

**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L’ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**ECOLE NATIONALE SUPERIEURE D’INFORMATIQUE**

**CPI 1 Année 2016 / 2017**

**TP**

**EN ALGORITHMIQUE ET STRUCTURES DE DONNEES STATIQUES (ALSDS)**

**REALISE PAR :**

**DARSOUNI Lotfi Rdjem**

**BENBAKHTA Mohamed Amine**

**Section : C**

**Groupe N° : 08**

Semestre 1

**Table des matières :**

[I-Introduction : 3](#_Toc474956341)

[II-Solution du TP1 : 3](#_Toc474956342)

[II-1-Découpage modulaire : 3](#_Toc474956343)

[II-1-1-Module Kaprekar : 3](#_Toc474956344)

[II-1-2-Module Automorph : 3](#_Toc474956345)

[II-1-3-Module Chifimp : 4](#_Toc474956346)

[II-2-Algorithme principal : 4](#_Toc474956347)

[II-2-1-Analyse : 4](#_Toc474956348)

[II-2-2-Algorithme : 5](#_Toc474956349)

[II-2-3-Programme : 6](#_Toc474956350)

[II-3-Conception des modules : 7](#_Toc474956351)

[II-3-1-Module Kaprekar : 7](#_Toc474956352)

[II-3-2-Module Automorph : 8](#_Toc474956353)

[II-3-3-Module Chifimp : 9](#_Toc474956354)

[III-Solution du TP2 : 10](#_Toc474956355)

[III-1-Découpage modulaire : 10](#_Toc474956356)

[III-1-1-Module Kramer3 : 10](#_Toc474956357)

[III-1-2-Module Determin3 : 11](#_Toc474956358)

[III-1-3-Module RempCol : 11](#_Toc474956359)

[III-2-Algorithme Principal : 11](#_Toc474956360)

[III-3-Conception des Modules : 14](#_Toc474956361)

[III-3-1-Module Kramer3 : 14](#_Toc474956362)

[III-3-2-Module Determin3 : 16](#_Toc474956363)

[III-3-3-Module RempCol : 17](#_Toc474956364)

[IV-Utilisation de la bibliothèque : 19](#_Toc474956365)

[V-Conclusion : 20](#_Toc474956366)

I-Introduction :

Depuis la nuit des temps, l’homme a toujours été fasciné par le monde des maths de par ses mystères auxquels il voue une passion obsessive à élucider. Parmi ces mystères : « Les chiffres ». Ces derniers regorgent de propriétés chacune plus étonnantes et impressionnantes que les autres, époustouflant les mathématiciens et captivant ainsi leur curiosité au point d’y consacrer toute une vie.

Nous allons donc nous intéresser dans le premier TP à quelques nombres aux propriétés magiques, et enchaîneront dans le deuxième en abordant la résolution d’un système d’équations d’ordre 3 en utilisant la méthode de Kramer.

II-Solution du TP1 :

Dans ce premier TP nous allons étudier les nombres de Kaprekar, les nombres automorphes et les nombres composés de chiffres impairs. Un nombre de Kaprekar est un nombre qui est égal à la somme de deux parties non nulles de son carré, un nombre automorphe est un nombre qui se trouve à la fin de son carré, et les nombres composés de chiffres impairs sont les nombres composés de 1,3,5,7 et 9. Il nous est demandé de trouver quelques-uns de ses nombres.

II-1-Découpage modulaire :

II-1-1-Module Kaprekar :

Pour savoir si un nombre est de Kaprekar on aura besoin d’un module qui donne Vrai si un nombre est de Kaprekar et de nous donner les deux parties dont est composé ce nombre.

**Procédure**

Kaprekar

P1 : entier

Kapr :Booléen

N :entier

P2 : entier

**Rôle :** Kapr contiendra Vrai si N est un nombre de Kaprekar (il est égal à la somme des 2 parties non nulles de son carré) et P1 et P2 seront ses deux parties.

II-1-2-Module Automorph :

Pour savoir si un nombre est automorphe, on aura besoin d’un module qui donne Vrai si un nombre est automorphe.

