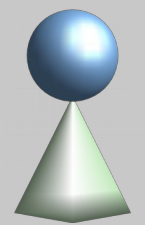
C9- Algorithme

Introduction

- On veut parfois faire des tâches / programmes pour lesquels on n’a pas de logiciel suffisant.

- On peut soi-même créer le logiciel / code nécessaire.



- Utilisation d’un langage de programmation.

Introduction

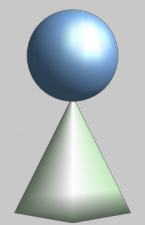
Javascript

C++

PHP

Delphi

Java



C#

VBA

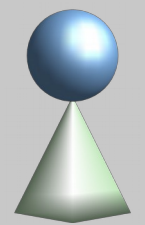
Perl

Cobol

Python

Introduction

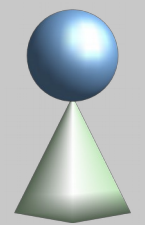
- VBA : Visual Basic for Applications
- Langage de programmation adapté au tableur
- Dans libre office : LibreOffice Basic et … VBA



Introduction

- Parfois Excel est très bien et le code serait tout à fait inutile

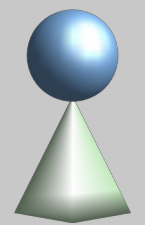
- Des problèmes que Excel peut résoudre mais qui sont beaucoup plus simples à lire et à modifier dans du code.



- Des problèmes qu’Excel ne pourrait pas résoudre.

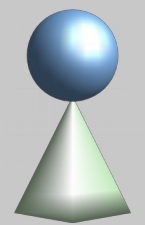
Introduction

- Des langages : des briques à assembler. Les briques peuvent être différentes en fonction des langages.
- Briques = fonctionalités de base
- Différents langages => différentes manières de construire un mur
- Un langage donné ne peut pas tout faire



Introduction

- Tout ce qui est fait par un ordinateur est codé sous forme de lignes de code.



- On parle de code source.

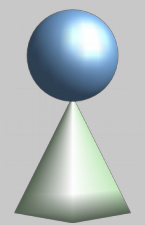
Des exemples d’algorithmes

Algorithme de calcul
de l’imposition
d’un individu
avec des règles complexes.

Algorithme de recherche
d’occurrences selon
des critères données

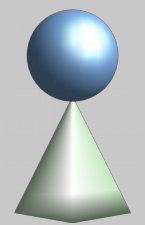
Algorithme d’analyse
financière

Algorithme de
simulations
de comportements
boursiers



Introduction

Dans les slides, on emploie le pseudo-algo ou VBA. Dans les exercices ou en examen : VBA.



Un exemple d'algorithme

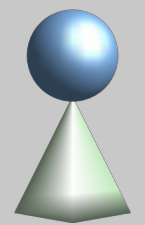
Un algorithme de calcul d'imposition

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← baseImp*0,13
        sinon
        imp ← (baseImp-20000)*0.25+20000*0.13
        fin si
afficher(imp)
FIN
```

Le nom de l'algo / procédure / macro

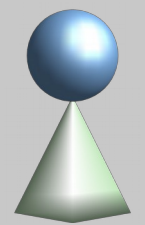
La déclaration des variables

Le corps de l'algo



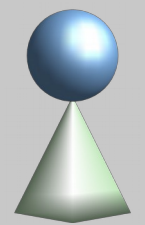
Plan

1) **Variables**
2) Entrées et sorties
3) Instruction conditionnelle
4) Boucle
5) Fonctions et procédures
6) Objets d’un tableur



Variables

- Les algorithmes utilisent des variables.
- Exemples
- Chaque variable doit être déclarée / typée avant d'être utilisée.

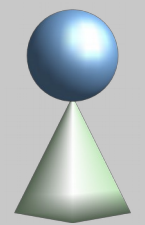


