

Algorithme et turbo pascal

Ce sera plutôt un cours sur la programmation pascal car c'est un langage qui est très proche de l'algorithme

La structure d'un **programme en pascal** est la suivante :

```
Program Nomduprog;
var
  i: integer;
  a: char;
  b: string;
begin
  i:=1;
  a := 'c' ;
  b := 'bonjour oumar' ;
end.
```

Les operateurs du langage pascal

L'opérateur d'affectation (:=) : il permet de donner à une variable déclarée une valeur, ainsi si la variable contenait une autre valeur il la supprime et lui donne la nouvelle valeur.

Ainsi si A est une variable integer

A :=5 fixe la valeur 5 à A.

L'opérateur de comparaison(=) : il permet de faire une comparaison entre 2 valeurs. Ainsi si la variable A contient un entier A=5 va vérifier si la valeur contenue dans A égale 5

La différence (< >) : il permet de faire une comparaison en vérifiant que les 2 valeurs sont différentes

Ainsi $A < 2$ vérifiera que la valeur de A est différente de 2

Infériorité(<) permet de vérifier qu'une valeur est plus petite que l'autre

Ainsi $A < 2$ vérifiera que la valeur de A est plus petite que 2

Supériorité(>) permet de vérifier qu'une valeur est plus grande que l'autre

Ainsi $A > 2$ vérifiera que la valeur contenue dans A est plus grande que 2

On a les operateurs + , - , * , / que vous connaissez déjà

Pour la division(/) on a en plus les operateurs **div** et **mod**

Div permet de motrer que la partie entiere de la division

Par exemple

$5 / 2 = 2,5$ mais

$5 \text{ div } 2 = 2$

Mod quant à lui il permet de montrer que le reste de la division

Ainsi

$5 / 2 = 2,5$

$5 \text{ div } 2 = 2$

$5 \text{ mod } 2 = 1$

Les types

Une variable utilisée dans un programme doit être déclarée à l'avance pour connaître son type. Les types les plus utilisés sont les suivants :

Integer : ce sont les nombres entiers sans virgules

Real : il regroupe les entiers avec ou sans virgule

Char : peut contenir un caractère alphabétique ou numérique

String : peut contenir une chaîne de caractères. Exemple une phrase

Boolean : il prend 2 valeurs seulement (vrai ou faux) et (0, ou 1)

Byte : prend des entiers compris entre 0 et 255

Les entrées-sorties conversationnelles

Ce sont les commandes qui nous permettent de communiquer avec le programme à travers l'ordinateur

Write permet d'afficher les sorties à l'écran tout comme **writeln** qui lui après avoir affiché les sorties place le curseur à la ligne suivante

Read permet de lire les valeurs saisies au clavier. **Readln** lit et se place à la ligne suivante

Les tableaux

Un tableau est un objet qui peut contenir des éléments de même type

Dans un tableau tous les éléments sont numérotés de 1 à N

Ainsi on peut avoir un tableau d'entiers, un tableau de caractères,.....

Voici la syntaxe de déclaration d'un tableau :

```
var montableau=array[1..15] of integer ;
```

Graphiquement le tableau ressemblera à la représentation suivante

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	100	43	6	7	...									

Dans ce tableau donné en exemple on a pour

i=1 alors montableau[i]=7

i=3 alors montableau[i]=43

ect....

Exemple de programme utilisant un tableau

```
Program tableau;
Var montableau : Array[1..10] Of Integer ;
i : Integer ;
  BEGIN
  For i:=1 To 10 Do
  begin
```

```

    write('entrez la valeur de l'element à l'emplacement ' i );
    readln(montableau[i]);
    end;
writeln('tapez une touche pour afficher le contenu du tableau') ;
readln ;
    writeln('liste des entiers dans le tableau');
    For i:=1 To 10 Do
    begin
    write(montableau[i]);
    end;
    writeln('tapez une touche pour sortir');
    readln ;
END.

