Entête
Nom Algorithme (arguments)
données en référence
données
Variable locale
Résultat

Corps
Debut
:
Fin

— espace

Ex 1:

```
affichage ()
Résultat: afficher Hello World!
Debut
    afficher 'Hello world!'
Fin
```

Ex 2: Initialisation () — nom, type, rôle
Variable locale : x, entier, variable à initialiser.

```
Début
    x ← 5
Fin
```

Ex 3: Demande à l'utilisateur son nom (John), puis afficher le message 'Bonjour John, comment allez vous?'

```
Saisie ()
Variable locale : nom, texte
Resultat :                chaîne de caractères
Début
    Afficher 'Saisir votre nom'
    nom ← saisir
    Afficher 'Bonjour' + nom + 'Comment allez vous ?'
Fin
```

Ex 4: Initialiser 2 variables mettre à la puissance 2, puis afficher les 6 premières puissances de 2, successives.

```
Puissance ()
Variable locale : p2, entier, calcul puissance
Variable locale : n, entier
Résultat : premier termes de 2 à la puissance.
Début
    p2 ← 1
    n ← 0
    Afficher n + ' ' + p2
    n ← n + 1
    p2 ← p2 x 2
    Afficher n + ' ' + p2
    n ← n + 1
    p2 ← p2 x 3
    Afficher n + ' ' + p2
```

Concevoir un algo qui initialise 2 var ent ~~x et y~~ à l'aide de la saisie - utilisateur puis affiche le résultat du produit sous forme

Ex 6 : Concevoir l'algo qui transpose les valeurs de 2 variables entières
1) en utilisant une variable intermédiaire
2) sans variable intermédiaire.

Ex 6 : Concevoir l'algo qui initialise 2 variables entiers x et y à l'aide de la saisie utilisateur puis affiche le résultat du produit sous la forme par ex : x = 2, y = 3, '2 × 3 = 6'

1ère Version Utilise une variable intermédiaire 3 pour stocker le résultat avt de l'afficher.

2ème Version Pas de variable 3.

1er ★ | 2ème ★

Produit10
Variable locale : x, entier
y, entier
~~3, entier~~ stocker le résultat.
Resultat : calculer et afficher x × y
Début
Afficher 'Saisisser x'
x ← saisi
Afficher 'saisisser y'
y ← saisi
~~3~~ ← x * y
Afficher x + ' _ x _ ' + y + ' _ = _ ' + ~~3~~ (x+y)

Ex 6 : Concevoir un algo qui transpose les valeurs de 2 var entière
1) en utilisant une variable intermédiaire
2) sans variable intermédiaire.

Transpose()
Variable locale x, entier
y, entier
3, entier,
Resultat :
Début
Afficher 'Saisisser x'
x ← saisi
Afficher 'Saisisser y'
y ← saisi
3 ← x
x ← y
y ← 3
Afficher x
Afficher y
Fin

Début
Afficher 'Saisisser x'
x ← saisi
Afficher 'Saisisser y'
y ← saisi
x ← (x+y) - x
y ← (x+y) - y
Afficher x
Afficher y
Fin

Ex 7 : Concevoir l'algo qui affiche les 10 premiers termes de la suite de Fibonacci en
1) utilisant 3 variable
2 variable

Fibonacci ()
Variable locale : u, entier
v, entier
w, entier
Résultat : terme de la suite de fibonacci

Début
u ← 0
v ← 1
Afficher 'F(' + v + ') = ' + v
w ← u + v
u ← w
w ← u + v
v ← w
w ← u + v
u ← w
w ← u + v

v ← v + w
u ← v − u
v ← v + u
u ← v − u
v ← v + u

Fin

Voir Impair (x: entier)
donnée : x, entier, vérifier la parité de x
Début
Si (x mod 2 = 0) alors afficher x + ' est pair !'
Sinon alors afficher x + ' est impair !'

