ALGORITHMIE

Objectif du cours

Démo application VBA

Plan du cours

Cours 1: Méthode de réalisation d'un algorithme. Introduction au pseudo code.

Cours 2: mise en œuvre d'algorithme en pseudo code avec AlgoBox. Découverte de l'environnement de développement VBA.

Cours 3: Ecriture d'algorithmes en VBA

Cours 4: Découverte des formulaires en VBA.

Cours 5: Réalisation d'une application avec des formulaires en VBA.

Evaluation

Evaluation sur la base d'un DM réalisé à partir d'un cahier des charges :

- Travail Effectué en groupe de 4 personnes.
- Remise d'un dossier de pseudo code.
- Remise d'une application en VBA.

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 »

Enchainements

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 enchainement au rangement de mon univers Vélo?

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

- définition de l'univers
- enumération des objets par univers

Puis-je appliquer cet enchainement 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, éxécuté 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 enchainements 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:

- Convention d'instructions organisées

Instruction:

- consigne formulée dans un langage de programmation selon un code

Programmation:

- permet de traduire l'algorithme dans un langage adapté à l'ordinateur

Pseudo code:

- 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 —*multiplateforme*
- Vba, Php…

Compilateur:

- le programme est traduit en une seule fois et stocké dans un exe
- 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, Alors, Sinon Tant Que, Jusqu'à…

Opérateurs

⑩← (affectation)

- +, - ,*, /
- =, ( )
- <, >, <>, <=, >=
- ^ (puissance)
- & (concaténation)
- Mod (reste division entiere)
  - 6 mod 3=0 mais 6 mod 4=2
  - permet de savoir si le diviseur est un multiple
  - 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

