre      : ENTIER
```

```
DEBUT
```

```
    LIRE ( nombre )
```

```
    TANTQUE ( nombre > 0 )
```

```
    FAIRE
```

```
        ECRIRE ( "Valeur du compteur à rebours:" + nombre )
```

```
        nombre <-- nombre - 1
```

```
    FINTANTQUE
```

```
FIN
```



Instruction répétitive « Répéter »

Instruction répétitive :
REPETER..JUSQU'A..

```
ALGORITHME: compteur_rebours2
//BUT : affiche la valeur d'un compteur à rebours
//ENTREE : un entier saisi par l'utilisateur
//SORTIE : la valeur du compteur à rebours
VAR:
    nombre      : ENTIER

DEBUT
    LIRE( nombre )
    REPETER
        ECRIRE( "Valeur du compteur à rebours: " + nombre )
        nombre <-- nombre - 1
    JUSQU'A( nombre < 0 )
FIN
```




Instruction répétitive « Répéter »

Instruction répétitive :
REPETER..TANTQUE..

```
ALGORITHME: compteur_rebours3
//BUT : affiche la valeur d'un compteur à rebours
//ENTREE : un entier saisi par l'utilisateur
//SORTIE : la valeur du compteur à rebours
VAR:
    nombre      : ENTIER

DEBUT
    LIRE( nombre )
    REPETER
        ECRIRE( "Valeur du compteur à rebours: " + nombre )
        nombre <-- nombre - 1
    TANTQUE ( nombre > 0 )
FIN
```



Exercice(s) résolu(s)

SI..ALORS..FINSI

Enoncé : Un élève entre la valeur d'une moyenne pour un examen, le programme indique la réussite (ou non) s'il obtient une note supérieure ou égale à 12.

ALGORITHME : examen

//BUT :détermine l'obtention d'un examen

//ENTREE :un entier saisi par l'utilisateur

//SORTIE :la valeur booléenne indiquant le résultat

VAR :

moyenne

: REEL

examen

: BOOLEEN

DEBUT

examen <-- FAUX

LIRE(moyenne)//affectation d'une valeur par l'util.

SI(moyenne >= 12) //si moyenne est sup.ou ég.à 12

ALORS //alors cette condition est vraie

examen <-- VRAI //et examen se trouve réaffectée

FINSI

ECRIRE("Cet élève a réussi son examen : " + examen)

FIN



Exercice(s) résolu(s)

SI..ALORS..SINON..FINSI

Enoncé : En fonction du prix d'un produit HT et de la catégorie de celui-ci (lux: TVA=19,6 % ; autre: TVA=5,5 %), le programme calcule sa valeur TTC.