```
Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier
```

Affectation

Une fois que l'on a déclaré une variable, on peut lui affecter une valeur : *x ← 100*

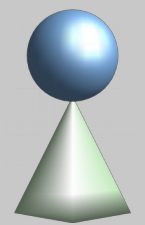
```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← baseImp*0,13
        sinon
        imp ← (baseImp-20000)*0.25+20000*0.13
        fin si
afficher(imp)
FIN
```



Types

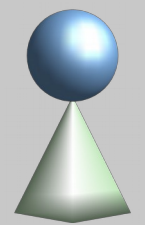
Les types de variables possibles :

- Entier
- Réel
- Chaîne de caractères
- Date
- Booléen



Noms de variable

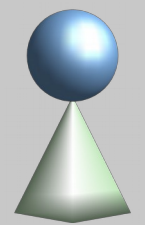
- Pas d’espace
- Le nom de variable ne peut pas commencer par un nombre
- On essaie de respecter les conventions / de prendre des noms signaficatifs.



Opérations sur les variables

- Les types numériques supportent des opérateurs standards : +, -, *, /

- On peut parenthèser avec les mêmes règles qu'en mathématiques.



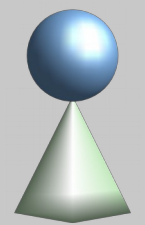
```
Déclaration des variables
x,y : reel
i,j : entier

DEBUT
...
```

```
x ← 0.6
i ← 10
y ← x*i
j ← (y+i)*5
i ← i+1
…
FIN
```

Variables en VBA

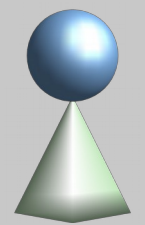
- Déclaration avec Dim
- Les types : Double, String, Boolean, Integer, Date
- Opérateurs.
- Localité des variables

```
Sub prog1()
Dim n as Integer, x as Double
n=7
x=n*5
End Sub

Sub prog2()
Dim x as Double, y as double, str as String
x=4.3
str="rdorat@paideia.be"
y=x*10
End Sub
```



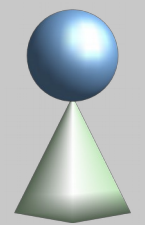
Plan

1) Variables
2) **Entrées et sorties**
3) Instruction conditionnelle
4) Boucle
5) Fonctions et procédures
6) Objets d’un tableur



Entrées et sorties

- Interaction avec l’utilisateur
- Entrée : l’utilisateur donne une valeur qui sera utilisée dans l’algorithme
- Sortie : l’algorithme renvoie une valeur à l’utilisateur. Sans opération de sortie, le résultat de l’algorithme est "perdu"

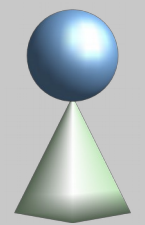


Entrées

```
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
```

Différentes formes d'Entrées :

- Saisie d'une valeur par l'utilisateur au clavier
- Sélection par l'utilisateur d'une valeur dans un formulaire
- Lecture d'une valeur dans un fichier
- Capture de la voix de l'utilisateur
- ...

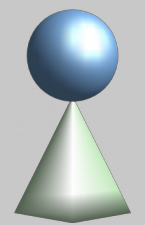


Sorties

affiche(imp)

Différentes formes de Sorties :

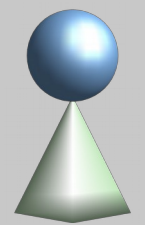
- Affichage sur l’écran
- Impression
- Son
- Envoi d’un mail
- Ecriture dans un fichier
- ...



Entrées et sorties en VBA

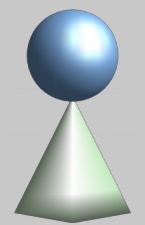
- Une version simplifiée dans un premier temps
- Msgbox pour afficher (sortie)
- Inputbox pour lire (entrée)

```
Sub prog3()
Dim n as Integer, x as Double
n=InputBox("Donner une valeur")
n=n*2
Msgbox("Le double de votre valeur " & n)
End Sub
```



Plan

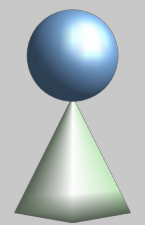
1) Variables
2) Entrées et sorties
3) **Instruction conditionnelle**
4) Boucle
5) Fonctions et procédures
6) Objets d’un tableur



Instruction conditionnelle

- Certains éléments du code ne s'exécutent que sous certaines conditions

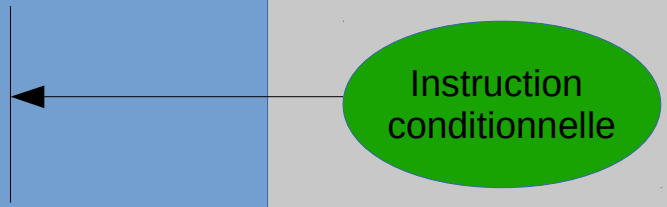
- Ex : si on a une connexion Internet, on affiche la page, sinon, un erreur de connexion.

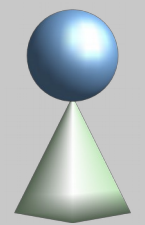


- Ex : si on a un camion de plus de 7 tonnes, le cacul de son itiniréraire pour aller de Grenoble à Gap ne sera pas le même que pour camion de 3 tonnes

Instruction conditionnelle

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← baseImp*0,13
        sinon
        imp ← (baseImp-20000)*0.25+20000*0.13
        fin si
afficher(imp)
FIN
```





Comment peut-on alors comprendre ce code ?