Supérieur Inférieur (x: entier)
donnée : x, entier, vérifier supérieur ou inférieur à y
Début y, entier
Afficher 'Saisir y'
y ← saisi
Si (x > y) alors afficher x + ' est supérieur à y'
Sinon alors afficher x + ' est inférieur à y'
Fin

Passage (a: entier, b: entier)
donnée : y, entier

Début
~~Afficher 'Saisir y'~~
~~y ← saisi~~

Si (a > b)
alors Afficher 'Saisir y' ★
y ← saisi
Sinon
alors Afficher 'Erreur'

★ Si (y > a) and (y < b)
alors Afficher 'y compris entre a et b'

Fin

Afficher(x: entier, y: entier, 3: entier)
Données: x, y, 3: entier
Variable locale: a, entier

Debut
Si (3 < y) et (3 > x)
alors Afficher 'Vrai'
Sinon
alors Afficher 'Saisir a'
a ← saisir
3 ← (x + y + a)
Afficher 3

Fin

Ex4: Age et Sexe

- nom
- age - si < 16 = fille
= garçon
Sinon = homme
= femme

Algo: AgeSexe()
Variable locale: annee, entier, année courante
nom, texte, nom de l'utilisateur
naissance, entier, année de l'utilisateur
sexe, caractère, caractère de l'utilisateur
Resultat, Presentation de l'utilisateur

Debut
Afficher "année"
annee ← saisir
Afficher "Nom"
nom ← saisir
Affiche "année de naissance (a a a a)
naissance ← saisir
Afficher "sexe (H ou F)"
sexe ← saisir
Si (annee - naissance) < 16 alor
Si sexe = 'H' alors
Afficher 'Bonjour ␣' + nom + ', tu as ␣' + (annee - naissance)
Sinon Afficher + 'ans et tu est un garçon'.
'Bonjour ␣' + nom + ', tu as ␣' + (annee - naissance)
+ 'ans et tu est un fille'.
Fin si

Fin

Ex5 Comparaison(): boolean
variable locales: x, entier
y, entier
z, entier
Debut
Afficher 'Saisir x'
x ← saisir
Afficher 'saisir y'
y ← saisir
Afficher 'saisir z'
z ← saisir
Si (x = y et y = z) alors retourner Vrai
sinon retourner Faux
finsi
Fin

Ex 6 Fonction () booléen
Variables locales : x, entier
y, entier
z, entier
Resultat comparaison entre 3 entiers.
Debut Afficher 'saisir x'
x ← saisir
Afficher 'saisir y'
y ← saisir
Afficher 'saisir z'
z ← saisir
Si (x = y ou y = 3 ou x = 3 alors) retour Vrai
Sinon
retourner faux
fin si
fin

Ex 8 Calculatrice ()
variable locale x, entier
y, entier
op, caractère Resultat : affiche des calcul d'une calculatrice
Debut Afficher 'saisir x'
x ← saisir
Afficher 'saisir y'
y ← saisir
Afficher 'saisir op'
op ← saisir Afficher x + op + y + '='
Si (op = '*') alors
Afficher (x*y) Afficher x + '*' + y + '=' (x*y)
Si (op = '/') alors
Si (y ≠ 0) alors
Afficher (x/y) Afficher x + '/' + y + '=' (x/y)
Sinon Afficher 'operation impossible'
Si (op = '+') alors
Affiche x + '+' + y + '=' (x+y)
Si (op = '-') alors
Afficher x + '-' + y + '=' (x-y)
Fin si
Fin

Concevoir 1 algo qui
1/ Affiche un menu comportant les choix a, b, c proposés à l'utili
2/ En fonction du choix de l'utili afficher
a. Execute Hello World et revient en 1
b. Execute How Are You et revient en 1
c. Termine le programme
d. Choix incorrect et revient en 1

BOUCLE
1/ Tant que (cond) faire
| Actions
fin tant que

2/ faire
| action
tant que (cond)

3/ Pour i allant de x à y
faire
| actions
fin pour

Menu()
Variable locale: choix, caractère, choix de l'utilisateur
Resultat: affichage d'un menu

Debut
faire
Afficher 'choisir : a ... b ... c ...'
choix ← saisir
Si choix = 'a' alors
| Affiche 'Ecrire Hello World'
Sinon
| Si choix = 'b' alors
| | Afficher 'Ecrire How Are you'
ou | Sinon
| | Si choix = 'c' alors
| | | Afficher 'Programme terminé'
| | Sinon Afficher 'choix incorrect'
| | Fin si
| Fin si Tanque (choix ≠ 'c')
fin

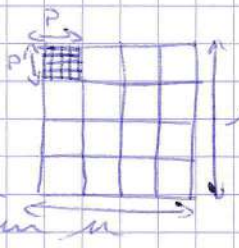
⇒ Selon (choix)
| cas 'a'
| cas 'b'
| cas 'c'
Défaut: affiche 'choix incorrect'
fin

