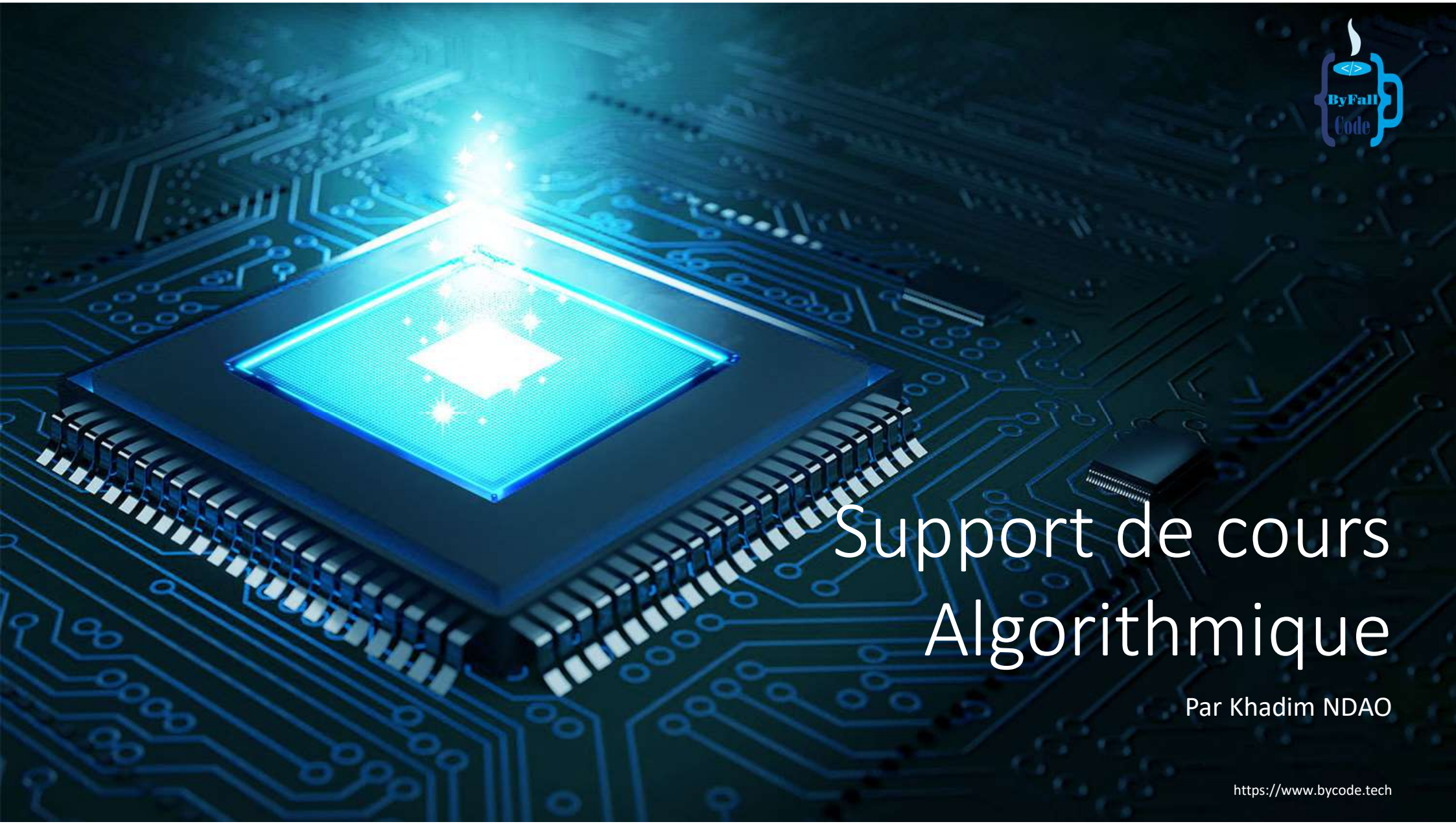
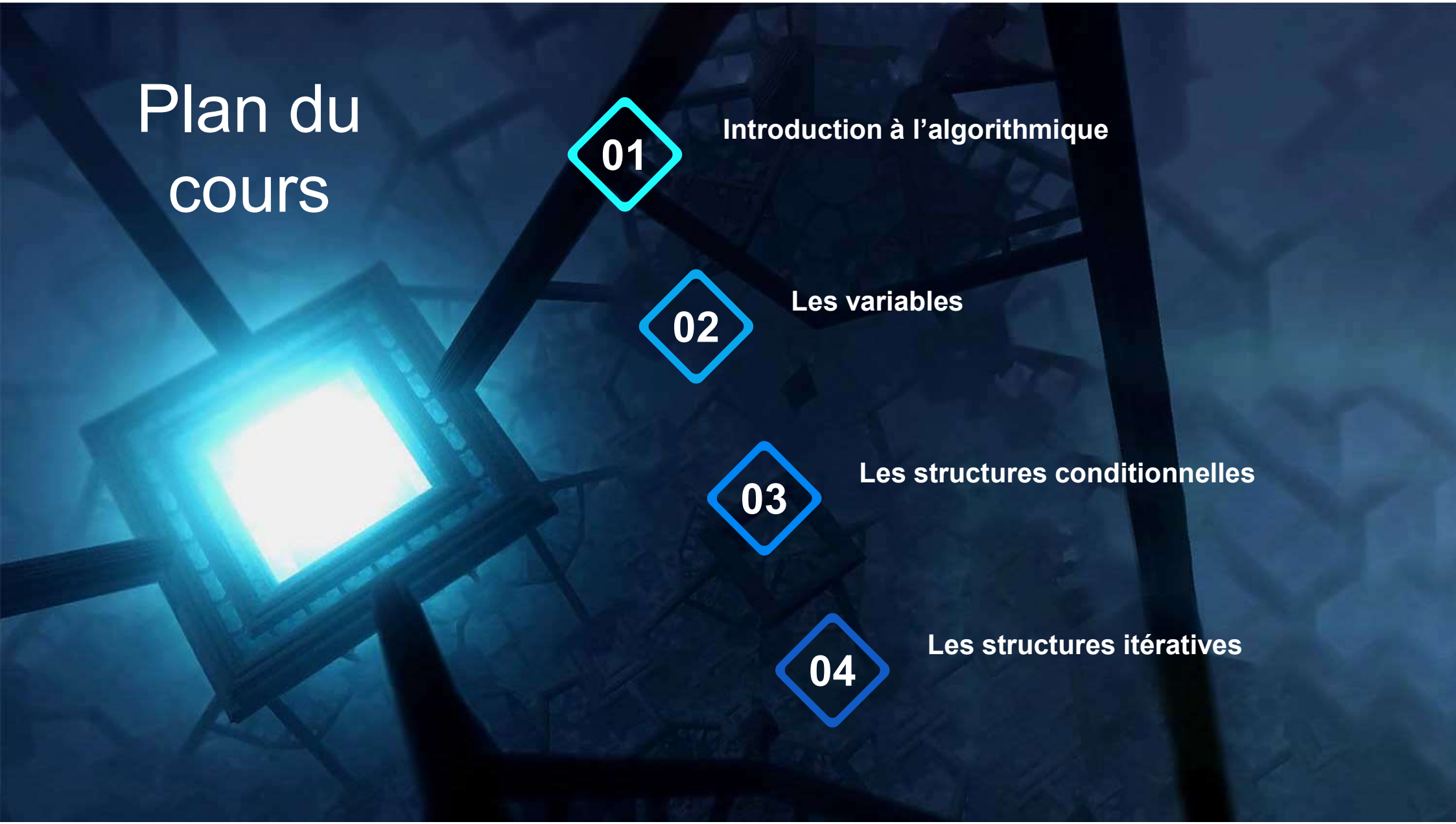