Instruction conditionnelle

L’instruction conditionnelle est très souple :

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp > 20000 alors
        imp ← baseImp*0,13
        fin si
afficher(imp)
FIN
```

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← 0
        sinon si baseImp<40000 alors
        imp ← baseImp*0,13
        sinon
        imp ← baseImp*0,2
        fin si
afficher(imp)
FIN
```

Instruction conditionnelle

Deux codes : font-ils la même chose ? Quel est le meilleur ?

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← 0
        sinon si baseImp<40000 alors
        imp ← baseImp*0,13
        sinon
        imp ← baseImp*0,2
        fin si
afficher(imp)
FIN
```

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← 0
        sinon si baseImp<40000 et baseImp>=20000 alors
        imp ← baseImp*0,13
        sinon
        imp ← baseImp*0,2
        fin si
afficher(imp)
FIN
```

Instruction conditionnelle

Deux codes : font-ils la même chose ?

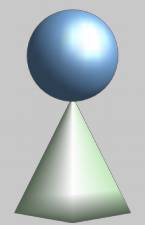
```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← 0
        sinon si baseImp<40000 alors
        imp ← baseImp*0,13
        sinon
        imp ← baseImp*0,2
        fin si
afficher(imp)
FIN
```

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp > 20000 alors
        imp ← 0,13
        sinon si baseImp>40000
        imp ← baseImp*0,2
        sinon
        imp ← 0
        fin si
afficher(imp)
FIN
```

Instruction conditionnelle

- A noter les opérateurs de comparaisons : ">", "<", "<>", "=", ">=", "<=". S'appliquent à tous les types. Expression booléenne

- Enchainement sur les instructions conditionnelles "et" / "ou" / "non".



- On ne peut pas écrire 5<a<=8

- Priorité du 'et" sur le 'ou'

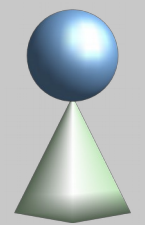
Instructions conditionnelles en VBA

```
Algo calcul_Impot

Déclaration des variables
revenu, baseImp, imp : reel
nbPerso : entier

DEBUT
revenu ← Lire("Donner le revenu")
nbPers ← Lire("Donner le nb de personnes du foyer")
baseImp ← (revenu-revenu*0.1)/nbPers
        si baseImp < 20000 alors
        imp ← 0
        sinon si baseImp<40000 alors
        imp ← baseImp*0,13
        sinon
        imp ← baseImp*0,2
        fin si
afficher(imp)
FIN
```

```
Sub calcul_Impot()
Dim revenu as double, baseImp as double, imp as double
Dim nbPerso as Integerr



revenu=Inputbox("Donner le revenu")
nbPers=Inputbox("Donner le nb de personnes du foyer")
baseImp=(revenu-revenu*0.1)/nbPers
        if baseImp < 20000 then
        imp=0
        elseif baseImp<40000 then
        imp=baseImp*0,13
        else
        imp=baseImp*0,2
        end if
msgbox(imp)
End Sub
```

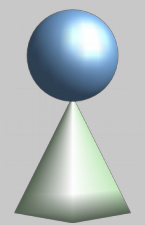
Plan

1) Variables
2) Entrées et sorties
3) Instruction conditionnelle
4) **Boucle**
5) Fonctions et procédures
6) Objets d’un tableur



Boucle

- Sur une machine, on peut vouloir répéter une opérations plusieurs fois.
- Ex : pour tous les fichiers d'un répertoire, on veut les copier.
- Ex : pour tous les utilisateurs d'une base de plus de 20 ans, on veut leur envoyer un mail
- Ex : tant qu'on a pas atteint une solution convenable pour une équation, on continue de chercher (solveur)
- Ex : on veut calculer tous les montant TTC d'une colonne Excel à partir des montant HT dans une autre colonne

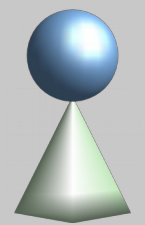


Boucle

Deux structures de boucle principales