Fibonacci()
I, entier
n, entier, nbr de terme
Variable locale: a, entier
b, entier
x, entier
Resultat: terme de la suite de fibonacci
Debut
| a ← 0
| b ← 1
| Afficher (Saisir n: nbr de terme)
| n ← saisir
| Pour I allant de 2 à n faire
| | Debut pour
| | x ← a + b
| | a ← b
| | b ← x
| | Fin pour
| Afficher x
Fin

Exo 4. TD n°3

Dessiner un damier de taille n x n dont chaque cellule est constituée de p x p - Le caractère blanc ' '
Le caractère noir '*'

1) Dessiner en blanc sur fond noir un damier simple, chaque pixel allumé en couleur avec ses voisins.



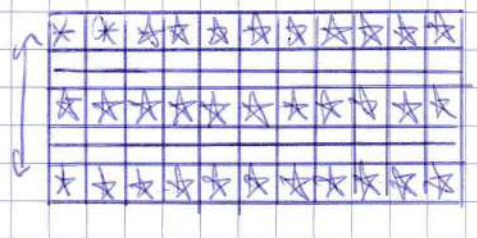
Damier (n: entier, p: entier)
données : n: entier
p: entier
variables locales : i, entier compteur
j, entier compteur

Début pour i allant de 1 à n x p inclus faire
pour j allant de 1 à n x p inclus faire
si (i+j) mod 2 = 0 alors
afficher ' '
sinon afficher '*'
fin si
fin pour
fin pour
Affiche "↵"
fin

Ou

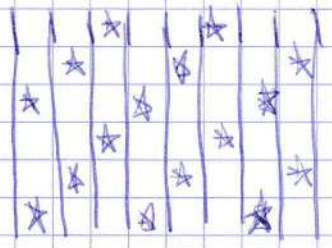
Données : n: entier
p: entier
Variables : i entier
j entier
Début Pour i allant de 1 à n x p
Pour j allant de 1 à n x p
Si i mod 2 = j mod 2 alors
Afficher (" ")
Sinon Afficher ("*")
Fin si
Fin pour
Fin pour
Afficher ("↵")
Fin

Exercice 4:
Une série de ligne horizontale à raison d'une ligne



D1-3 (n entier p entier)
Pour i allant de 1 à n x p par pas de 3 faire
Pour j allant de 1 à n x p faire
Si i mod 3 = 1 alors affiche('*')
fin si
fin pour
Affiche ("↵")
fin pour
fin

Une suite de lignes diagonales montantes à raison d'une sur 3.



Pour i allant de 1 à nxp faire
  Pr j allant de 1 à nxp faire
    Si (i+j) mod 3 = 1 alors
      Afficher('*')
    Fin si

Demander à l'utilisateur de saisir des nbres positifs ou nuls et les sommer à mesure qu'il les saisit.
Qd l'util saisit un nbr negatif l'algo se termine après avoir affiché la somme des nbres positifs.
Si la première saisie est un nbr negatif le prog retourne 0.

Variable x entier x=0
  y entier
~~Pr y>0 faire~~
Afficher "saisir un nbr"
y ← saisir nbr
Pr y>0 faire
~~si y positif~~
x = x+y

~~Sinon~~ retourne x.
~~fin pr~~
~~Si y<0~~
~~retourner 0~~
return 0.
Fin

SommePositif()
Variables x entier
  y entier
Debut x ← 0
Afficher "Saisir nbr positif"
y ← sai
Si y < 0 abs retourner 0
Tan que y ≥ 0 faire
x ← x + y
Afficher saisir y'
y ← saisi
Afficher la somme "+x"

Ex6
Concevoir un algo qui génère la suite de Syracuse.

Syracuse()
Variables locales : S, entier naturel, terme de la suite
  n, entier, le rang
Début n ← 0
Afficher saisir le premier terme de la suite de Syracuse
S ← saisi
Tantque S ≠ 1 faire
Afficher 'S('+n+') = '+S
Si S est pair alors
S ← S/2
Sinon S ← 3*S+1
fin tant que