Support de cours Algorithmique

Par Khadim NDAO

https://www.bycode.tech

Algorithmique 1


Plan du cours
01
Introduction à l'algorithmique
02
Les variables
03
Les structures conditionnelles
04
Les structures itératives


Plan du cours
05
Les tableaux
06
Les types composés
07
Le type chaîne de caractère
08
Les sous-programmes

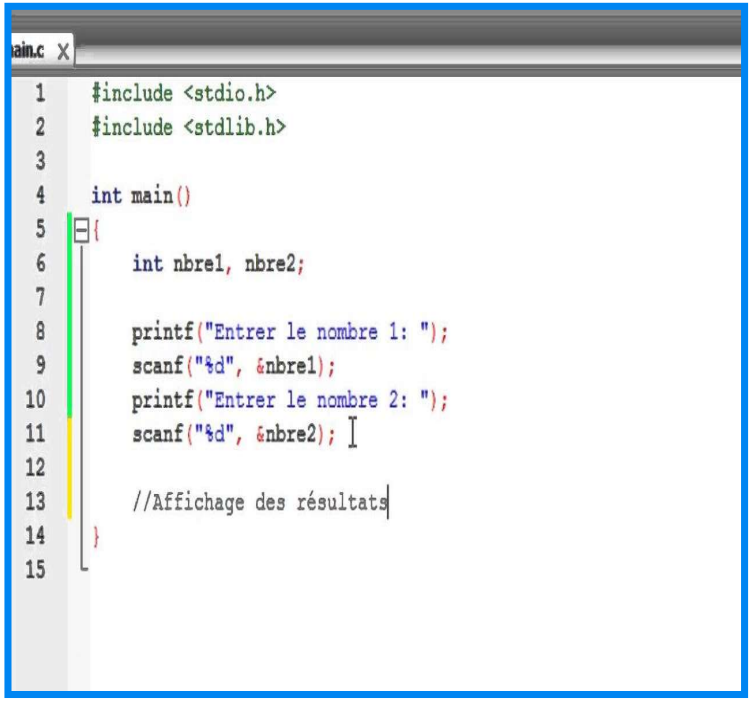
Introduction à l'algorithmique

Qu'est-ce que l'algorithmique?

L'algorithmique est le domaine qui étudie les algorithmes. Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème. Le mot algorithme vient du nom d'un mathématicien perse du IXe siècle, Al-Khwarizmi. **Wikipédia**

Pourquoi commencer par l'algorithmique?

Parce que l'algorithmique exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage. Apprendre l'algorithmique, c'est donc apprendre à manier la structure logique d'un programme informatique.



Exemple de code en langage C

Introduction à l'algorithmique

Quelle est la structure d'un algorithme?

Un algorithme est composé de 2 partie :

- **La partie entête**

C'est ici qu'on déclare le nom du programme et les variables qu'on va utiliser dans la 2e partie.