**Fonction**

Automorph

Booléen

N:entier

**Rôle :** Donne Vrai si N est un nombre automorphe (s’il se trouve à la fin de son carré)

II-1-3-Module Chifimp :

On aura besoin d’un module qui donne Vrai si un nombre est composé que des chiffres impairs 1,3,5,7,9 sans répétition.

**Fonction**

Chifimp

Booléen

N:entier

**Rôle :** Donne Vrai si N est un nombre est composé des chiffres 1,3,5,7 et 9.

**Nota :** Les modules précédents utilisent des modules déjà faits en classe qui sont :

**Fonction Nbpos (n : entier) : entier**

//Rôle : donne le nombre de positions de N

**Fonction Puiss (n, p : entier) : entier**

//Rôle : donne N à la puissance P

**Fonction Distinct (n : entier) : booléen**

//Rôle : donne vrai si N est composé de chiffres distincts.

II-2-Algorithme principal :

II-2-1-Analyse :

🡺On fait varier i de 5000 à 10000

🡺Si i est un nombre de kaprekar on affiche i, son carré, et ses deux parties.

🡺On fait varier i de 1 à 1000

🡺Si i est un nombre automorphe alors on affiche i et son carré.

🡺On fait varier i de 10000 à 99999

🡺Si i est composé des chiffres impairs on affiche i et on incrémente deux variables :

* Une variable Cpt qui représente le nombre de solutions (Cpt recevra Cpt + 1).
* Une variable Somme qui représente la somme des solutions (Somme recevra Somme+i).

🡺 On affiche le contenu de Cpt et Somme.

II-2-2-Algorithme :

Algorithme TP1

Variable  i, cpt, somme, p1, p2 : entier ; kapr : booléen.

Fonction Automorph (n : entier) : booléen

Chifimp (n : entier) :booléen

Procédure Kaprekar (n : entier ; var p1,p2 : entier ; var kapr : booléen)

Ecrire (‘Les nombres de kaprekar compris entre 5000 et 1000 sont :’)  
Pour i allant de 5000 à 10000 faire

Dpour

Kaprekar(i, p1, p2, kapr)  
 Si Kapr=Vrai Alors  
 Ecrire (i,’ ‘,i\*i,’ ‘,p1,’ ‘,p2)

Fpour

Ecrire (‘Les nombres automorphes compris entre 1 et 1000 sont : ‘)  
Pour i allant de 1 à 1000 faire  
 Si Automorph(i)=Vrai Alors  
 Ecrire (i,’ ‘,i\*i,’ |’)

Ecrire (‘les nombres compose de tous les chiffres impairs sont : ‘)  
Pour i allant de 10000 à 99999 faire

Dpour

Si Chifimp(i)=vrai Alors

Dsi  
 Ecrire(‘ ‘,i,’ |’)  
 Cpt 🡸 cpt +1   
 Somme 🡸 Somme +i  
 Fsi

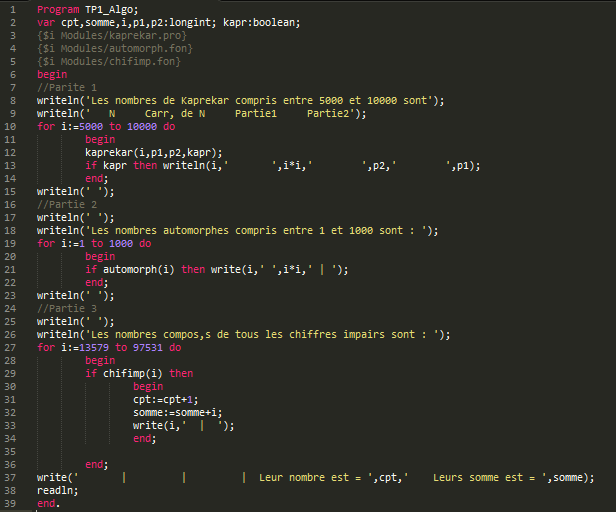
Fpour

Ecrire (‘leurs nombre est :’,Cpt,’ Leurs somme est : ‘,Somme)