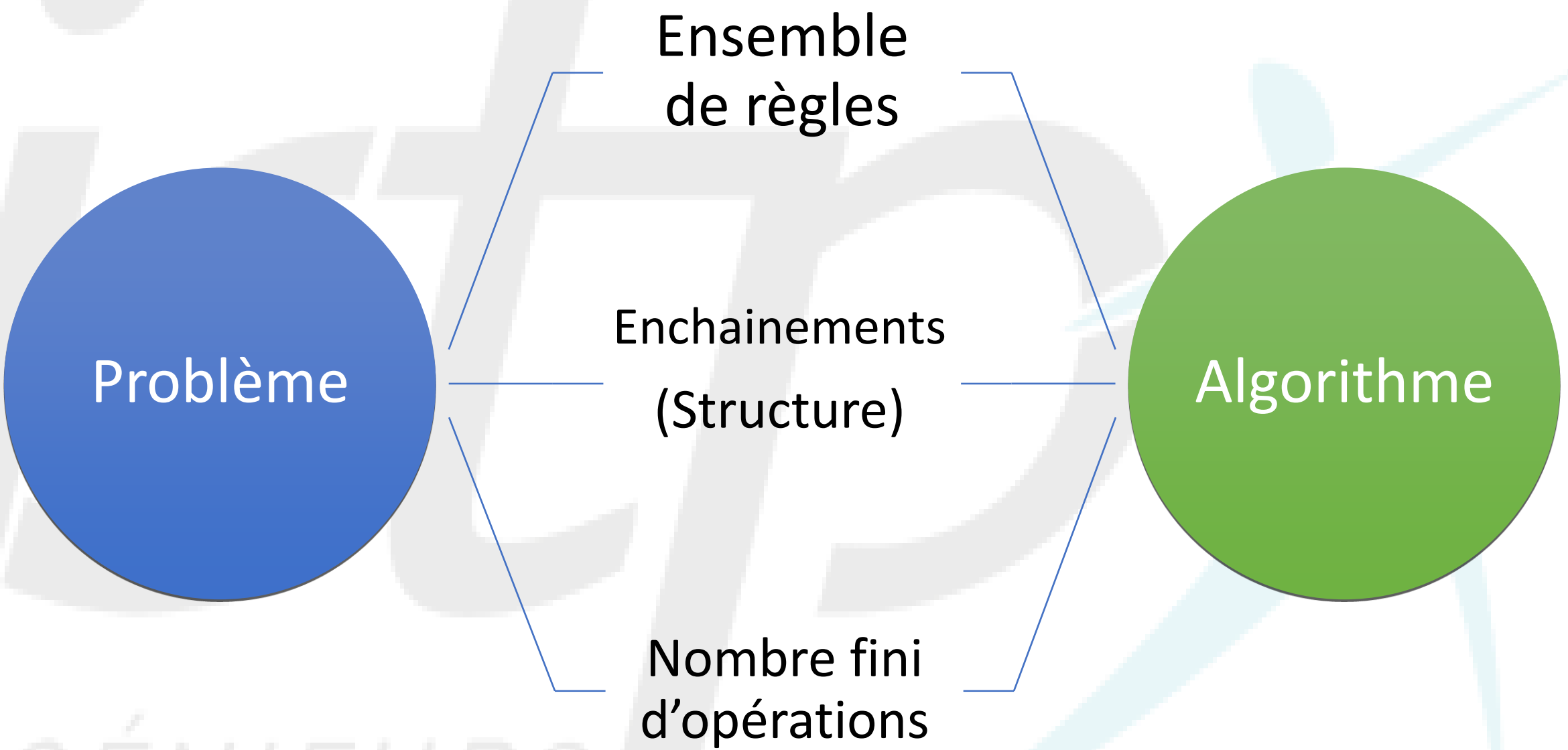
FIN TANT QUE

Fermer agenda

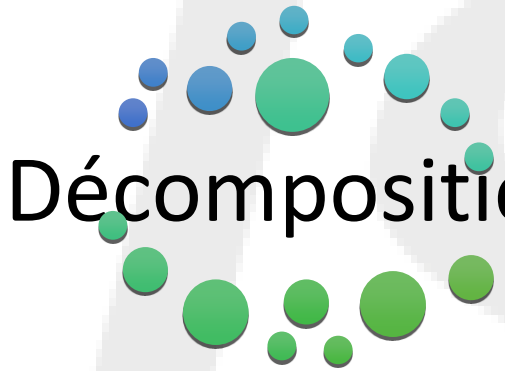
Fin

L'ALGORITHME

DE QUOI S'AGIT-IL?



Méthodologie de conception: analyse descendante



Décomposition → Ss Pbs simples → Recomposition → Algorithme principal

Problème	• Ss Pb 1 • … • Ss Pb X	• Algo 1 • … • Algo X	Résolution

objectifs de la Méthodologie de conception

Modularité:

- 1 problème simple = 1 algorithme simple
- réutilisable

Lisibilité:

- mise en page
- commentaires
- déscription

Complexité:

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

Méthodologie de conception: les structures

Sequentielle

ordonnancement des instructions

Conditionnelle

bloc d'instructions à executer 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

- **Description**

Usage

- stocker des valeurs
  - définitives
  - intermédiaires
- où?
  - mémoire Pc

Déclaration

- Nom
  - simple, sans espace, sans ponctuation

exemple

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

Les Variables

- **Descriptions**

Taille: optimisée pour économiser ressources PC

numérique

- byte (0 à 255)
- entier simple (-32 768 à 32767)
- entier long (-2147483648 à 2147483647)
- réel simple (-3,4x10^{38} à 3,4x10^{38})
- réel double (-1.79x10^{308}"à 1.79x10^{308})

monétaire

- 2 chiffres après la virgule, devise

date

- numéro de série
- format

alphanumérique (entre guillemets: " ")"

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

booléen

- vrai/faux
- oui/non
- 0/1

exemples

- Var A :byte
- Var coefA: réel double
- Var prixHT: monétaire
- Var dateEffet: date (jj/mm/yyyy)
- Var indiceBruit: caractères
- Var nomFamille: string,
  calculAutorise: bool