- **La partie corps**

C'est ici qu'on va mettre les blocs d'instructions du programme.

```
Programme addition
var nombre1, nombre2, somme : entier
Début
|           afficher("saisir le nombre 1")
|           saisir(nombre1)
|           afficher("saisir le nombre 2")
|           saisir(nombre2)
|           somme ← nombre1 + nombre2
|           afficher("la somme des 2 nombres est ", somme)
Fin
```

Programme *nom_du_programme*
Var1 : type
……
Var2, var3 :type
Debut
…….
…….
C'est ici qu'on ecrit les instructions
…….
…….
fin

Entete

corps

Les instructions de saisie et d'affichage

Un programme informatique est plus souvent destiné à avoir une interaction avec les utilisateurs. C'est pour cela qu'en algo on a les opérations de saisie et d'affichage

- Pour afficher un message à l'écran on utilise l'instruction : **afficher**() ou **ecrire**()
- Pour l'instruction de saisir on utilise **saisir**() ou **lire**()

Exemple :

Afficher(''bonjour tout le monde'')

Ecrire(''je suis content'')

Saisir(age)

Lire(taille)

Les variables

Qu’est-ce qu’une variable?

Une variable permet de stocker des informations au cours de notre programme. Une variable est définit par :

- Son nom

NB : *ce nom doit être unique dans le programme, ne doit pas comporter d’espace ni de caractères spéciaux, si c’est un mot composé on peut utiliser l’underscore « _ »*

- Son type
- Sa valeur (attribuée ou modifiée au cours du programme)

Exemple de déclaration

var age: entier

Var taille: reel

Var prenom: chaine

Les variables

Quels sont les types de variable ?

On peut trouver dans un programme des informations de plusieurs types par exemple du texte, des nombres, dates …

En algo on plusieurs types de valeurs que sont les types composés et les types simples ou primitifs ou encore type de base. Ce sont ces derniers qu'on va voir dans ce chapitre.

Les types de base en algo sont :

- **Le type entier** : tous les nombres sans chiffre après la virgule.
- **Le type reel** : tous les nombres avec des chiffres après la virgule.
- **Le type caractère** : tout les caractères alphabétiques, numériques et de punctuation.
- **Le type chaine de caractères** : toute représentation de texte.
- **Le type booléen** : constitué de 2 valeurs vrai et faux.

Les variables

Quels sont les types de variable ?

var nombre1, nombre2, somme : entier

Var nom, prenom : chaine

Var lettre : caractere

Var taille : reel

Exemple de déclaration de variable?

Les variables

Les opérations sur les variables

On peut faire plusieurs opérations sur une variable

1) Les opérations saisie et d'affichage

En algorithmique on peut afficher la valeur d'une variable ou donner à l'utilisateur la possibilité de saisir la valeur d'une variable.

Syntaxe :

- *Pour l'affichage*
  afficher(var)
  ecrire(var)
- *Pour saisir*
  saisir(var)
  lire(var)

Exemple

Afficher("saisir votre age")
Saisir(age)
Afficher("votre age est ",age)

Les variables

Les opérations sur les variables

2) l'affectation

C'est attribuer une valeur à une variable. En algo on utilise le symbole « ← » pour faire une affectation

Exemple :
Age ← 25
Cette ligne est lu : *âge prend la valeur de 25* et alors la valeur de âge est 25 à ce moment du programme.

NB : on peut faire plusieurs affectation sur une variable durant l'exécution du programme, mais la variable conserve toujours la dernière valeur qui lui est affectée

3) Les opérations arithmétiques

On peut des opérations arithmétiques comme l'addition, la soustraction, la division et la multiplication avec les variables de types numériques (entiers, réels). Deux opérations sont spécifiques aux entiers

- DIV la division entière
- MOD le reste de la division entière

Les opérations sur les variables

Exemple
A ←12
B←8
Somme ← A+B //somme est égale à 20

3) Les opérateurs de comparaison

Les opérateurs de comparaison qu'ont peut utiliser sur les variables sont :

- < strictement inférieur à
- > strictement supérieur à
- = égal à
- ≤ inférieur ou égal à
- ≥ supérieur ou égal à
- ≠ différent de

Exemple
a←12
B←13
A < B // la comparaison donne **vrai**
A > B // la comparaison donne **faux**

Excercices sur les variables

Exercice 1 :
Ecrire un programme Age qui

1. demande son âge à l'utilisateur ;
2. affiche l'âge que l'utilisateur à saisi.

Exercice 3 :
Ecrire un programme Age qui

1. demande son âge à l'utilisateur ;
2. lit la réponse de l'utilisateur et l'enregistre dans une variable âge de type entier ;
3. calcule l'année de naissance (à un an près) de l'utilisateur et l'enregistre dans la variable **annee** de type entier ;
4. affiche l'année de naissance ainsi calculée.

Excercice 5 :
Ecrire un programme qui permet de saisir 2 nombres a et b.
1) Afficher la valeur a et b
2) Echanger les valeurs de a et b
3) Afficher les nouvelles valeurs de a et b

Exercice 2 :
Ecrire un programme qui demande à l'utilisateur de saisir 2 valeurs val1 et val2. Le programme calcul et affiche

1. La somme des 2 valeurs
2. La différence
3. Le produit
4. La division
5. La division entière
6. Le reste de la division

Exercice 4 :

Ecrire un programme qui demande à l'utilisateur de saisir ses informations (nom, prénom, genre, âge, taille, poids) puis afficher toutes les informations de l'utilisateur.