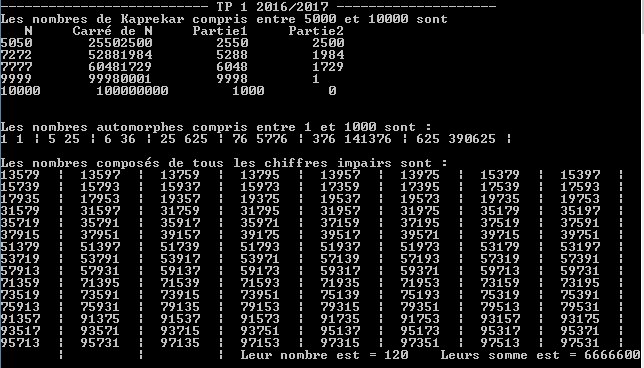
**Début**

**Fin**

II-2-3-Programme :



**Résultat**

****

II-3-Conception des modules :

II-3-1-Module Kaprekar :

**Analyse :**

🡺Une Variable P recevra le produit de N\*N

🡺Kapr sera initialisé à Faux

🡺 Pour j allant de 1 au nombre de positions de N on fait ce qui suit : //Pour prendre toutes les parties possibles de P

🡺p1 recevra p modulo 10 à la puissance j //pour prendre la première partie de P

🡺p2 recevra p divisé par 10 à la puissance j //pour prendre ce qui reste de P

🡺 si la somme de p1 et p2 est égal à N alors kapr recevra Vrai

**Algorithme :**

Procédure Kaprekar(n ;entier ; var p1,p2 : entier ; var kapr :booléen)

Variable j,p :entier

Fonction Nbpos(n :entier) :entier

Puiss(n,p :entier) :entier

Kapr 🡸 Faux

P 🡸 n\*n

Pour j allant de 1 à Nbpos(n) faire

Dpour

P1 🡸 p mod Puiss(10,j)

P2 🡸 p div Puiss(10,j)

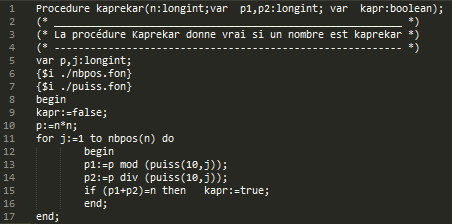
Si (p1+p2)=n Alors Kapr 🡸 Vrai

Fpour

**Début**

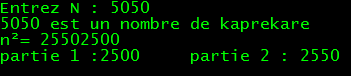
**Fin**

**Programme :**

****

**Jeu d’essai :** 5050 et 6542

**Résultat**

 essai2Kapr

II-3-2-Module Automorph :

**Analyse :**

🡺Une variable P recevra le produit de n\*n

🡺Si p modulo 10 à la puissance le nombre de positions de n est égal à n alors la fonction recevra Vrai sinon elle recevra faux

**Algorithme :**

Fonction Automorph(n :entier) :booléen

Variable p :entier

Fonction Nbpos(n :entier) :entier

Puiss(n,p :entier) :entier

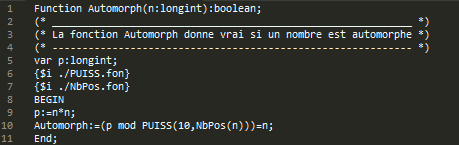
**Début**

P 🡸 n\*n

Si p mod (Puiss(10,Nbpos(n))) = n alors Automorph 🡸 Vrai Sinon Automorph 🡸 Faux

**Fin**

**Programme :**



**Jeu d’essai :** 25 et 49

**Résultat**

essaiAutomorph1 EssaiAutomorph2

II-3-3-Module Chifimp :

**Analyse :**

🡺Si n est composé de chiffres distincts on fait ce qui suit //On utilise le module distinct

🡺Pour i allant de 1 au nombre de positions de n on fait ce qui suit

🡺Si n est impair alors n recevra n divisé par 10

//Ceci nous permet de savoir si tous les chiffres composant n sont impairs

🡺 si N est égal à 0 à la fin de la boucle cela veut dire qu’il a été divisé entièrement donc tous ses chiffres sont impairs donc la fonction recevra Vrai sinon elle recevra faux.