Ex 3

```
moyenne_serie()                                   var
Var loc : nb_series, entier, contient le nbr de séries
          i, entier compteur séries(i)
          n, entier, nbr de valeur associée à chaque serie
          val, entier, valeur saisie par l'utilisateur
          somme, entier, calcul somme des val.
Début Afficher 'Saisir le nbre de series'
      nb_serie ← saisir

      pour i allant de 1 à nb_serie inclus faire
          Afficher 'Saisir le nombre de valeurs pour la série ' + i
          n ← saisir
          somme ← 0
          pour j allant de 1 à n inclus faire
              Afficher 'saisir la valeur ' + j
              val ← saisir
              somme ← somme + val
          fin pour
          Afficher la moyenne des ' + n + ' valeurs = ' + somme/n '
      fin pour
```

Ex 4

	p=0	p=1	p=2	p=3	p=4
n=0	1				
n=1	1	1			
n=2	1	2	1		
n=3	1	3	3	1	
n=4	1	4	6	4	1

Coefficient binomial

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

```
factoriel (n: entier naturel) entier naturel donnée n
Var loc : i entier, compteur boucle
          f entier naturel
Début
    f ← 1
    pour i allant de 1 à n inclus faire
        f ← f × i
    fin pour
    retourner f
fin
```

```
Pascal (n: entier)
Données: n, entier, nbr de ligne des triangles de pascal
var loc : p entier indice de chaque colonne
          i entier, compteur
Début pour n allant de 0 à n-1 inclus faire
    pour p allant de 0 à i inclus faire
        Afficher factoriel(i) / (factoriel(i-p) × (factoriel(p))
    fin pour
    Afficher '↵'
fin pour
Fin
```

Autrement
Pascal (n)
(
n
Début pour i allant de 0 à n-1 inclus faire

#define TAILLE 100 //declarat° de la cte taille
int tab[TAILLE]

Ex 1. (n TD)
Concevoir un algo qui permet de remplir un tableau de taille n en affectant à chaque cellule l'indice de la boucle.

Tableau (n: entier) : tableau entier
donné n, entier taille du tableau
var loc : tab = (tab_i)_{i∈[1;n-1]} tableau d'entiers
i entier compteur.

Début
| pour i allant de 0 à (n-1) inclus faire
| | tab_i ← i
| fin pour
| retourn tab;
Fin

| 1 | 2 | 3 | 4 | 5 | 6 |

Ex 2
un algo qui demande à l'utilisateur de saisir n valeurs pour remplir le tab.

Début pour i allant de 0 à (n-1) inclus faire
afficher "saisir la valeur" + (i+1)
tab_i ← saisir
fin pour
fin

Ex 3 Afficher le tableau
affichage (n: entier, tab: tableau d'entier)
donnée, n entier, taille tableau
var loc i entier compteur
Début pour i allant de 0 à (n-1) inclus faire
afficher tab_i + " "
fin pour
fin

Ex 4 Concevoir un algo qui calcule et retourne la moyenne, du tableau d'entiers.

Moyenne (n, entier, tab: tableau d'entier) : ~~tableau~~ entier
donnée n entier taille tableau
var loc tab = (tab_i)_{i∈[1;n-1]}
i entier compteur
moyenne, somme : entier
Début pour i allant de 0 à (n-1) inclus faire
tab_i ← i
somme ← somme + tab_i
moyenne ← 1/n somme
fin pour
Fin return moyenne.

Début moyenne ← 0
pour i allant 0 à (n-1) faire
moyenne ← moyenne + tab(i)
fin pour

TD n°5

Exercice 2

T | 5 | 3 | 2 | 4 | 6 |

Tableau (t : tableau d'entier ⓪, n : entier, ~~x entier~~) : entier
données t=(ti) i∈[0 ; n-1]
n, entier taille tableau = 5
var loc : i, entier compteur boucle
cpt, entier compteur le nbr élément pair

Début cpt ← 0
pour i allant de ⓪ à (n-1) inclus faire
| si (ti est pair) alors cpt ← cpt + 1
| (ti mod 2 = 0)
| fin pour
retourner cpt ;
fin

Exercice 4

a	1	2	3	4
1	2-1 = 1	3-2 = 1	4-3 = 1	4-1 = 3
2	1-1 = 0	2-1 = 1	3-2 = 1	3-1 = 2

$a_{i+1,j} = |a_{i,j+1} - a_{ij}|$
$a_{i+1,n} = |a_{i0} - a_{in-1}|$

Prettyprint (t) :

0	0	0	2

Prettyprint (a tableau ⓪, n : entier)
donnée a = (aij) i∈[0 .. n[
j∈[0 .. n[
var loc : i, entier compteur boucle (lignes)
j, entier compteur boucle (colonnes)
n, nbr de lignes/colonnes (n×n)

