ALGORITHMIE

Partie algobox

Apprendre à réaliser un algorithme

ou comment résoudre un problème en structurant la mise en œuvre d'une solution

Exemple

Enjeu: Être plus efficace dans le bricolage de ma voiture

Objectif : Ranger son armoire de bricolage pour la voiture

Étapes

- Regrouper les objets « voiture »
- Trier les objets « voiture »
- Créer/ adapter le rangement « voiture »

Enchaînements

Pour chaque objet de l’univers Voiture

regrouper avec le même type d’objet

outils/pièces/vis, etc.

regrouper à nouveau par fonctionnalites

clés avec clés, tournevis avec tournevis, etc.

trier par sa taille

clés de taille mini à maxi

Choisir le casier adapté

une vis est la plus petite mais la quantité la plus grande

si casier n’existe pas alors le créer

ranger

et reprendre un objet

Généralisation de la solution

Puis-je appliquer cet enchaînement au rangement de mon univers Vélo?

si oui, quel changement va avoir lieu dans l'enchaînement?

définition de l'univers
énumération des objets par univers

Puis-je appliquer cet enchaînement au rangement de mon univers Menuiserie? de la cuisine? de mon magasin à l'atelier?

Quel autre exemple, significativement différent proposez-vous?

Complexité

Ce n’est pas parce qu’un ordinateur est plus puissant qu’un autre, que le même algorithme, exécuté sur les 2 postes, sera plus rapide.

L’efficacité d’un algorithme: c’est l’utilisation correcte de la mémoire, la simplicité du traitement, la vitesse des enchaînements quelque soit le nombre d’opérations

Parfois on ne peut tester en conditions réelles une solution

Une mesure: « si je donne à mon programme une entrée de taille N, quel est l'ordre de grandeur, en fonction de N, du nombre d'opérations qu'il va effectuer ? »

Traiter 2 X plus de données ne devrait pas prendre plus de temps

Consommation de la mémoire doit être évaluée mais aussi de l’énergie (vitesse ou consommation dans les systèmes embarqués?)

QUELQUES DEFINITIONS

Lexique informatique

RAM: Mémoire vive faite de plusieurs millions de composants qui "filtrent" une charge électrique

BIT: information binaire (0 ou 1)

OCTET: groupe de 8 bits (en anglais , Byte)

2^8 possibilités soit 256 nombres différents

2 octets: 65 536 possibilités (256*256)

3 octets: 16 777 216 possibilités (256*256*256)

ASCII: (American Standard Code Information Interchange) standard international de codage des caractères et ponctuations

Vocabulaire du développeur

Langage de programmation:	Instruction:	Programmation:	Pseudo code:
Convention d'instructions organisées	consigne formulée dans un langage de programmation selon un code	permet de traduire l'algorithme dans un langage adapté à l'ordinateur	organisé comme un langage de programmation mais sans les soucis de syntaxes (conventions)

Vocabulaire du développeur

Interprétation:

chaque ligne du programme source est traduite en instructions du langage machine au fur et à mesure,

moins rapide mais plus de polyvalence —*multi-plateforme*

Vba, Php…

Compilateur:

le programme est traduit en une seule fois et stocké dans un exécutable

plus rapide

C, C++,…

Semi-compilé:

combine les 2 techniques en compilant d’abord et en les interprétant ensuite

Python, Java…

LE PSEUDO CODE

Description

Écriture d’algorithme à l’aide d’un vocabulaire simple

support informatique optionnel

possibilité d’échanger avec un développeur même sans connaître de langage particulier

Syntaxe

Pas de standard mais des conventions

Mots clés en gras

Début, Fin, Si, Alors, Sinon Tant Que, Jusqu'à…

Opérateurs

<- (affectation)

+, - ,*, /

==, ( )

<, >, !=, <=, >=

** (puissance)

+ (concaténation)