**Algorithme :**

Fonction Chifimp (n : entier) : booléen  
Variable i : entier  
Fonction Nbpos (n : entier) : entier  
 Distinct (n : entier) : entier

**Début**

Si Distinct(n)=Vrai Alors

Dsi

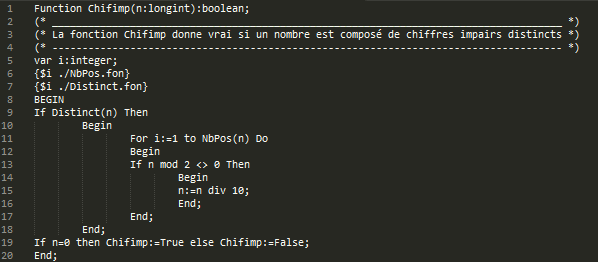
Pour i allant de 1 à Nbpos(n) faire  
 Si (n mod 2) <> 0 Alors n🡸 n div 10

Fsi

Si n=0 Alors Chifimp🡸Vrai sinon Chifimp🡸 Faux

**Fin**

**Programme :**



**Jeu d’essai :** 15397 et 25698

**Résultat**

essaiIChifimp1 essaiIChifimp2

III-Solution du TP2 :

Dans le deuxième TP nous allons concevoir un programme de résolution d’un système d’équations d’ordre 3 grâce aux matrices via la méthode de Kramer. Cette méthode consiste à mettre les coefficients des inconnus dans une matrice A de 3x3 et les termes constants dans un vecteur B de 3x1. On calculera ensuite le déterminant de la matrice A qu’on nommera D. on remplacera ensuite la première colonne de A par le vecteur B et on calcule le nouveau déterminant Dx. La solution X sera le résultat de Dx/D. Et ainsi de suite pour les solutions Y et Z en remplaçant, respectivement, la deuxième et la troisième colonne de A par le vecteur B.

III-1-Découpage modulaire :

III-1-1-Module Kramer3 :

On aura besoin d’un module qui nous permet de savoir si le système d’équations admet une solution et si c’est le cas il nous les fournit

**Procédure**

M : Tab2d

VAR n :booléen

Kramer3

VAR x : réel

R : Tab2d

VAR y : réel

VAR z : réel

**Rôle :** Donne les solutions (x, y, z) d’un système d’équations d’ordre 3 selon la méthode de Kramer. M(3,3) contiendra les coefficients des inconnus et R(3,1) les termes constants. Si n est faux alors la matrice est singulière.

III-1-2-Module Determin3 :

On aura besoin d’un module qui calcule le déterminant d’une matrice d’ordre 3.

**Fonction**

Determin3

M : Tab2d

Réel

**Rôle :** Donne le déterminant d’une matrice M d’ordre 3

III-1-3-Module RempCol :

On aura besoin d’un module qui remplacera une colonne de la matrice des coefficients par le vecteur des termes constants

**Procédure**

T : Tab2d

L : entier

RempCol

VAR T : Tab2d

C : entier

Col : entier

V : Tab2d

**Rôle :** remplace la colonne col du tableau T(L,C) par le vecteur colonne V(L,1)

**Nota :** on aura aussi besoin de modules déjà faits en cours :

**Procédure Lect2D(T :Tab2d ; l, m : entier)**

**//**Rôle : permet de lire un tableau à deux dimensions.

**Procédure Ecrit2D(T : Tab2d ; l, m : entier)**

**//**Rôle : permet d’afficher à l’écran un tableau à deux dimensions d’une manière soignée.

III-2-Algorithme Principal :

**Analyse :**

🡺On lit la matrice des coefficients A

🡺On lit le vecteur des termes constants B

🡺On affiche les deux matrices A et B

🡺On appelle la procédure Kramer3

🡺Si le N qu’elle renvoie est égal à Vrai Alors :

🡺On affiche les trois solutions x, y, z.

🡺Sinon

🡺La matrice est singulière il n’y a pas de solutions.