Exercice 6 :
Ecrire un programme qui permet de résoudre l'équation **ax+b= 0**

Les structures conditionnelles

1) Définition

Une structure conditionnelle détermine si le bloc d'instruction suivant est à exécuter ou non. La condition est une expression booléen dont la valeur détermine le bloc d'instructions à exécuter.

Pour les conditions on utilise les opérateurs de comparaison qu'on avez vue sur avec les variables.

- $<$ strictement inférieur à
- $>$ strictement supérieur à
- $=$ égal à
- $\leq$ inférieur ou égal à
- $\geq$ supérieur ou égal à
- $\neq$ différent de

L'expression de la condition retourne une valeur booléen (vrai ou faux).

Les structures conditionnelles

2) Syntaxe

Il existe 2 type de conditions : le **si** et le **si … sinon**

Condition simple :

Si (condition[s]) Alors

Instructions

Finsi

Condition complexe :

Si (condition[s]) Alors

Instructions

Sinon

Instructions

Finsi

Exemple 1:

Lire(age)
Si (age>=18) Alors
| afficher(“vous etes majeur”)
Finsi

Exemple 2:

Lire(age)
Si (age>=18) Alors
| afficher(“vous etes majeur”)
Sinon
| afficher(“vous etes mineur”)
Finsi

Les structures conditionnelles

3) Conditions composées

Certains problèmes exigent parfois de formuler des conditions qui ne peuvent pas être exprimées sous la forme simple exposée ci-dessus. Pour lier des conditions on utilise des opérateurs logiques.

Nous disposons de 3 opérateurs logiques.

- Et
- Ou
- Non

Voir le tableau récapitulatif des combinaisons ci-contre

C1 et C2	C2 vrai	C2 faux
C1 vrai	vrai	Faux
C1 faux	faux	faux

C1 ou C2	C2 vrai	C2 faux
C1 vrai	vrai	Vrai
C1 faux	vrai	faux

Non C	
C vrai	Faux
C faux	Vrai

Les structures conditionnelles

4) La structure à choix multiple

La structure à choix multiple permet de proposé une liste d'option et de laisser le soin à l'utilisateur de faire son choix. Les instructions à exécuter dépendent du choix effectuer par l'utilisateur.

2) Syntaxe

```
Suivant choix faire
|  option A : < InstructionA1 >
|             < InstructionA2 >
|  option B : < InstructionB1 >
|             < InstructionB2 >
|  option C : < InstructionC1 >
|             < InstructionC2 >
Sinon
|             < Instruction1 >
|             < InstructionN >
FinSuivant
```

Exemple :

Programme Restaurant

```
Var choix;
Debut
    Afficher("A - Thiebou dieune")
    Afficher("B - Thiebou guinaar")
    Afficher("C - Thiebou yapp")
    Afficher("D - Yassa")
    Afficher("Faites votre choix")
    Lire(choix)
    Suivant choix faire
    |  option A : afficher("ca coute 500 frcs")
    |  option B : afficher("ca coute 7800 frcs")
    |  option C : afficher("ca coute 5800 frcs")
    |  option D : afficher("ca coute 9000 frcs")
    Sinon
    |     afficher("Plat non disponible")
    FinSuivant
Fin
```

Les structures conditionnelles

5) Imbrication de structures conditionnelles

C'est le fait de définir une structure conditionnelle à l'intérieur d'un autre structure conditionnelle. On peut aussi le nommé par les branchements

2) Syntaxe

```
Si (condition[s]) Alors
        Instructions
Sinon
      Si (condition[s]) Alors
            Instructions
      FinSi
Finsi
```

Exemple :

Programme Note

```
Var note;
Debut
      Afficher("saisir la note")
      Lire(note)
      Si (note <= 8) Alors
            Afficher("Insuffisant")
           Sinon
                Si (note <= 12) Alors
                      Afficher("Moyen")
                    Sinon
                          Si (note <= 12) Alors
                                Afficher("Bien")
                                Sinon
                                      Afficher("Très Bien")
                          FinSi
                FinSi
      Finsi
Fin
```

Excercices sur les conditions

Exercice 1 : écrire un programme qui demande l'Age d'un utilisateur. Le programme détermine si ce dernier est mineur ou majeur.

Exercice 2 : écrire un programme qui demande à un utilisateur de saisir le une heure (Heure, minute, seconde). Le programme vérifie et affiche l'heure sous le format **HH :MM : SS** si cette dernière est valide. Sinon, il dit à l'utilisateur que l'heure n'est pas valide.

Exercice 3 : Ecrire un programme qui permet de réaliser le menu d'orange money : #144#

Exercice 4 : Ecrire un programme qui permet de tester si un nombre est positif, négatif ou nul

Exercice 5 : Ecrire un programme qui permet donner 3 nombre puis de déterminer le plus grand, le plus petit et la moyenne de ces 3 nombres.

Exercice 6 : Ecrire un programme qui permet de tester si un nombre est paire ou impaire

Exercice 7 : Ecrire un programme qui permet de résoudre l'équation : $Ax^2 + Bx + C = 0$

Exercice 8 : Ecrire un programme qui permet de créer une calculatrice. D'abord le programme affiche les différentes opérations à l'utilisateur et lui demande de choisir. Ensuite, l'utilisateur saisir les 2 valeurs. Enfin le programme affiche le résultat.

