UNIVERSITÉ PARIS DESCARTES

UFR MI

Programmation 2

Licence 1ère année année 2009 - 2010

Recueil
de
documents de travail

Pr. Nicole VINCENT

Table des matières

Programme du semestre

Semaine 1	Cours : révisions et sous-programmes
Semaine 2	Cours : sous programmes TD et TP : révisions : structures de contrôle
Semaine 3	Cours : types scalaires : énuméré - intervalle - ensemble TD et TP : les procédures
Semaine 4	Cours : types structurés : chaînes TD et TP : les fonctions
Semaine 5	Cours : types structurés : enregistrements TD et TP : sous-programmes **CC1** papier – 23 février
Semaine 6	Cours : types structurés : tableaux à plusieurs dimensions TD : types scalaires TP : partiel **CC2**
Semaine 7	Cours : correction du CC fichiers TD et TP : chaînes
Semaine 8	Cours : fichiers TD et TP : enregistrements
Semaine 9	Cours : piles/files/listes TD et TP : tableaux à plusieurs dimensions **CC3** papier – 30 mars
Semaine 10	TD et TP : révisions
Semaine 11	Cours : correction du CC retour sur les fonctions et procédures auto appelantes TD et TP : fichiers
Semaine 12	Cours : les unités et bibliothèques TD et TP : fichiers
Semaine 13	Cours : complexité et révisions TD : révisions TP : partiel **CC4**
17 mai	CC5 papier

Semaine n°1

Le but est ici de revoir les structures de données et de contrôle étudiées durant le premier semestre.

la structure d'un programme :

```
Programme nom_du_programme
Spécifications objet de la solution
Utilise si nécessaire
Déclarations constantes, types, sous-programmes, variables, …
Corps du programme
```

les entrées sorties

les itérations

- For

```
somme := 0 ;
for i := 1 to 10 do somme := somme + i ;
```

- Repeat … until (condition)

```
somme := 0 ; i := 1 ;
repeat
        somme := somme + i ;
        i := i + 1
until (i > 10) ;
```

- While (condition) do …

```
somme := 0 ; i := 1 ;
while (i<=10) do begin
                somme := somme + i ;
                i := i+1
                end ;
```

conditionnelles

- **if** (condition) **then** … **else** …
- **case** nom_de_variable **of**

```
valeur1 : instruction ;
valeur2 : instruction ;
…
valeurn : instruction
end ;
```

le type tableau

- **array [**1..7**] of** integer

un exemple

Ecrire un programme qui calcule la somme des premiers entiers, le nombre de ces entiers est un nombre inférieur à 50 qui doit être demandé à l'utilisateur, le programme affiche la somme.

solution 1 :

```
program somme1 ;
var i , N , S: integer ;
  begin
        writeln('Indiquer un entier') ;                                  sortie
        readln(N) ;                                                      entrée
        if (N > 50) then writeln( 'respectez les contraintes du programmes')   conditionnelle
                    else begin
                        S := 0 ;                                         initialisation
                        for i := 1 to N   do   S = S + i ;               boucle
                        writeln('La somme des ' , N , ' entiers est égale à ' , S)   sortie
                        end
  end .
```

solution 2 :

```
program somme2 ;
var i , N , S: integer ;
  begin
        writeln('Indiquer un entier') ;                                  sortie
        readln(N) ;                                                      entrée
        if (N > 50) then writeln( 'respectez les contraintes du programmes')   conditionnelle
                    else begin
                        S := 0 ;                                         initialisation
                        i := 1 ;                                         initialisation
                        repeat                                           boucle
                           S := S + i ;
                           i := i + 1
                        until (i > N) ;
                        writeln('La somme des ' , N , ' entiers est égale à ' , S)   sortie
                        end
  end .
```

solution 3 :

```
program somme3 ;
var i , N , S: integer ;
  begin
        writeln('Indiquer un entier') ;                                  sortie
        readln(N) ;                                                      entrée
        if (N > 50) then writeln( 'respectez les contraintes du programmes')   conditionnelle
                    else begin
                        S := 0 ;                                         initialisation
                        i := 1 ;                                         initialisation
                        while (i <= N) do   begin                        boucle
                                               S := S + i ;
                                               i := i + 1
                                            end ;
                        writeln('La somme des ' , N , ' entiers est égale à ' , S)   sortie
                        end
  end .
```

Travail à réaliser

Exercice 1

On souhaite réaliser un programme qui vérifie que l'utilisateur saisit au clavier un nombre pair compris entre 7 et 21.
Ecrire un programme **saisie** qui permette de demander la saisie d'un nombre, vérifie que ce nombre vérifie les contraintes imposées et affiche un diagnostic.
Le programme doit permettre de répéter la demande jusqu'à ce que le nombre 0 soit saisi par l'utilisateur.

exercice 2

Calculer l'expression $4 \cdot \sum_{i=0}^{N} \frac{(-1)^i}{2i+1}$ et afficher le résultat toutes les dix itérations.