% modulo (reste division entière)

attention, Mod est une fonction Excel mais un opérateur VBA

Exemple

algorithme : Travail de la journée

Début

prendre l'agenda

aller à aujourd'hui

TANT QUE il y a une tâche FAIRE

Lire tâche

Réaliser tâche

Passer à la tâche suivante

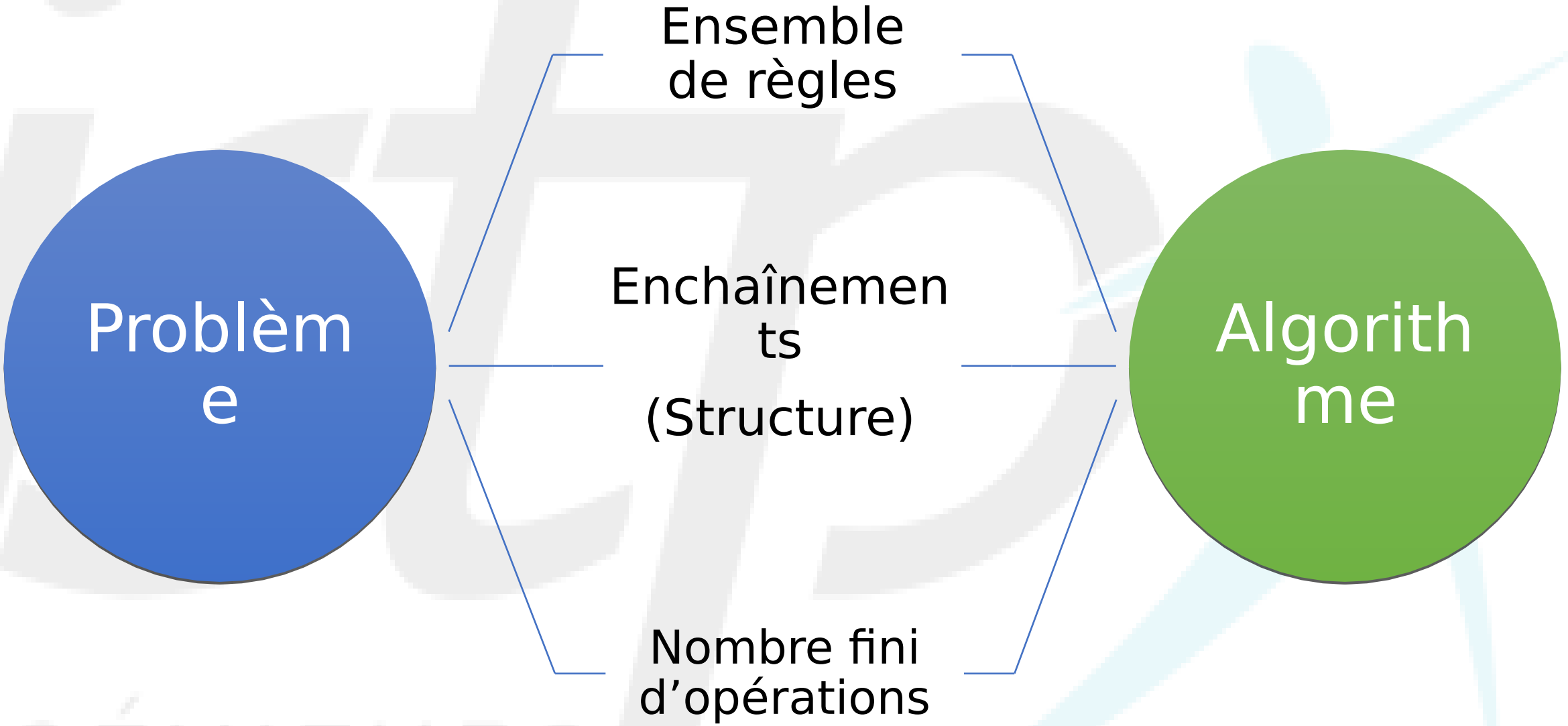
FIN TANT QUE

Fermer agenda

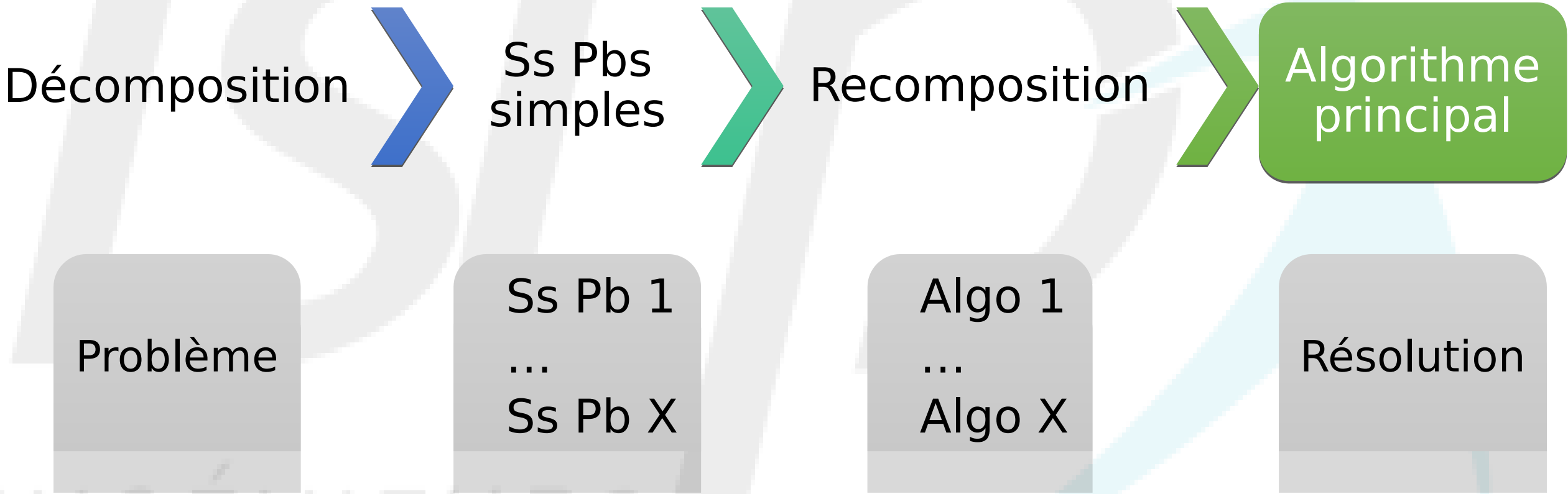
Fin

L'ALGORITHME

DE QUOI S'AGIT-IL?



Méthodologie de conception: analyse descendante



objectifs de la Méthodologie de conception

Modularité:

- 1 problème simple = 1 algorithme simple
- réutilisable

Lisibilité:

- mise en page
- commentaires
- description

Complexité:

- enchaînements
- mesure de la durée d'exécution
- mesure de l'espace mémoire

Méthodologie de conception: les structures

Séquentielle

ordonnancement des instructions

Conditionnelle

bloc d’instructions à exécuter selon circonstances

Itérative

bloc d’instructions à exécuter plusieurs fois

Syntaxe globale

- **Écriture**

ALGORITHME *<Nom>*
FONCTIONS_UTILISEES *<Fonctions>*
VARIABLES
<Déclaration des variables>
DEBUT_ALGORITHME
<Bloc Instructions>
FIN_ALGORITHME

Exemple

```
ALGORITHME Tension
FONCTIONS_UTILISEES
VARIABLES
   P EST_DU_TYPE NOMBRE
   I EST_DU_TYPE NOMBRE
   U EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
   P PREND_LA_VALEUR 4400
   I PREND_LA_VALEUR 20
   U PREND_LA_VALEUR P / I
   AFFICHER U
FIN_ALGORITHME
```