NB: il faut prendre en contre que la division par 0 est impossible ici.

Voici le menu que l'utilisateur va voir :

1. Addition
2. Soustraction
3. Multiplication
4. Division entière
5. Division réelle
6. Reste de la division
7. Quitter

Faites votre choix

Les structures itératives

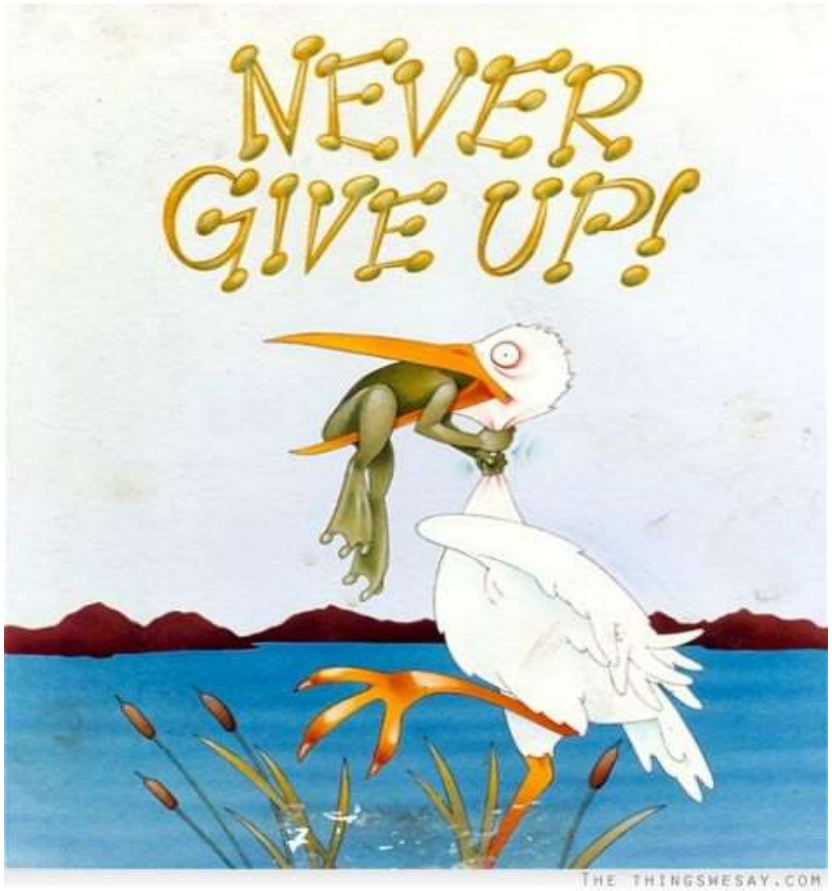
1) Définition

Les structures itératives ou encore appelé instructions de répétitions ou encore boucles permettent d'exécuter plusieurs fois un même bloc d'instructions dans un programme.

Ici aussi on utilise des opérations de comparaison et la répétition s'effectue tant que la valeur booléenne de la condition est égale à vrai.

On dispose de trois type boucles en algorithmique

- Tanque … faire
- Répéter … jusqu'à
- Pour … faire



Les structures itératives (boucles)

2) La boucle *tantque … faire*

La boucle tanque permet d'effectuer plusieurs fois le même traitement. Elle répéte les instructions tanque la condition n'est pas à **faux**

2) Syntaxe

TantQue (condition[s])

… Instructions …

FinTantQue

Exemple :

Programme nombre_positif

```
Var nombre;
Debut
        Afficher("Saisir un nombre")
        Lire(nombre)
        Tanque (nombre <= 0)faire
        |   Lire(nombre)
        Fin Tanque
        Afficher("vous avez saisi ", nombre)
Fin
```

2) La boucle *repeter … jusqu'a*

La boucle repeter … jusqu'à permet de répeter une bloc d'instruction jusqu'à ce qu'une condition soit remplie.

On l'utilise en générale pour faire des contrôles de saisie de l'utilisateurs.

2) Syntaxe

```
Repeter
|  < Instruction1 >
|        …….
|  < InstructionN >
Jusqu'à (condition)
```

Exemple :

Programme nombre

```
Var nombre;
Debut
      Repeter
      |      Afficher("Saisir un nombre positif")
      |      Lire(nombre)
      Jusqu'à (nombre > 0)
      Afficher("vous avez saisi ", nombre)
Fin
```

Les structures itératives (boucles)

4) La boucle *pour*

La boucle pour est très utiliser en programmation pour réexécuter une bloc d'instruction un nombre de fois connu. Son écriture est pratique car elle permet de désigner l'ensemble de la boucle en une seule ligne : l'incrémentation de la boucle (par 1) est sous-entendue à la fin de la boucle.

NB : si on veux incrémenter par 2 on le précise avant la fin de la boucle.