```
nbV ← Lire("Donner le nombre des valeurs)
        pour i=1 à nbV
        val ← Lire("Donner une valeur")
        som ← val+som
        fin pour
afficher(som)
```

```
repUtilisateur ← "Oui"
        Tant que reponseUtilisateur="oui"
        repUtilisateur ← Lire("Voulez vous continuer ?")
        fin tant que
```



Boucle pour / for

On répète une séquence d'instructions un nombre donné de fois.

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age, i  : entier

DEBUT
nb ← 0
        pour i=1 à 10
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        fin pour
afficher(nb)
FIN
```

```
Algo calcul_Moyenne_notes

Déclaration des variables
note, S, moy : réel
nb,i  : entier

DEBUT
nb ← Lire("Combien de notes ?")
S ← 0
        pour i=1 à nb
        note ← Lire("Saisir une note")
        S ← note+S
        fin pour
moy ← S/nb
afficher(moy)
FIN
```

Boucle tant que / while

On répète une séquence d'instructions tant qu'une instruction conditionnelle est vraie.

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

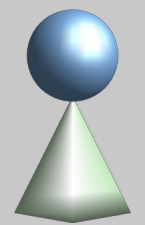
```
Algo calcul_Moyenne_notes

Déclaration des variables
note, S, moy : réel
nb  : entier
rep : chaîne de caractères
DEBUT
nb ← 0
S ← 0
rep ←"oui"
        Tant que rep="oui"
        note ← Lire("Saisir une note")
        S ← S+note
        nb ← nb+1
        rep ← Lire("Avez vous d'autres notes à saisir ?")
        fin tant que
moy ← S/nb
afficher(moy)
FIN
```

Boucle tant que / while

Des variations à commenter. Le code de gauche marche. Quid de celui de droite ?

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

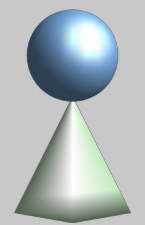


Boucle tant que / while

Des variations à commenter. Le code de gauche marche. Quid de celui de droite ?

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

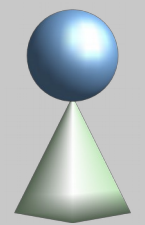


Boucle tant que / while

Des variations à commenter. Le code de gauche marche. Quid de celui de droite ?

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        fin tant que
afficher(nb)
FIN
```

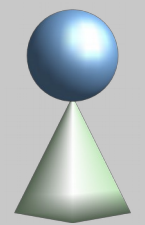


Boucle tant que / while

Des variations à commenter. Le code de gauche marche. Quid de celui de droite ?

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

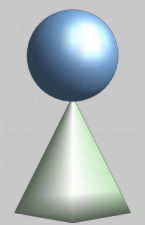
```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        afficher(nb)
        fin tant que
FIN
```



Les boucles en VBA - for

```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age, i  : entier

DEBUT
nb ← 0
        pour i=1 à 10
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        fin pour
afficher(nb)
FIN
```

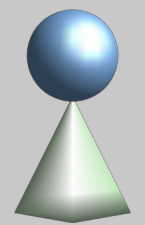
```
Sub compteIndividusMajeursSur100()


Dim nb as Integer, age as Integer, i as Integer


nb=0
        for i=1 to 10
        age=Inputbox("Donner age")
                if age>=18 then
                nb=nb+1
                end if
        next
Msgbox(nb)
FIN
```



Les boucles en VBA - while

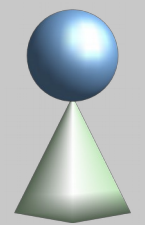
```
Algo compteIndividusMajeursSur100

Déclaration des variables
nb, age  : entier
rep : chaîne de caractères

DEBUT
nb ← 0
rep ← "oui"
        tant que rep="oui"
        age ← Lire("Donner age")
                si age>=18 alors
                nb ← nb+1
                fin si
        rep ← Lire("D'autres individus ?")
        fin tant que
afficher(nb)
FIN
```

```
Sub compteIndividusMajeursSur100()