AlgoBox

Site web : https://www.xm1math.net/algobox/index.html
Téléchargement : https://www.xm1math.net/algobox/download.html
Documentation : https://www.xm1math.net/algobox/doc.html

LES VARIABLES

Les Variables

- **Description**

Usage

stocker des valeurs

définitives

intermédiaires

où?

mémoire Pc

Déclaration

Nom

simple, sans espace, sans ponctuation, pas d’accents

exemple

```
VARIABLES
    qtePiece EST_DU_TYPE NOMBRE
    nom_piece EST_DU_TYPE CHAINE
```

Les Variables

- **Descriptions**

Nombre

Nombre entiers
Nombres à virgule flottante

Chaine alphanumérique (entre guillemets: "chaine"

texte
caractères
chaîne
nombre sous forme de texte (code postal)

Liste

Liste de nombres

exemples

- A EST DE TYPE NOMBRE
  PI EST DE TYPE NOMBRE
- A PRENDS LA VALEUR 10
- PI PRENDS LA VALEUR 3.14

- CAR EST DE TYPE CHAINE
- STR EST DE TYPE CHAINE
- CAR PRENDS LA VALEUR « A »
- STR PRENDS LA VALEUR « Hello »

- TAB EST DE TYPE LISTE
- TAB[1] PRENDS LA VALEUR 12
- TAB[2] PRENDS LA VALEUR 25

- **Syntaxe**

Affectation

"prend la valeur de ..."

Exemples

- nbHeures EST DE TYPE NOMBRE
- nbHeures PRENDS LA VALEUR 15.25

- Famille EST DE TYPE CHAINE
- Famille PRENDS LA VALEUR "Moteurs"

- N EST DE TYPE NOMBRE
- N PRENDS LA VALEUR 10

Exercices

Valeurs des variables au cours et à la fin de l'exécution

Exercices Variables

- **Exo 1:**

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR 4
  B PREND_LA_VALEUR A
+ 5
  A PREND_LA_VALEUR 8
FIN_ALGORITHME
```

exo_affectation01.alg

Exo 2:

```
FONCTIONS_UTILISEES
VARIABLES
    A EST_DU_TYPE NOMBRE
    B EST_DU_TYPE NOMBRE
    C EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    A PREND_LA_VALEUR 2
    B PREND_LA_VALEUR 6
    C PREND_LA_VALEUR A+B
    A PREND_LA_VALEUR 3
    C PREND_LA_VALEUR B-A
FIN_ALGORITHME
```

exo_affectation02.alg

Exercices Variables

- **Exo 3:**

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR 1
  B PREND_LA_VALEUR A+3
  A PREND_LA_VALEUR A+3
  B PREND_LA_VALEUR 4-A
FIN_ALGORITHME
```

exo_affectation03.alg

Exo 4:

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR 6
  B PREND_LA_VALEUR 4
  A PREND_LA_VALEUR B
  B PREND_LA_VALEUR A
FIN_ALGORITHME
```

exo_affectation04.alg

Exercices Variables

- **Exo 5:**

Écrire l'algorithme qui permet d'échanger les valeurs de 2 entiers A et B

exo_affectation05.alg

Exo 6:

Écrire l'algorithme qui permet d'échanger les valeurs de 3 entiers A,B et C (B à A, C à B et A à C)

exo_affectation06.alg

Exercices Variables

• Exo 7:

Pour Ajouter A et B qui sont de type différents, il faut convertir la chaine A en nombre.

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE CHAINE
  B EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR "100"
  B PREND_LA_VALEUR 200
  //Convertir la CHAINE A en NOMBRE
  B PREND_LA_VALEUR parseInt(A) +B
  AFFICHER* B
FIN_ALGORITHME
```

exo_affectation07.alg

Exo 8:

Concaténer des variables de type différents.

```
FONCTIONS_UTILISEES
VARIABLES
  jour EST_DU_TYPE NOMBRE
  mois EST_DU_TYPE CHAINE
  fete_nationale EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
  jour PREND_LA_VALEUR 14
  mois PREND_LA_VALEUR "Juillet"
  //Concaténer les parties du message
  fete_nationale PREND_LA_VALEUR jour.toString() + " " + mois
  AFFICHER* fete_nationale
FIN_ALGORITHME
```

exo_affectation08.alg

LECTURE ET ECRITURE

Lecture / écriture

Lecture: Récupérer une valeur provenant de l'extérieur (clavier)

Instruction: LIRE

Écriture: Afficher une valeur (écran)

Instructions : AFFICHER, AFFICHERCALCUL

exemple

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE A
  LIRE B
  AFFICHERCALCUL A+B
FIN_ALGORITHME
```

Exercices Lesture/Ecriture

- **Exo 09:**

Programme qui demande un nombre puis affiche le carré de ce nombre sous la forme:" "le carré de ce nombre est ...«

exo_affectation09.alg

Exo 10:

Programme de caisse qui affiche le montant à payer, le montant reçu et le reste à rendre

exo_affectation10.alg

LES STRUCTURES

SEQUENTIELLES (ALTERNATIVES):

les conditions (tests)

Sequentielle

- **SI ALORS SINON**

```
SI <CONDITION>
ALORS
   DEBUT_SI
   < instruction1>
   FIN_SI
[SINON
   DEBUT_SINON
   < instruction2>
   FIN_SINON]
```

[SINON]: facultatif

exemple

```
FONCTIONS_UTILISEES
VARIABLES
  temperature EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  SI (temperature < 50) ALORS
    DEBUT_SI
    AFFICHER "OK"
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER "Arrêt système"
      FIN_SINON
FIN_ALGORITHME
```

condition01.alg

Conditions

- **Condition=comparaison**

1.une valeur

2.un opérateur de comparaison

==

!=

<, <=

>, >=

3.une autre valeur

exemple

```
FONCTIONS_UTILISEES
VARIABLES
  A EST_DU_TYPE CHAINE
  B EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR "A"
  B PREND_LA_VALEUR "B"
  SI (A > B) ALORS
    DEBUT_SI
    AFFICHER "A est plus grand que B"
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER* "B est plus grand que A"
      FIN_SINON
FIN_ALGORITHME
```

condition02.alg

Conditions composées

- **Et / Ou**

ET : les 2 conditions doivent être Vraies pour que le tout soit Vrai

OU : 1 condition doit être Vraie pour que le tout soit Vrai

Exemple

```
SI (temperature < 50 ET pression < 180) ALORS
    DEBUT_SI
    AFFICHER* "OK"
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER* "Arrêt système"
      FIN_SINON

  SI (temperature >= 50 OU pression >= 180)
ALORS
    DEBUT_SI
    AFFICHER* "Arrêt système"
    FIN_SI
    SINON
      DEBUT_SINON
      AFFICHER* "OK"
      FIN_SINON
```

condition03.alg

Exercices

conditions logiques

Exercices Logiques

• Exo 1:

Faire saisir 2 nombres différents et vérifier si l'un est strictement plus grand que l'autre

solutions\exo_condition01.alg

Exo 2:

Faire saisir 2 nombres et vérifier si

- **Ils sont égaux**
- **Ils sont inférieurs à 10**
- **lequel est strictement plus grand que l'autre**

solutions\exo_condition02.alg

Exercices Logiques

• Exo 3:

Faire saisir 1 température et afficher l'état du système tel que:

- **correct si <50°C**
- **à surveiller si >=50°C et <100°C**
- **Arrêter système si >=100°C**

solutions\exo_condition03.alg

Exo 4:

Une machine est en maintenance selon:

si le nb de jours depuis la dernière date de maintenance >35

si son nbre d'heures d'utilisation>3000

sa production<2000 ou >10000 depuis la date de dernière maintenance

Quelles questions doit poser le programme? et comment va-t-il résoudre ce problème?

solutions\exo_condition04.alg

STRUCTURES

ITÉRATIVES: les boucles

Les boucles

Usage

répéter une série d’instructions
possibilité d’imbriquer les boucles

Utilisation: (exemples)

remplir un tableau
parcourir des champs de formulaires
itérer sur des milliers de lignes très rapidement
trier des listes

2 types:

nombre d’itérations connues à l’avance gérées par un compteur
ex: "Pour i=1 jusqu’à 3 , enrouler film autour palette »

la boucle s’arrête quand une condition est remplie, gérée par un booléen
ex: "Tant que le MDP <> MotDePasseSaisi , ressaisir"

Pour...jusqu'à

Syntaxe