Les Variables

- **Descriptions**

Portée-Durée

accessibilité

- publique: Projet
- privée: Zone de déclaration

conservation

- constante: initialisée 1 fois
- static: conserve la valeur précédente

Exemples

- publique si déclarée avant la procédure

- Const Entier nbMachines←10
- Static Entier tentativesMDP←3

Les Variables

- **Descriptions**

Spéciales

tableaux

- spécialisés
- variant

objets

- sélections
- plage
- feuille Excel, classeur, …

énumératives

- regroupement de plusieurs constantes ordonnées

structurées

- regroupement de types différents

intervalles

- définition de valeurs ordonnées et bornées au mini et maxi

Exemples

- Tableau Entier: t(10) \\10 cases d'entiers

  Tableau String: tab(3) \\3 cases de textes

  Tableau Entier tab(5,6) \\(5 lignes, 6 colonnes) d'entiers

  Tableau tab \\variant, taille non définie

- monClasseur: objet classeur Excel
- Enum droits

  lecture=-1

  ecriture=0

  lecture_ecriture=1

  FinEnum \\ lecture < ecriture < lecture_ecriture
- Struct Famille

  Var Nom: string

  Var Code: string

  Var nbSousEnsembles: byte

  Var nbArticles: entier

  Var statutActif: bool

  FinStruct
- JourOuvrés: Lundi..Vendredi

  Pourcentage: 0..100

  IndiceBruit: A..E \\descriptif, inconnu en vba hormis range(selection), ou tableau

- **Syntaxe**

Affectation

- "prend la valeur de …"

Exemples

- Var nbHeures: double
  nbHeures ← 15.25
- Var Famille: string
  Famille ← "Moteurs"
- Var n: entier
  n←10
  Tableau Entier: t(n) \\10 cases

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
  A PREND_LA_VALEUR parseInt(A)+B
  AFFICHER* A
FIN_ALGORITHME
```

exo_affectation07.alg

Exo 8:

Concaténer des variables de type différents.

```
FONCTIONS_UTILISEES
VARIABLES
  jour_debut EST_DU_TYPE NOMBRE
  mois EST_DU_TYPE CHAINE
  fete_nationale EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
  jour_debut PREND_LA_VALEUR 14
  mois PREND_LA_VALEUR "Juillet"
  //Concatener les parties du message
  fete_nationale PREND_LA_VALEUR
jour_debut + " " + 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

lecture/ecriture

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 Vrai pour que le tout soit Vrai

OU

- 1 condition doit être Vrai 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

- **différent de l'autre**
- **<10**
- **l'un 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 booleenne> 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

- **Exo 1**

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

« X AE A-XII ». 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.

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 connait 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 chaines de caractères (extraction, recherche,…)

Utilisation

- pendant toute la programmation
- en appel « exterieur » 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

gauche(chaine, n): n car depuis gauche

droite(chaine, n): idem depuis droite

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

remplacer(chaine,texte1,texte2): remplace dans une chaine la chaine trouvée texte1 par la texte2

trouve(chaine1,chaine2): renvoie la position du caractère qui commence la chaine dans la chaine1 ou erreur si pas trouvé

Equivalent VBA

- len(string)
- left(string, n)
- right(string,n)
- mid(string,n1,n2)
- replace(string, t1,t2)
- Instr(string1,string2)

Exemples fonctions

- **Conversion**

Asc("caractère"): renvoie le nombre auquel il correspond dans table ASCII

Chr(nombre):renvoie le caractère Ascii

Cnum(texte): convertit une chaine en numerique

Cint(nombre): renvoie la partie entiere (en fait convertit en entier)

Cdbl(nombre):convertit un entier en double

Cstr(nombre):convertit un nombre en texte

texte(Chaine,format): renvoie le format defini de la chaine

Equivalent VBA

- Asc("A")=65
- Chr(97)="a"
- Val("24 kg")=24 <> Val("kg 24")=0
- Cint(25.32)=25
- Cdbl(25)=25.00
- Cstr(25)="25"
- Format("22/09/2019", "yyyy\mm\dd")="2019\09\22"

Fonctions

- **Exo1**

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

- **Exo2**

- Compter le nb de voyelles d'une phrase
- solutions\exo_fonctions02.alg

- **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

FONCTIONS PERSONNALISEES - VBA

Fonctions personnalisées

Le compilateur repère le début du programme: **la procédure Principale**

appelée par le système d'exploitation	n'existe pas en VBA (≈ evenement Workbook_Open si on veut lancer à l'ouverture du fichier)

Procédure principale suit les instructions: **Déroulement**

si actions répétitives de traitements: **Sous- procédures**	si actions répétitives avec attente d'un retour: **Fonctions personnalisées**