Début
pour i allant de 0 à (n-1) faire
pour j allant de 0 à (n-1) faire
si (i=0) alors aij ← j+1
si (j=n-1) alors ai+1,j ← |ai0 - aij|
ai+1,j ← |ai+1,j - aij|
fin pour
fin pour
fin

#include <math.h>
en C valeur absolu fct° abs

Premier (int x) : booléen
données : x, entier
var loc :
Résultat : retourne vrai si x est un nbr premier sinon faux

Début si (x=0 ou x=1) alors retourner faux
si (x = 2) retourner vrai

pour i allant de 2 à (x^1/2) inclus faire (x-1) ou (x/2) ou (√x)
si (x mod i = 0) alors
retourner faux
fin pour
retourner vrai

Exercice 5 : Duplicat° d'un tableau

```
Duplication (a tableau 2D, b tableau 2D, n: entier)
Donnée n, entier
       tab1 = (tab i,j) i ∈ [0, n-1] et j [0, n-1]
Donnée tab2 = (tab2)
var loc : i, entier, compteur de boucle

Debut pour i allant de 0 à (n-1) inclus faire
        pour j allant de 0 à (n-1) inclus faire
            tab2 i,j ← tab1 i,j
        fin pour
    fin pour
fin
```

Ex : Concevoir un algorithme qui retourne la ~~variance~~ variance d'un tableau

$$V = \frac{1}{n} \sum_{i=0}^{n-1} (t_i - m)^2 \qquad m = \text{moyenne des élément du tableau} = \frac{1}{n} \sum_{i=0}^{n-1} t_i$$

```
Variance (t: tableau entier, n: entier) : ~~entier~~ reel
  données n, entier, t=(t_i) i ∈ [0; n-1]
  var loc : i, entier, compteur de boucle
            m, réel moyenne   v, réel variance
Début
    v ← 0
    m ← moyenne // appel de l'algo de calcul moyenne
    pour i allant de 0 à (n-1) inclus faire
        v ← v + (t_i - m)(t_i - m)
    fin pour
    v ← v/n
    retourner v
fin
```

Ex : Concevoir un algo qui retourne l'indice du premier élément strictement négatif dans un tableau d'entiers.
Si aucun élément négatif dans le tableau, retourner (-1)

```
Indice (t: tableau entiers, n: entier) : ~~ind~~ entier
Données n, entier
        t=(t_i) i ∈ [0, n-1]
var loc : i, indice, entier compteur

Debut  pour i allant de 0 à (n-1) inclus faire
           si (t_i < 0)
               retourner i
           fin si
       fin pour
       retourner -1
fin
```

TD5 Ex 3

Concevoir un algo qui demande à l'utilisateur de remplir un tableau à 2D de dimensions n x m.

remplissage (n: entier, m: entier): tableau 2D d'entiers
donnée n, entier, nbr ligne
m, entier, nbr colonnes
var loc tab = (tab$_{i,j}$)$_{\substack{i \in [1, n-1] \\ j \in [1, m-1]}}$ tableau d'en d'enter

Début pour i allant de 0 à (n-1) inclus faire
| pour j allant de 0 à (m-1) inclus faire
| | tab$_{i,j}$ ← saisir Afficher tab$_{i,j}$ + " "
| fin pour
fin pour
retourn tab / Afficher 'd' réfléchir Carré Lagrange
Fin

Tp n°5 suite

Exo Concevoir un algo qui inverse un tableau d'entier 1D

t	6	3	2	1
	0	1	2	3

inverse t

t	1	2	3	6
	3	2	1	0

Echange ~~Inverse tab~~ (t: tableau entier, n: entier): entier
Données n, entier
$t = (t_i)_{i \in [0 \text{ à } n-1]}$
var loc : i, entier compteur 1ère variable
i2, entier compteur 2ème variable
Début
$t_{i2} \leftarrow t_{i1} + t_{i2}$
$t_{i1} \leftarrow t_{i2} - t_{i1}$
$t_{i2} \leftarrow t_{i2} - t_{i1}$

Inverse tab (t: tab entier, n: entier)
données $t = (t_i)_{i \in [0 \ldots n-1]}$
var : i entier compteur
Début
| pour i allant de 0 à $(\frac{n-1}{2})$ faire
| | echange (t, i, n-1-i)
| fin pour
fin