```
POUR index ALLANT_DE valeurDebut A valeurFin
  DEBUT_POUR
  Instructions
  FIN_POUR
```

Exemple

```
FONCTIONS_UTILISEES
VARIABLES
  jour EST_DU_TYPE NOMBRE
  production_jour EST_DU_TYPE LISTE
  total EST_DU_TYPE NOMBRE
  moyenne EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR jour ALLANT_DE 1 A 7
    DEBUT_POUR
    AFFICHER "Jour "
    AFFICHER* jour
    LIRE production_jour[jour]
    FIN_POUR
  total PREND_LA_VALEUR 0
  POUR jour ALLANT_DE 1 A 7
    DEBUT_POUR
    total PREND_LA_VALEUR total + production_jour[jour]
    FIN_POUR
  moyenne PREND_LA_VALEUR total / 7
  AFFICHER* moyenne
FIN_ALGORITHME
```

boucle01.alg

Tant que… Jusqu'à

Syntaxe

Pour tester au moins une fois la condition

TANT_QUE <expression booléenne> **FAIRE**
DEBUT_TANT_QUE

Instructions

FIN_TANT_QUE

boucle02.alg

Exemple

```
FONCTIONS_UTILISEES
VARIABLES
  mot_passe EST_DU_TYPE CHAINE
  essai_password EST_DU_TYPE CHAINE
  valide EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  valide PREND_LA_VALEUR 0
  mot_passe PREND_LA_VALEUR
"SECRET"
  TANT_QUE (valide == 0) FAIRE
    DEBUT_TANT_QUE
    LIRE essai_password
    SI (essai_password == mot_passe)
ALORS
      DEBUT_SI
      AFFICHER* "OK"
      valide PREND_LA_VALEUR 1
      FIN_SI
      SINON
        DEBUT_SINON
        AFFICHER* "Echec"
        FIN_SINON
    FIN_TANT_QUE
FIN_ALGORITHME
```

Exercices

Boucles

• Exo 1

Lire les prénoms et les notes des élèves de la classe, tant que le prénom saisi est différent de:

« FIN ». Vérifier que la note saisie soit comprise entre 0 et 20.

Afficher ensuite:

- la moyenne de la classe
- la meilleure note de la classe et le prénom correspondant.
- la moins bonne note de la classe et le prénom correspondant.

solutions\exo_boucle01.alg

Exo 2

Lire le nombre de joueurs et le nombre de tirages pour paramétrer le jeu.

A chaque tirage, chaque joueur jette 2 dés. Vous utiliserez la fonction ALGOBOX_ALEA_ENT(p,n) qui renvoie un entier pseudo-aléatoire compris entre p et n. Le joueur disposant du plus grand total gagne.

Afficher le joueur gagnant pour chaque tirage.

solutions\exo_boucle02.alg

les Tableaux

par l'exemple

Description

Tableau

variable pouvant stocker N éléménts

de type nombre

déclaration:

tableau EST_DU_TYPE LISTE

pour les remplir:

affectation simple: tableau[2] PREND_LA_VALEUR 5

itération:

```
POUR index ALLANT_DE 1 A 10
        DEBUT_POUR
        tableau[index] PREND_LA_VALEUR index
        FIN_POUR
```

• Exemple

```
FONCTIONS_UTILISEES
VARIABLES
  hasard EST_DU_TYPE LISTE
  index EST_DU_TYPE NOMBRE
  total EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  // renseigner la liste avec nombre aléatoire
  POUR index ALLANT_DE 1 A 10
    DEBUT_POUR
    hasard[index] PREND_LA_VALEUR ALGOBOX_ALEA_ENT(0,100)
    FIN_POUR
  // Calcul moyenne
  total PREND_LA_VALEUR 0
  POUR index ALLANT_DE 1 A 10
    DEBUT_POUR
    total PREND_LA_VALEUR total + hasard[index]
    FIN_POUR
  AFFICHER "Moyenne "
  AFFICHERCALCUL* total / 10
FIN_ALGORITHME
```

tableau01.alg

• Exo 1

Lire le nombre de joueurs et le nombre de tirages pour paramétrer le jeu.

A chaque tirage, chaque joueur jette 2 dés. Vous utiliserez la fonction ALGOBOX_ALEA_ENT(p,n) qui renvoie un entier pseudo-aléatoire compris entre p et n. Le joueur disposant du plus grand total gagne.

Afficher le joueur gagnant pour chaque tirage et le joueur ayant gagné le plus grand nombre de tirages.

Variante : paramétrer le nombre de dés.

solutions\exo_tableau01.alg

Exo 2

Refaire l'exercice 1 en 2 phases:

- Phase 1: Réaliser tous les tirages et stocker les données dans un tableau à 2 dimensions.
- Phase 2: Analyser les données pour afficher les informations demandées.

Astuce: AlgoBox ne connaît que les tableaux à 1 dimension. Vous pouvez simuler un tableau à 2 dimensions.

https://www.xm1math.net/algobox/doc.html#SECTION9

solutions\exo_tableau02.alg

FONCTIONS

Description

Usage

Algorithme prédéfini livré avec le langage (comme les calculettes)
librairie mathématique (trigo, géo, finance): sin, cos, loi.normale, etc.
traitements de chaînes de caractères (extraction, recherche,...)

Utilisation

pendant toute la programmation
en appel « extérieur » pour effectuer un traitement intermédiaire

Syntaxe

un nom
2 parenthèses
de 0 à N arguments séparés par virgule(s)

nomFonction(*[Argument1]*,*[Argument2]*, *[Argument3]*,...)

Exemples fonctions

- **Principales fonctions textes**

taille(chaine): nb caractères

Transformer un nombre en chaîne

Transformer une chaîne en nombre

extraire(chaîne, départ, n): extrait une partie de la chaîne commençant au caractère de départ et long de n caractères

extraire_ascii(chaîne, pos): retourne le code ASCII du caractère à la position pos

caractere_ascii(code_ascii): renvoie le caractère correspondant au code ASCII

Equivalent VBA

- Chaine.length
- Nombre.toString()
- parseInt(chaine)
- Chaine.substr(debut, nombre)

chaine.charCodeAt(pos)

chaine.fromCharCode(ascii)

Fonctions

• Exo1

- Compter le nombre de caractères d’une phrase sans les espaces
- solutions\exo_fonctions01.alg

• Exo2

- Compter le nombres de voyelles d’une phrase
- solutions\exo_fonctions02.alg

Fonctions

• Exo3

- on souhaite inviter l'utilisateur à saisir une date au format jjmmaa mais il faudra l'afficher au format classique jj/mm/aaaa

- solutions\exo_fonctions03.alg

• Exo4

Calculer une approximation de PI en utilisant la série (ou formule) de Madhava-Leibniz

Demander à l'utilisateur le plus grand dénominateur n pour le calcul.

PI=4×(1−1/3+1/5−1/7+1/9−1/11+1/n−….)

solutions\exo_fonctions04.alg