2) Syntaxe

```
Pour var allant de debut à fin faire
|  < Instruction1 >
|        .....
|  < InstructionN >
FinPour
```

Exemple :

Ecrire un programme qui permet de compter jusqu'à 10

Programme compteur

```
Var cpt;
Debut
    Pour cpt allant de 1 à 10 faire
    |  affciher(cpt)
    FinPour
Fin
```

5) L’imbrication de boucle

Dans un programme il est fréquent qu’une boucle soit définit à l’intérieur d’une autre boucle. Dans ce cas on parle d’imbrication de boucle.
Pour chaque occurrence de la boucle externe, il y’aura autant de traitement pour la boucle interne

Exercice d’application :
Écrire un programme qui permet de saisir les notes des étudiants d’une classe. Puis le programme détermine et affiche la moyenne la classe, la plus petite moyenne et la plus grande moyenne de la classe
Une note doit être comprise en 0 et 20

5) L'imbrication de boucle

Dans un programme il est fréquent qu'une boucle soit définit à l'intérieur d'une autre boucle. Dans ce cas on parle d'imbrication de boucle.
Pour chaque occurrence de la boucle externe, il y'aura autant de traitement pour la boucle interne

Exercice d'application :
Écrire un programme qui permet de saisir les notes des étudiants d'une classe. Puis le programme détermine et affiche la moyenne la classe, la plus petite moyenne et la plus grande moyenne de la classe
Une note doit être comprise en 0 et 20, le nombre d'étudiant doit être saisie par l'utilisateur et doit être supérieur à 2.

Les tableaux

Introduction

Les tableaux sont des structures **linéaires homogènes** composées de cellules dont chacune peut prendre une valeur.

Tout tableau est caractérisé par :

- ✓ Son ***nom*** : il représente la variable à utiliser pour manipuler le tableau dans le programme
- ✓ Son ***type*** de valeur : il détermine le type de valeur que va prendre les cellules du tableau
- ✓ Sa ***dimension*** : il détermine la nature du tableau, c'est-à-dire si la dimension est égale à 1 alors le tableau est appelé **vecteur**, si elle est égale à 2 alors on parle de **matrice** et si la dimension est égale à 3 on parle **d'espace**. Dans ce chapitre on va se limiter sur les vecteurs et les matrices.
- ✓ Sa ***taille*** : elle représente le nombre de cellule qui compose le tableau. Pour un vecteur le nombre de colonnes est égal au nombre de cellules et pour une matrice c'est égale à **nombre de ligne x nombre de colonne**.

Les tableaux

I) Les vecteurs

1) Définition

Un vecteur est un tableau a une dimension c'est-à-dire une tableau à une ligne avec plusieurs cellules (colonnes).

Chaque cellule peut contenir une valeur. En algo les cellules sont numérotées à partir de 1

2) Déclaration

Syntaxe 1 :
Const N = nombreDeColonnes
Type tab = tableau[1 – N] type de valeur
Var t : tab
Syntaxe 2 :
Const N = nombreDeColonnes
Var t : tableau[1 – N] type de valeur

Syntaxe 1 :
Const N = 175
Type tab = tableau[1 – N] entier
Var t : tab
Syntaxe 2 :
Const N = 175
Var t : tableau[1 – N] entier

Les tableaux

I) Les vecteurs

3) Manipulations des cellules d'un vecteur

Soit le vecteur d'entier ci contre, T une variable

De type tableau et i l'indice de parcours des

Cellules du tableau on a :

Cellule no 1	Cellule no 2	Cellule no 3	Cellule no 4	Cellule no 5	Cellule no 6
10	32	43	9	45	34

Si i = 1 ➔ T[i] = 10

Si i = 2 ➔ T[i] = 32

…….

Si i = 6 ➔ T[i] = 34

NB : La notation T[i] représente la valeur contenue à la cellule i du tableau

Les tableaux

I) Les vecteurs

4) Création d'un vecteur

Pour créer un vecteur il faut d'abord

- ✓ déclarer la variable qui permettra de manipuler le vecteur,
- ✓ ensuite une variable d'indice pour parcourir le tableau (indice de parcours),
- ✓ enfin remplir toutes les cellules du vecteur en utilisant l'indice de parcours pour affecter une valeur à chacune d'elles.

Les tableaux

Exercice d’application

Ecrire un programme qui permet de remplir un vecteur d’entier de 150 cellules. Le programme affiche le contenue du vecteur, calcules et affiche la moyenne des nombres paires et le pourcentage de présence de nombre positif dans le vecteur.

Moyenne nombre paire = somme(nombre pair) / nombre de nombre paires

Pourcentage nombre positif = nombre de nombre positif * 100 / 150

On déclarera N comme constante qui représente le nombre de cellules du vecteur.

I) Les vecteurs

5) Les operations sur les vecteurs

Il est possible de faire des traitements sur les valeurs d'un vecteur comme les opération scalaires, la recherche, la modification, le décalage cyclique, le tri, l'union, la fusion, l'intersection …

Exercice d'application :

Ecrire un programme qui permet de remplir et d'afficher un tableau de 75 nombres entiers. Le programme affiche l'inverse ainsi que les nombres premiers du vecteurs.

Les tableaux

I) Les matrices

1) Définition

Une matrice est un tableau à 2 dimensions c'est-à-dire un tableau qui a plusieurs ligne et chacune de ces lignes est composée des colonnes et se comporte comme un vecteur (tableau à une dimension).
Si dans une matrice le nombre de ligne est égale au nombre de colonne on parle de matrice carré sinon on parle de matrice non carré.