```

Ce programme appelle l'utilisateur à saisir les 10 valeurs du tableau et affiche les 10 entiers qu'il a saisi dans le tableau

On aurait pu afficher un élément quelconque du tableau en appelant son index (numéro)

Exemple

```
Writeln(montableau[7]);
```

Affiche l'élément du tableau qui se trouve à l'emplacement numéro 7.

Les Matrices

Une matrice est un tableau à 2 dimensions, c'est-à-dire 2 tableaux dont un en longueur et l'autre en largeur

Syntaxe de déclaration d'une matrice

```
Var matrice : array[1..7,1..5] of integer ;
```

On va vous montrer comment ce tableau de 7 lignes et 5 colonnes sera représenté graphiquement

Les indexes	1	2	3	4	5	6	7
1							
2							
3		56			22	...	
4			45				
5							

Ce tableau pourra contenir $7 \times 5 = 35$ entiers

Contrairement à un tableau donc un élément d'une matrice aura deux index qui sont le numéro de la ligne et le numéro de la colonne à laquelle il se trouve

ainsi dans notre matrice :

matrice[3][2] correspond à la valeur 56

matrice[4][3] correspond à la valeur 45

ect.....

Maintenant que vous comprenez ce que c'est qu'une matrice, on va passer à la programmation de la matrice

```

Program lamatrice ;
Var matrice: array [1..7, 1..5]of integer ;
  i, j: integer;
begin
  writeln('Remplissage de la matrice');
  for i=1 to 7 do
  for j=1 to 5 do
  begin
  writeln('Entrez la valeur à l''emplacement ['i,']','[j,]');
  readln(matrice[i][j]);
end ;
writeln('fin de la saisie des valeurs de la matrice') ;
writeln('tapez sur entrez pour afficher la matrice') ;
readln ;
  writeln('Affichage de la matrice') ;
  for i :=1 to 7 do
  begin
    for j :=1 to 5 do
      write(matrice[i][j], ' ');
      writeln;
    end;
  readln;
end.

```

Ce programme appelle l'utilisateur à saisir un par un toutes les valeurs de la matrice ensuite il affiche toutes les valeurs que vous avez saisi dans la matrice

Les instructions de contrôle

Les instructions de contrôle permettent à un programme de faire les traitements nécessaires
Ainsi on a :

L'instruction **if** (*if=si* et *else=sinon* en français)

Il permet de vérifier si l'expression qui la suit est juste et de faire les instructions demandées
Syntaxe de l'instruction if

If (expression)

Begin

//mettre vos instruction

End

Else

Begin

//les instructions au cas où le if serait faux

End ;

La boucle **for** (*for=pour* et *do=faire* en français)

La boucle **for** permet de parcourir d'une valeur de départ à une valeur de fin de boucle et de répéter à chaque fois l'instruction qui suit le **for**
Exemple d'utilisation de la boucle **for**

```
For i :=1 to 15 do
Write( i, ' , ' );
```

Cette boucle permet de parcourir les valeurs comprises entre 1 et 15 et d'afficher à chaque fois la valeur de *i*
Il affichera donc 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

La boucle **repeat** (*repeat=repeter* et *until=jusqu'à*)

Cette boucle permet de repeter une instruction jusqu'à ce que la condition qui se trouve dans le *until* soit atteinte

Exemple

Cet exemple fera le même traitement qu'on vient de faire avec la boucle **for** et affichera donc les valeurs 1, 2,, 15

```
i:=0 ;
Repeat
  i :=i+1 ;
  Write(i, ' , ');
Until(i=15);
Readln;
```

La boucle **while** (*while=tant que* en français)

La boucle **while** permet de faire un traitement tant que l'expression qu'elle porte n'est pas satisfaite
Syntaxe de **while**

```
While(expression) do
begin
//instructions
End;
```

On prend toujours le même exemple pour afficher les valeurs 1, 2,, 15 avec la boucle **while**

```
i:=0 ;
While (i<15) do
begin
  i :=i+1;
  Write(i, ' , ');
End;
```