```
ALGORITHME: prixttc
```

```
CONST:
```

```
    tva_luxe <-- 1,196           :REEL
```

```
    tva_reduite <-- 1,055       :REEL
```

```
VAR:
```

```
    prix_ht, prix_ttc           :REEL
```

```
    produit_luxe                :BOOLEEN
```

```
DEBUT
```

```
    LIRE( prix_ht, produit_luxe)
```

```
    SI( produit_luxe = VRAI )
```

```
        ALORS
```

```
            prix_ttc <-- prix_ht * tva_luxe
```

```
        SINON
```

```
            prix_ttc <-- prix_ht * tva_reduite
```

```
        FINSI
```

```
    ECRIRE ("Le prix TTC du produit est : " + prix_ttc)
```

```
FIN
```



Exercice(s) résolu(s)

CAS..PARMI..FINCASPARI

Enoncé: Lorsque que l'utilisateur entre le choix d'un menu, pour 1, il affiche : "Nom du fichier : Algo1.txt"; pour 2 : "Nom du répertoire : C:"; pour 3 : "Nom complet : C:\Algo1.txt".

ALGORITHME: menu_fichier

VAR:

choix_menu : **ENTIER**

DEBUT

LIRE(choix_menu)

CAS(choix_menu) **PARMI :**

CAS1: 1 ECRIRE("Nom du fichier : Algo1.txt")

CAS2: 2 ECRIRE("Nom du répertoire : C:")

CAS3: 3 ECRIRE("Nom complet : C:\Algo1.txt")

FINCASPARI

FIN



Exercice(s) résolu(s)

CAS..PARMI..PARDEFAUT..FINCASPARI

Enoncé: En fonction du numéro du mois choisi par l'utilisateur, le programme affiche le nombre de jours correspondant (les années bissextiles ne sont pas prises en compte).

ALGORITHME: nombre_jours

VAR:

mois	: CHAINE
nbre_jours	: ENTIER

DEBUT

LIRE(mois)

CAS(mois) **PARMI:**

CAS1: "Février" nbre_jours <-- 28

CAS2: "Avril" nbre_jours <-- 30

CAS3: "Juin" nbre_jours <-- 30

CAS4: "Septembre" nbre_jours <-- 30

CAS5: "Novembre" nbre_jours <-- 30

PARDEFAUT: nbre_jours <-- 31

FINCASPARI

ECRIRE ("Il y a « + nbre_jours+ " jours en "+ mois)

FIN



Exercice(s) résolu(s)

TANTQUE..FAIRE..FINTANTQUE

Enoncé: Le programme affiche systématiquement chaque touche que l'utilisateur frappe jusqu'à ce que la touche 'q' soit entrée ('q' correspond à QUITTER).

```
ALGORITHME: touche_appuyee
//affichage de la touche choisi dès lors que celle-ci
//n'est pas un 'q' qui correspond au menu "Quitte"
VAR:
    touche      : CARACTERE

DEBUT
    LIRE( touche )
    TANTQUE ( touche <> 'q' )           //'q' pour Quitte
    FAIRE
        ECRIRE( "La touche est : " + touche )
        LIRE( touche )
    FINTANTQUE
FIN
```



Exercice(s) résolu(s)

REPETER..JUSQU'A..

Enoncé: Le programme affiche systématiquement chaque mot que l'utilisateur frappe jusqu'à ce que le mot "fin" soit saisi. Le programme s'arrête à ce moment-là, le mot "fin" étant le dernier à apparaître sur l'écran.

```
ALGORITHME: mots_tapes
//affichage du mot tapé dès lors que celui-ci
//n'est pas le mot "fin"
VAR:
    mot          : CHaine

DEBUT
    REPETER
        LIRE ( mot )
        ECRIRE ( mot )
    JUSQU'A ( mot = "fin" )
FIN
```



Exercice(s) résolu(s)

REPETER..TANTQUE..

Enoncé: Le programme affiche systématiquement chaque mot que l'utilisateur frappe jusqu'à ce que le mot "fin" soit saisi. Le programme s'arrête à ce moment-là, le mot "fin" étant le dernier à apparaître sur l'écran.

```
ALGORITHME: mots_tapes2
//affichage du mot tapé dès lors que celui-ci
//n'est pas le mot "fin"
VAR:
    mot          : CHaine

DEBUT
    REPETER
        LIRE ( mot )
        ECRIRE ( mot )
    TANTQUE ( mot <> "fin" )
FIN
```




Exercice(s) résolu(s)

POUR..FAIRE..FINPOUR

Enoncé: Le programme affiche un compteur à rebours pour indiquer le nombre de photocopies restantes, avant d'avoir lancé l'impression d'un fichier: "C:\Rapport.txt". (L'impression est réalisée après l'affichage. Noter seulement en commentaire la fonction d'impression).

```
ALGORITHME:  compteur_copies
VAR:
    nombre,  compteur           : ENTIER

DEBUT
    LIRE nombre    //nombre de copie à photocopier
    POUR compteur de 1 à nombre
        FAIRE
            ECRIRE "Photocopies restantes:", nombre+1- compteur)
//    IMPRIMER LE FICHIER "C:\Rapport.txt"
    FINPOUR
FIN
```

FIN
de la Partie 2