**Algorithme :**

Algorithme TP2

Type Tab2d=tableau [1..100 ,1..100] d’entier

Variable M, R :Tab2d **;** x, y, z :réel **;**   
 l, c, c1 :entier **;**  
 n : booléen ;

Procédure Kramer3(M, R : Tab2d , N : booléen , x, y, z :réel)

Lect2D(var T: Tab2d; var l, c : entier)

Ecrit2D( T : Tab2d ; l, c :entier)

**Début**

Lect2D(M, l, c)

Lect2D(R, l ,c1)

Ecrire (‘La matrice M des coefficients : ‘)

Ecrit2D (M, l, c)

Ecrire (‘le vecteur B ‘)

Ecrit2D (R, l ,c1 )

Kramer3(M, R, N, x, y, z)

Si N=Vrai Alors

Dsi

Ecrire(‘les solutions sont : ‘)

Ecrire (‘x = ‘, x)

Ecrire (‘y = ‘, y)

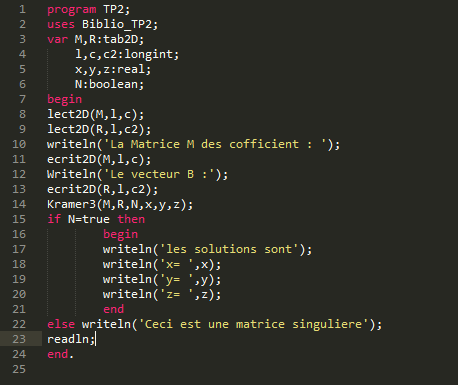
Ecrire (‘z = ‘, z)

Fsi

Sinon Ecrire (‘ Ceci est une matrice singulière’)

**Fin**

**Programme :**



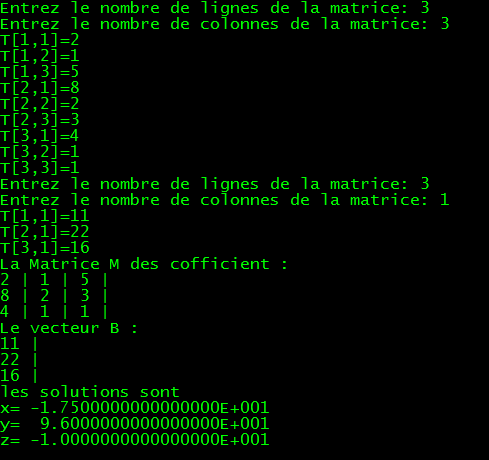
**Jeu d’essai :**

La matrice A : Le vecteur B :

|  |  |  |
| --- | --- | --- |
| **2** | **1** | **5** |
| **8** | **2** | **3** |
| **4** | **1** | **1** |

|  |
| --- |
| **11** |
| **22** |
| **16** |

**Résultat**



III-3-Conception des Modules :

III-3-1-Module Kramer3 :

**Analyse :**

🡺On initialise N à Vrai

🡺On affecte la matrice M dans deux variables intermédiaires M2 et M3 // qui nous serviront plus tard

🡺On calcule le déterminant D de la matrice M

🡺Si D est nul N recevra Faux et le déroulement s’arrête

🡺Sinon

🡺On affecte 1 à col //pour remplacer la première colonne

🡺On remplace la colonne col de M par le vecteur R //M est maintenant modifiée

🡺On calcule la solution X qui est égal à (déterminant de M) divisé par D

🡺On affecte 2 à col // pour remplacer la deuxième colonne

🡺On remplace la colonne col de M2 par le vecteur R // on utilise M2 car M a été modifié

🡺On calcule la solution Y qui est égal à (déterminant de M2) divisé par D

🡺On affecte 3 à col //pour remplacer la troisième colonne

🡺On remplace la colonne col de M3 par le vecteur R //on utilise M3 car M et M2 ont été modifié

🡺On calcule la solution Z qui est égal à (déterminant de M3) divisé par D.