TP5 ALGO

Ex 8 – Recherche Dichotomique

Tabdicho(tab: tableau d'entiers, n: taille tableau, val: entier) entier

Données: tab = (tab$_i$)$_{i\in[0,n]}$ tableau 1D trié
n, entier, taille du tableau
val, valeur à chercher, entier

variable locale: deb, entier, indice de début
fin, entier, indice de fin
milieu, entier, désigne le milieu de l'intervalle de recherche

Résultat: retourne l'indice de val si la valeur existe dans le tableau sinon retourne −1

1 faire algo de tri de tableau
Avt dichotomie

```
Début  deb ← 0
       fin ← n-1
  Tq (deb <= fin) faire
    milieu ← (deb + fin) / 2
    si (tab_milieu = val) alors
      retourner milieu
    sinon
      si (tab_milieu < val) alors
        deb ← milieu + 1
      sinon fin milieu
      fin si
    fin tant que
    retourner -1 // valeur val n'existe pas dans tab.
  fin
```

Exo ⊕

Calcul du nombre d'occurences d'un élément. (nbr de fois qu'il apparait)

tab: tableau 1D d'entier de taille n

Tabnbrxélement(tab: tableau d'entiers 1D, n: entier, e, entier) entier

Données: tab = (tab$_i$)$_{i\in[0,n]}$ taille n
n = entier taille tableau
e = entier, élément d'occurence

Var loc: cpt, compteur entier
i, entier, compteur

```
Début  cpt ← 0
  pour i allant de 0 à n-1 inclus faire
    si tab i = e alors
      cpt ← cpt + 1
  fin pour
  retourner cpt
```

Ex 10

tab	1	2	3	1	2	3	1	3
t	1	2	3	4	5	6	1	2
res	1	2	3	3				

fct° occurences = tbx élément

Intersection (tab: tableau 1D d'entier de taille n,
t : tableau 1D d'entier de taille m):
tableau d'entiers de taille n -
données : tab = (tab_i)_{i∈[0,n[} occ nbr occurence tab
t = (t_i)_{i∈[0,m[} occ " " t
var loc : res = (res_i)_{i∈[0,n[} occ, nbr occurence le + petit entre les 2 tab
résultat - retourne un tableau des éléments communs -

début
  pour i allant de 0 à (n-1) inclus faire
    res_i ← 0
  fin pour
  pour i allant de 0 à (n-1) inclus faire
    e ← tab_i
    occ1 ← occurence (tab, n, e)
    occ2 ← occurence (t, m, e)
    si (occ1 = occ2) alors
      occ ← occ2
    sinon occ ← occ1
    si (occ > occ3) alors
      res_k ← tab_i
      k ← k+1
    fin si
  fin si
  fin pour
  retourner res
fin

Ex 11 -

Partition (tab: tableau 1D d'entier de taille n,
données : tab = (tab_i)_{i∈[0,n[} ; n, taille tableau
var loc : ~~m~~, entier compteur
m, entier, nbr de valeur strictement
occ, entier, nbre d'occurence
début m ← 0 pour i allant de 0 à n-1 inclus faire
  occ ← occurence à (tab_i, tab, n, i+1)
  m ← m+1
  fin pour
  retourner n
  retourner à m -
fin -

---

occurence à (tab, tableau 1D d'entier de taille n, e, i) entier
donnée : tab = (tab_i)_{i∈[0,n[}
e, entier élément recherché
i, entier, indice

occurence_i ...
var loc : j, entier compteur
cpt, entier nbr d'occurence
Résultat : retourner cpt

Debut cpt ← 0
  Pour j allant de i à (n-1) inclus faire
    si tab=e alors
      cpt ← cpt+1
    fin si
  fin pour
  retourne cpt
fin

---

occ_tab (t: tableau 1D d'entiers de taille n) : tableau d'entiers
données : t=(t_i)_{i∈⟦0,n⟦}
n, entier, taille du tableau.
var loc : i, entier compteur
occ = (occ_i)_{i∈⟦0,m⟦}
m = valeur max du tableau

Début m ← max_tab(t,n)
  pour i allant de 0 à (m-1) inclus faire
    occ_i ← 0
  fin pour
  pour i allant de 0 à (n-1) inclus faire
    e ← tab_i
    occ_e ← occ_e + 1
  fin pour
  retourner occ
fin