Dim nb as integer, age as integer
Dim rep as String


nb=0
rep="oui"
        while rep="oui"
        age=Inputbox("Donner age")
                if age>=18 then
                nb=nb+1
                end if
        rep=Inputbox("D'autres individus ?")
        wend
Msgbox(nb)
End Sub
```



Plan

1) Variables
2) Entrées et sorties
3) Instruction conditionnelle
4) Boucle
5) **Fonctions et procédures**
6) Objets d’un tableur



De l’intérêt des fonctions

```
Algo calcul_Appreciation

Déclaration des variables
note : reel
bonus, app : châine de caractères

DEBUT
note ← Lire("Donner la note de l’étudiant")
bonus ← Lire("Bonnus de l’étudiant : ‘aucun’, ‘peu’, ‘fort’")
        si bonus="aucun" alors
                si note >=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        sinon si bonus="peu" alors
        note ← note+1
                si note>=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        sinon
        note <-note+2
                si note >=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        fin si
afficher(app)
FIN
```

De l’intérêt des fonctions

```
Algo calcul_Appreciation

Déclaration des variables
note : reel
bonus, app : châine de caractères

DEBUT
note ← Lire("Donner la note de l’étudiant")
bonus ← Lire("Bonnus de l’étudiant : ‘aucun’, ‘peu’, ‘fort’")
        si bonus="aucun" alors
                si note >=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        sinon si bonus="peu" alors
        note ← note+1
                si note>=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        sinon
        note <-note+2
                si note >=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
        fin si
afficher(app)
FIN
```

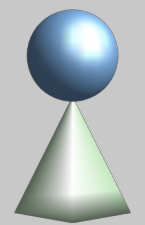
De l’intérêt des fonctions

```
Algo calcul_Appreciation

Déclaration des variables
note : reel
bonus, app : châine de caractères

DEBUT
note ← Lire("Donner la note de l’étudiant")
bonus ← Lire("Bonnus de l’étudiant : ‘aucun’, ‘peu’, ‘fort’")
        si bonus="aucun" alors
        calculAppreciation(note)
        sinon si bonus="peu" alors
        note ← note+1
        calculAppreciation(note)
        sinon
        note ←note+2
        calculAppreciation(note)
        fin si
afficher(app)
FIN
```

```
Fonction calculAppreciation(note : reel) : chaine de caractères
app : chaîne de caractères
                si note >=14 alors
                app ← "Bien"
                sinon si note >=10 alors
                app ← "AB"
                sinon
                app ← "Recalé"
                fin si
Renvoi app
Fin fonction
```

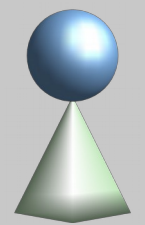
Modulariser le code

- On peut créer des fonctions
- On peut créer des procédures (chaque programme est une procédure / macro)
- On peut paramétrer les procédures et les fonctions



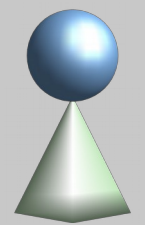
Modulariser le code

- Réutilisabilité
- Efficacité du code
- Lecture + aisée



Des fonctions pré-implémentées

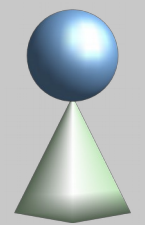
- Extraction de chaîne
- Aujourd’hui
- Max
- Somme
- Moyenne
- ...



Notions avancées

- Localité des variables.

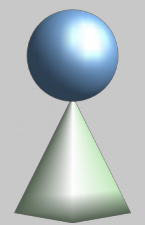
- ByRef / ByVal



VBA : construire des fonctions pour Excel

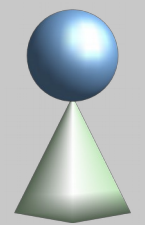
```
Function app(note)
app=""
                if note >=14 then
                app="Bien"
                elseif note >=10 then
                app="AB"
                else
                app="Recalé"
                end if
End function
```

- Note sur le retour des fonctions
- A placer dans un module
- Utilisation directe en Excel =app(17) dans une cellule.



Fonctions et procédures en VBA

- On peut aussi écrire des fonctions et des procédures en VBA et les utiliser dans le cadre du code VBA

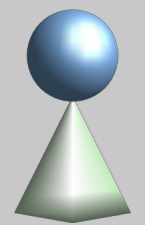


- Cf code avec plusieurs procédures plus haut.

- Fonctions et procédures -> TD.

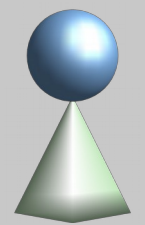
Plan

1) Variables
2) Entrées et sorties
3) Instruction conditionnelle
4) Boucle
5) Fonctions et procédures
6) **Objets d’un tableur**



Objets d’un tableurs

- On peut utiliser les objets d’un tableur dans les langages type VBA ou Libre Office Basic.

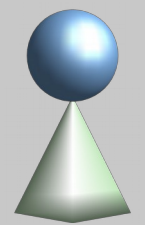


- Pas de syntaxe de référence en pseudo-algorithme.

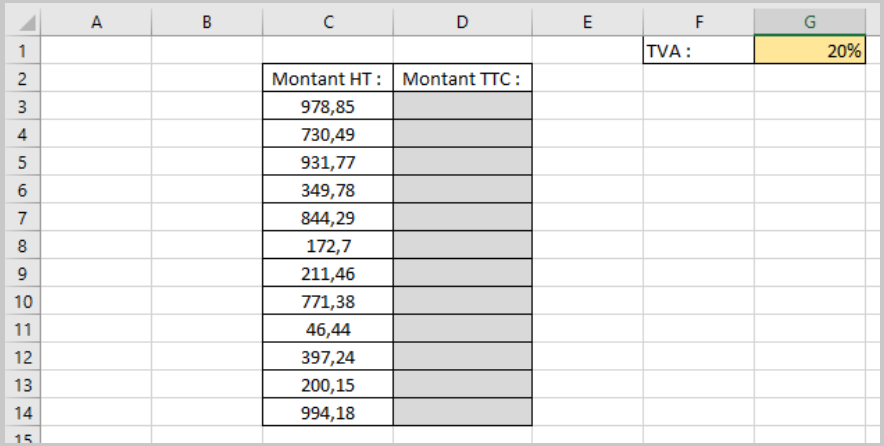
Accèder les cases / cellules d’une feuille

valCell(i,j) permet d’accéder la cellule de ligne i et de colonne j.