Exercice 3

Ecrire, sans utiliser de sous-programme, un programme **tableau1** qui permette
1°/ de saisir les éléments d'un tableau de nombres. On pourra envisager deux approches.
a/ On saisit le nombre d'éléments du tableau a priori,
b/ On restreint les éléments du tableau à être des nombres positifs ou nuls et on arrête la saisie lorsqu'on trouve un nombre strictement négatif. Les nombres saisis seront affichés en fin de saisie.

2°/ d'afficher les valeurs contenues dans le tableau, augmentées de la valeur constante 3.

3°/ de conserver la première et la dernière valeur du tableau et de modifier les autres valeurs en remplaçant le terme d'indice j par la moyenne de trois termes, la valeur de l'élément d'indice j du tableau et les valeurs des termes immédiatement avant et après dans le tableau (j-1 et j+1). On affichera les nouvelles valeurs contenues dans le tableau. Les moyennes se calculent à partir des valeurs initiales du tableau.

4°/ de trier les éléments du tableau suivant les valeurs décroissantes, en échangeant éventuellement les contenus de deux éléments consécutifs jusqu'à stabilisation de la valeur de tous les éléments du tableau. Le résultat final sera affiché.

Semaine n°2

Le but est de maîtriser la notion de sous programme.
ici les **procédures**

Il s'agit d'organiser le code pour le rendre plus lisible et mieux réutilisable.
Il faut transmettre à une partie indépendante et réutilisable de code des variables et pouvoir disposer des résultats utilisables dans un programme principal.

Transmission des variables

Variables transmises "**par valeur**"
leur valeur ne peut pas changer pendant l'exécution du sous-programme

Variables transmises "**par référence**" ou "par adresse" → **var**leur valeur peut évoluer pendant l'exécution du sous-programme

remarque : les variables en donnée sont en général transmises par valeur, les variables en résultats sont toujours transmises par référence (**var**)

Les procédures

définition

```
procedure proce_maxi (X1, X2 : integer ; var Y : integer) ;
begin
        if (X1 > X2) then Y := X1 else Y := X2 ;
end ;
```

appel

```
proce_maxi(5,8,Z) ;
```

un exemple

Ecrire un sous-programme qui permette de saisir les éléments d'un tableau d'entiers dont le nombre de termes est connu.

Remarques :
T et N ne sont pas des variables déclarées dans le programme principal.

par contre

t_tab est déclaré dans le programme principal
type t_tab : array[1..100] of integer ;

solution

```
procedure saisie(N : integer ; var T : t_tab) ;
var i : integer ;
begin
        for i := 1 to N do begin
                                write('indiquer le terme d'indice ' , i , ' : ') ;
                                readln(T[i])
                        end
end ;
```

Travail à réaliser

Exercice 1

Ecrire un programme **histo** qui permet d'afficher des histogrammes de valeurs associés à des tableaux. On utilisera 3 procédures
saisie, qui permet de choisir la taille d'un tableau, de le saisir des éléments entiers de ce tableau, compris entre 0 et 20,
occur qui permet de stocker dans un autre tableau d'indices variant de 0 à 20 les nombres d'occurrences des valeurs de 0 à 20 dans un tableau du type précédent,
affich_histo qui permet d'afficher sur 21 colonnes consécutives les X correspondant au nombre d'occurrences. On utilisera deux boucles imbriquées l'une dans l'autre.

Exercice 2

On se propose de calculer les caractéristiques statistiques d'un ensemble de valeurs entières et positives. Ecrire un programme qui après avoir saisi un tableau d'entiers peut afficher

le nombre de valeurs saisies,
la valeur la plus petite, la valeur la plus grande,
la moyenne $\bar{x}$ $\quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$,
l'écart type σ $\quad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2 - \bar{x}^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2 \quad - \bar{x}^2}$.

On utilisera des procédures pour assurer la saisie des données qui seront stockées dans un tableau. Par cette procédure on obtiendra les données, le nombre de valeurs saisies, la plus petite valeur et la plus grande.
La seconde procédure permet de calculer la moyenne et l'écart type.
Remarque : ici, la saisie des nombres dans le tableau s'arrête quand la valeur saisie est négative.