Le programme devient **Modulaire**:
Transmission d'**arguments** (paramètres) **typés**

procédures ou fonctions	Retour typé du traitement: fonctions seulement

- **Procédures sans et avec arguments**

```
Procedure NomProcedure()
        Instructions
FinProcedure


Procedure NomProcedure( Arg1, [Arg2] )
    SI Arg1=… ALORS \\exemple...
        instructions
    FINSI
FinProcedure
```

Exemple

```
Procedure EliminerLignes()
    Pour lig←50 à 100
        supprimer.ligneEntiere(lig)
    FinPour
FinProcedure

Procedure ElimineCertainesLignes(str en string)
    Pour lig←50 à 100
        SI gauche(cellule(lig,2),3)=str ALORS
            supprimer.ligneEntiere(lig)
        FinSI
    FinPour
FinProcedure
```

Fonctions

- **Fonctions sans et avec arguments**

```
Fonction NomFonction () en type
        Instructions
FinFonction

Fonction NomFonction(Arg1, [Arg2]) en type
    SI Arg1=… ALORS \\exemple...
        instructions
        NomFonction←…
    SINON
        NomFonction← …
    FINSI
FinFonction
```

Exemple

```
Fonction AdditionnerPlage() en long
Var calc: double
    Pour lig←50 à 100
        calc←calc+cellule(lig,1)
    FinPour
     AdditionnerPlage←calc
FinFonction

Fonction testDroits(str en string) en bool
    Selon str
        cas "Admin"
            testdroits←Vrai
        cas "User"
            tesdroits←Faux
    FinSelon
FinFonction
```

Exercices Fonctions

- **Exo1**

- Ecrire un prog qui va afficher un message d'avertissement au moyen d'une sous procedure

```
Debut
    avertissement
Fin
procedure avertissement()
    Ecrire "La valeur saisie..."
FinProcedure
```

- **Exo2**

- idem précédent mais qui permettra d'afficher un message différent à chaque appel

```
Var Msg: string
Debut
    Msg←"Erreur de…"
    avertissement(Msg)
    …
    Msg←"La valeur saisie…"
    avertissement(Msg)
    …
Fin
Procedure avertissement(strMsg: string)
    Ecrire strMsg
FinProcedure
```

Exercices Fonctions

- **Exo3**

- Ecrire une fonction qui retournerait la tension d'après les valeurs de Puissance et d'intensité proposées

```
Var P, I, U: Entiers reels
Debut
    P←4400
    I←20
    U←Tension(P,I)
Fin

Fonction Tension(intP:Entier, intI:Entier) en Entier
    SI intI=0 ALORS          \\si oubli, Erreur!
        Tension←0
    SINON
        Tension←intP/intI
    FinSi
FinFonction
```

Exercices Fonctions

- **Exo4**

- Ecrire une fonction qui verifierait si une valeur texte appartient à un tableau en affichant vrai ou faux

```
Var saisie: string
Var Tableau String: tab (10)
Debut
    tab(0)←("test")
    tab(…)←("…")
    tab(9)←("essai")
        Ecrire "entrez un texte"
            lire saisie
        Ecrire verifieTexte(saisie,tab)
Fin

Fonction verifieTexte(strSaisie: string, tablo: variant) en boleen
    verifieTexte←Trouve(strSaisie,joindre(tablo))>0
FinFonction

\\joindre: fonction qui concatène tous les éléments d'un tableau; VBA: Join
```

- **Exo**

Créer 1 fonction récursive qui permet d'entrer 2 nombres en paramètres, le second étant l'exposant du premier; renvoyer le résultat du calcul depuis la procédure d'appel.

expo(2,5)=32
(2*2*2*2*2)

```
Algo Appel_exposant
Var Nb, Ex: long
Debut
lire Nb: lire Ex
        Ecrire exposant(Nb,Ex)
Fin

Fonction exposant(Nbr:long,Expo:long) en long
Si Expo >=1 Alors
   exposant=exposant(Nbr,expo-1)*Nbr
Sinon
   exposant=1     \\permet de sortir de la pile
FinSi
FinFonction
```