```
Algo case1
DEBUT
valCell(5,5) ← valCell(1,5)+valCell(2,5)
FIN

Algo case2
DEBUT
valCell(3,2) ← valCell(3,1)*1.2
FIN
```



Accèder les cases / cellules d’une feuille



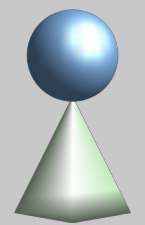
	A	B	C	D	E	F	G
1						TVA :	20%
2			Montant HT :	Montant TTC :			
3			978,85				
4			730,49				
5			931,77				
6			349,78				
7			844,29				
8			172,7				
9			211,46				
10			771,38				
11			46,44				
12			397,24				
13			200,15				
14			994,18				
15							

```
Algo case3
DECLARATION DES VARIABLES
i : entier

DEBUT
        pour i=3 à 14
        valCell(i,4) ← valCell(i,3)*valCell(1,7)
        fin pour
FIN
```

Travailler d’une feuille à l’autre

- Des syntaxes plus générales permettent de passer d’une feuille à l’autre

- Data.valCell(7,3) désigne la cellule C7 de la feuille Data.

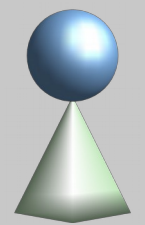


- On peut ainsi échanger des données entre les feuilles.

Travailler d’une feuille à l’autre

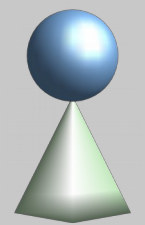
```
Algo recopie
DECLARATION DES VARIABLES
i,j : entier

DEBUT
        pour i=1 à 4
                pour j=1 to 100
                Data.valCell(j,i) ← Source.valCell(j,i)
                fin pour
        fin pour
FIN
```

```
Algo recopie2
DECLARATION DES VARIABLES
i : entier
x : reel


DEBUT
x ← 0
        pour i=1 à 100
        x ← x+Source.valCell(i,1)
        fin pour
Data.valCell(3,4) ← x/100
FIN
```



Syntaxes VBA

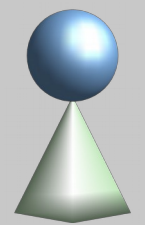
- Cells(i,j) désigne la cellule de ligne i et de colonne j.



- Worksheets("Data").Cells(7,5) : la cellule E7 de la feuille Data.

Au delà ...

- Pour bien programmer : Option Explicit.



- Aller vers d'autres langages : Python, Java...