2) Déclaration

Syntaxe 1 :

Const N = lignes

M = colonnes

Type nomMatrice = matrice[1 - - N, 1 - - M] type de valeur

Var m : nomMatrice

Syntaxe 2 :

Const N = lignes

M = colonnes

Var nomVarMatrice = matrice[1 - - N, 1 - -M] type de valeur

Les tableaux

I) Les matrices

3) Manipulation d'une matrice

Soit

- **Mat** : la variable matrice
- **i** indice de parcours des lignes
- **j** indice de parcours des colonnes

La notation **Mat[i][j]** signifie la valeur qui est dans **l'intersection** de la **ligne i** et la **colonne j** autrement dit la valeur qui se trouve à la ligne i et à la colonne j de cette ligne.

Colonne / ligne i=j=1	Colonne 2 J=2	Colonne 3 J=3	Colonne 4 J=4
Ligne 2 i=2	32	67	90
ligne 3 i=3	23	76	8
ligne 4 i=4	21	6	9

Les tableaux

I) Les matrices

4) Création d'une matrice

Pour créer une matrices il faut remplir ces cellules une à une pour ce faire il faut déclarer 2 variables d'indice, l'un pour les lignes et l'autre pour les colonnes. Puisque chaque ligne est composée de plusieurs colonnes on imbrique 2 boucles pour le remplissage.

Exercice d'application

Ecrire un programme qui permet de remplir une matrice d'entier de 9 ligne et 11 colonne, afficher le contenu due la matrice et puis de déterminer la moyenne des nombre pair de la matrice.

I) Les matrices

5) Calculs applicables aux matrices

On peut appliquer plusieurs opérations : la recherche, la somme, produit, opération de diagonalisation …

Exercice d'application

Ecrire un programme qui permet de remplir 2 matrices carrés d'ordre 3, de les afficher, de déterminer et d'afficher la matrice somme de ces 2 matrices.

Les types composés

Introduction

Comme on l'avait vue sur le chapitre ou on traiter les variables, en algo on a les types simples ou élémentaires (entier, chaine, caractere, reel, booléen) et les types composés. Ces derniers sont créés sur la base des types élémentaires. On distingue de l'énumération, l'intervalle et les structures ou enregistrements.

Les types composés

I) L'énumération

1) Définition

L'énumération est un type de données homogènes dans lequel il faut spécifier tous ces composants. En général les valeurs possible d'une énumération sont limitées.

2) Syntaxe

Syntaxe 1 :

Type nomEnum = valeur1, valeur2, …., valeurN

Var enum : nomEnum

Syntaxe 2:

var variableEnum = valeur1, valeur2, …., valeurN

Exemples :

Ex1 :
Var sexe : 'm', 'f', 'F', 'M'
Ex 2 :
Type couleur_primaire = 'rouge', 'vert', 'bleu'
Var couleur : couleur_primaire
Ex 3
Type voyelle = 'e', 'i', 'o', 'u', 'y', 'a'
Var voy : voyelle

Les types composés

II) L'intervalle

1) Définition

C'est une type de données homogènes dans lequel il faut spécifier la valeur initiale et la valeur finale. Ces 2 valeurs sont séparées par **..**

NB : les types de valeurs qui peuvent être associé à l'intervalle sont : l'entier et le caractére.

2) Syntaxe

Syntaxe 1 :

Type nomInterval = valInitiale **..** valeurfinale

Var nomvariable : nomInterval

Syntaxe 2:

var variableInterval : valInitiale **..** valeurfinale

Exemples :

Ex1 :

Var chiffre : 0 **..** 9

Ex 2 :

Type Alphabet = 'A' **..** 'Z'

Var lettre : Alphabet

Les types composés

III) Les structures

1) Définition

Les structures ou enregistrements représentent des données composées dans laquelle on peut disposer au moins de 2 champs qui peuvent être de type simple ou composé (énumération, intervalle, structures)

2) Syntaxe

Syntaxe :

```
nomStructure = structure
Debut
    champ1: type1
    champ2 : type2
      …..
    champN
Fin
Var varStructure : nomStructure
```

Exemple : Définir la structure l'enregistrement Personne définie comme suit (nom, prenom, age, adresse, sexe)

```
Personne = structure
Debut
        Nom, prenom, adresse : chaine
        Age : entier
        Sexe : 'm', 'f', 'M', 'F'
Fin
Var p : personne
```

III) Les structures

3) Manipulation des champs

Soit *A une variable de type structure et b un champ de cette structure*, la notation **A.b** permet d'accéder à la valeur du **champ b** de **A.**

Par exemple considérons l'enregistrement qu'on en pris en exemple précédemment on a :

- ✓ **P.nom** permet d'accéder au champ **nom**
- ✓ **P.prenom** permet d'accéder au champ **prenom**
- ✓ **P.adresse** permet d'accéder au champ **adresse**
- ✓ **P.age** permet d'accéder au champ **age**
- ✓ **P.sexe** permet d'accéder au champ **sexe**

```
Personne = structure
Debut
        Nom, prenom, adresse : chaine
        Age : entier
        Sexe : 'm', 'f', 'M', 'F'
Fin
Var p : personne
```

III) Les structures

Exercice d'application

Ecrire un programme qui permet de saisir une série de N étudiant. Le programme affiche les données de chaque étudiant ainsi que l'étudiant qui a eu la plus petite moyenne et celle qui la plus grande moyenne.

Un étudiant est caractérisé par son matricule, son nom, son prénom, sa classe et sa moyenne.