**Algorithme :**

Procédure Kramer3(var M, R: Tab2d; var x, y, z; var n: booléen)

Variable col, l, c : entier  
 D : réel  
 M2, M3 :tab2d

Fonction Determin3 (M : Tab2d ) : réel

Procédure RempCol (var T :Tab2d ; var l, c, col :entier ; v : Tab2d)

M2 🡸M

M3 🡸M

N🡸Vrai

D🡸Determin3(M)

Si D=0 Alors N🡸Faux

Sinon

Dsinon

Col🡸1

RempCol (M, L, C, col, R)

x🡸Determin3(M)/D

col🡸2

RempCol (M2, L, C, Col, R)

y🡸Determin3(M2)/D

col🡸3

RempCol (M3, L, C, Col, R)

z🡸Determin3(M3)/D

Fsinon

**Début**

**Fin**

**Jeu d’essai : le jeu d’essai de Kramer3 est semblable à celui de l’algorithme principal car le test de Kramer3 est l’algorithme principal lui-même.**

III-3-2-Module Determin3 :

**Analyse :**

🡺On affecte à D1 la somme des produits des cases diagonales en partant du haut et en descendant (en considérant les 2 dernières lignes de la matrice comme les deux premières).

🡺On affecte à D2 la somme des produits des cases diagonales en partant du bas et en remontant (en considérant les 2 dernières lignes de la matrice comme les deux premières).

🡺On affecte à la fonction la différence entre D1 et D2.

**Algorithme :**

Fonction Determin3(t : Tab2d) :réel

Variable D1, D2 :réel

**Début**

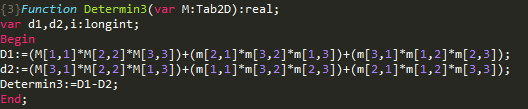
D1🡸(M[1,1]\*M[2,2]\*M[3,3])+(m[2,1]\*m[3,2]\*m[1,3])+(m[3,1]\*m[1,2]\*m[2,3])

D2🡸(M[3,1]\*M[2,2]\*M[1,3])+(m[1,1]\*m[3,2]\*m[2,3])+(m[2,1]\*m[1,2]\*m[3,3])

Determin3 🡸D1-D2

**Fin**

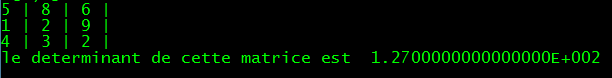
**Programme :**

****

**Jeu d’essai :**

|  |  |  |
| --- | --- | --- |
| **5** | **8** | **6** |
| **1** | **2** | **9** |
| **4** | **3** | **2** |

**Résultat**

****

III-3-3-Module RempCol :

**Analyse :**

🡺Pour i allant de 1 au nombre de lignes du tableau on fait ce qui suit :  
 🡺On affecte toute la colonne V[i,1] dans la colonne Col désignée du tableau T (T[i,col])

**Algorithme :**

Procédure RempCol (var T : Tab2d ; var L, c, col :entier ; V :Tab2d)

Variable i : entier

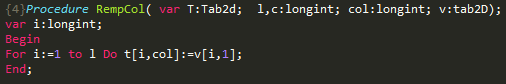
**Début**

Pour i allant de 1 à L faire

T[i, col] 🡸 V[i,1]

**Fin**

**Programme :**

****

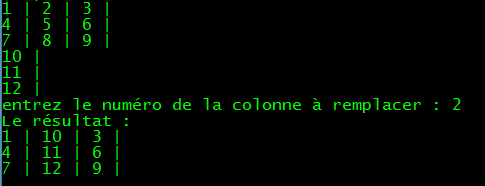
**Jeu d’essai :**

|  |
| --- |
| **10** |
| **11** |
| **12** |

La matrice A : Le vecteur B :

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

**Résultat**



IV-Utilisation de la bibliothèque :

V-Conclusion :

En conclusion, il existe une multitude de nombres fascinants dans le vaste univers des mathématiques, et c’est cette fascination qui a poussé l’homme à la recherche, lui permettant ainsi de découvrir ces nombres et d’étudier leurs propriétés, mais le travail fournit pour cela était trop important.

Aujourd’hui grâce aux algorithmes et à leurs programmes, ce genre de recherches devient bien plus banal, et en y combinant l’approche modulaire, le gain de temps et d’effort la transformerait presque en un jeu d’enfants.