Les types composés

III) Les structures

4) Les tableaux de structures

C'est un tableau dont les cellules contiennent des enregistrements.

a) Syntaxe

Syntaxe :

```
Const colonne = valeur
nomStructure = structure
Debut
    champ1:type1
    champ2:type2
      …..
    champN: typeN
Fin
Type nomTabStruct = tableau[1 - - colonne] nomStructure
Var nomVarTabStruct : nomTabstruct
```

Exemple :

```
Const colonne = 100
Personne = structure
Debut
    nom, prenom, adresse : chaine
    age : entier
    Sexe : 'm', 'f', 'M', 'F'
Fin
Type tabEtudiant = tableau[1 - - colonne] Personne
Var tabE : tabEtudiant
```

III) Les structures

4) Les tableaux de structures

b) Manipulation

Soit

- **T** la variable de type tab structure
- **I** l'indice de parcours
- **X** un champ de T

La notation T[i].X permet d'accéder à la **valeur du champ X** de la structure situé à la **cellule i du tableau T**

III) Les structures

4) Les tableaux de structures

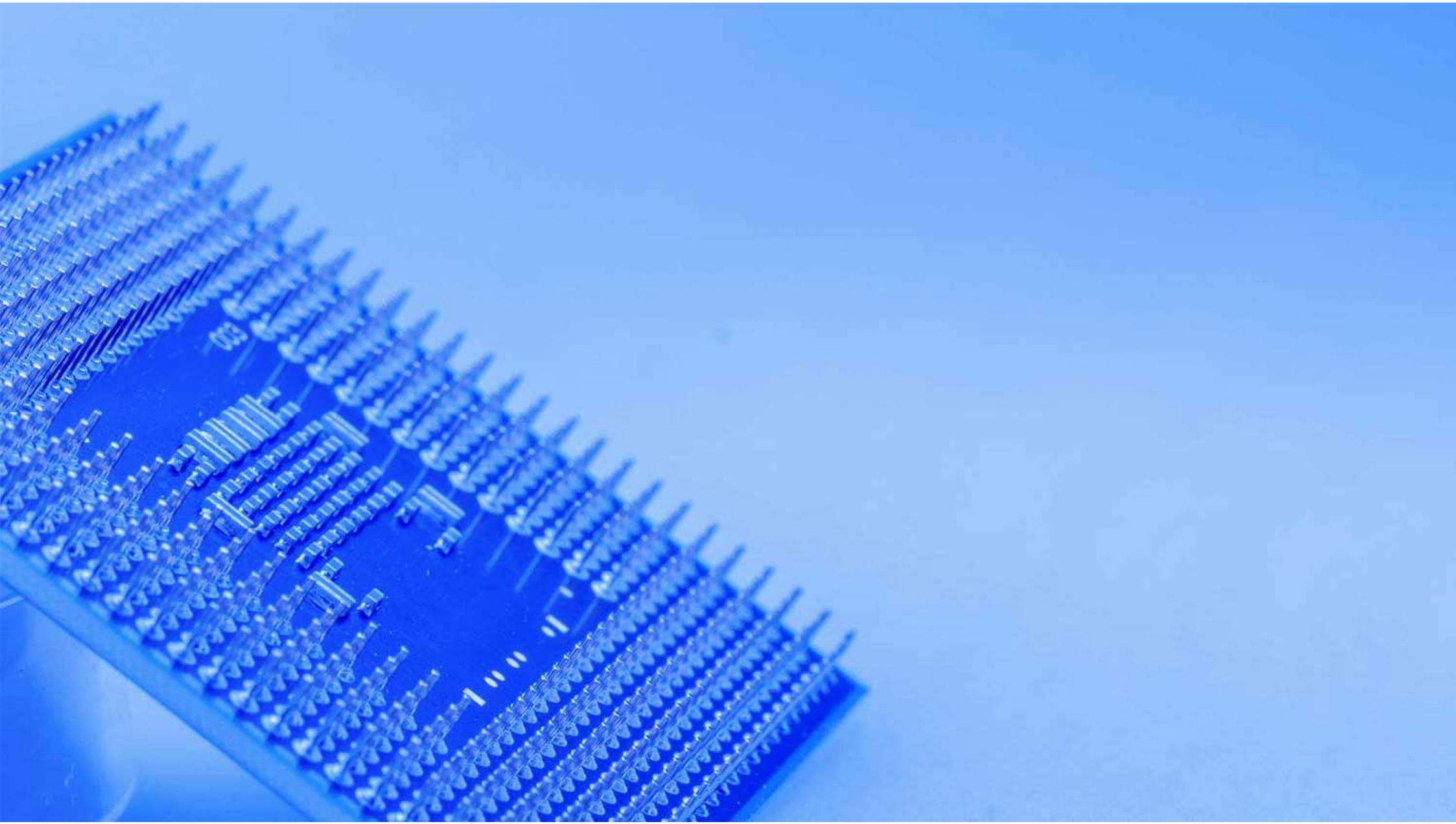
c) Création

Pour créer un tableau de structures il faut remplir ces cellules et sous cellules. Pour traiter toutes les valeurs du tableau il faut un indice de parcours i afin de donner à chaque sous cellule une valeur

Exercice d'application

Ecrire un programme qui permet de remplir un tableau de 750 produit, d'afficher les produits du tableau, de déterminer et d'afficher le nombre de produit dont la catégorie est « lait » ainsi que le montant total des produits de son stock.

Un produit est caractérisé par son code, son nom, son prix unitaire, sa quantité et sa catégorie




Merci!
Pour toutes suggestions, veuillez me contactez par contact@bycode.tech