Semaine n°3

Le but est de maîtriser la notion de sous programme.
ici les **fonctions**

C'est une forme particulière de sous programme où un résultat est privilégié, il est calculé dans le corps de la fonction.

Les fonctions

définition

```
function fct_maxi (X1, X2 : integer) : integer ;
Var Y: integer ;
begin
    if (X1 > X2) then Y := X1 else Y:= X2 ;
    fct_maxi := Y ;
end ;
```

appel

```
Z := fct_maxi(5,8) ;
```

un exemple

Ecrire un sous-programme qui permette de calculer le maximum des éléments d'un tableau d'entiers dont le nombre de termes est connu.

Remarques :
T et N ne sont pas des variables déclarées dans le programme principal.

par contre

t_tab est déclaré dans le programme principal
type t_tab : array [1..100] of integer ;

solution

```
function plusgrand(N : integer ; var T : t_tab) : integer;
var i , sup : integer ;
begin
        sup := T[1] ;
        for i := 2 to N do if T[i] > sup then sup := T[i] ;
        plusgrand := sup
end ;
```

Travail à réaliser

Exercice 1

Reprendre le troisième exercice de la semaine 1 et restructurer le programme en faisant apparaître des sous programmes

- **saisie** pour la saisie d'un tableau :
- **affiche** pour l'affichage d'un tableau
- **moyenne** pour le calcul de moyenne (moyenne de 3 éléments consécutifs dans le tableau)
- **échange** pour échanger les positions de deux éléments d'un tableau
- **tri** pour le rangement des éléments du tableau dans l'ordre décroissant
- **moyennage** pour le calcul expliqué semaine 1 exercice 2 3°/.

Conserver en mémoire ces éléments qui pourront être utilisés par la suite dans d'autres circonstances.

Exercice 2

Introduction des sous-programmes liés aux nombres aléatoires : RANDOM et RANDOMIZE
Ecrire un programme qui :

- crée un entier aléatoire A compris entre 0 et 1000
- permet de trouver ce nombre A en moins de 10 essais successifs, en indiquant si le nombre est plus grand ou plus petit que le nombre à trouver A
- permet de rejouer autant de fois qu'on veut.

Exercice 3

On considère la suite numérique définie par la formule de récurrence.

$$\begin{cases} U_{n+1} = \begin{vmatrix} 3U_n - 2U_{n-1} + 2^n & si\ n\ est\ pair \\ 3U_{n-1} - 2U_n + 3^n & si\ n\ est\ impair \end{vmatrix} \\ U_1 = \alpha \qquad U_0 = \beta \end{cases}$$

pour n strictement positif

1°/ Ecrire une fonction **calcul** dont l'en-tête est le suivant

calcul(a,b,c : integer) : integer ;

et dont le résultat est égal à 3a-2b+c

2°/ Ecrire une procédure **formule** qui permette le même calcul à partir des mêmes données.

3°/ Ecrire une fonction **puissance** qui pour une valeur donnée et un entier indiquant l'exposant calcule la valeur de la puissance.

4°/ Ecrire un programme **suite1** qui permette de lire trois entiers, d'une part les deux premiers termes α et β de la suite, d'autre part l'indice n du terme que l'on veut calculer, et qui permette d'afficher la valeur de U_n. Ce programme doit utiliser la fonction **calcul**.

5°/ Ecrire un second programme **suite2** qui ait la même fonctionnalité mais qui utilise la procédure **formule**.

.

Semaine n°4

Le but est de consolider les connaissances sur les sous-programmes.

Travail à réaliser

Exercice 1
Indiquer l'affichage que l'on peut prévoir correspondant au programme suivant :

```
program dirige2 ;
    procedure pro1(k:integer) ;
        begin
            writeln('k = ' , k) ;
            k := k + 1 ;
            writeln('k = ' , k)
        end ;
    procedure pro2(var k:integer) ;
        begin
            writeln('k = ' , k) ;
            k := k + 1 ;
            writeln('k = ' , k)
        end ;
    function fon1(k:integer) : integer ;
        begin
            writeln('k = ' , k) ;
            fon1 := k + 1 ;
            writeln('k = ' , k)
        end ;
    function fon2(var k:integer) : integer ;
        begin
            writeln('k = ' , k) ;
            fon2 := k + 1 ;
            writeln('k = ' , k)
        end ;
    var i : integer ;
    begin
        i := 1 ;
        writeln('i = ' , i) ;
        pro1(i) ;
        writeln('i = ' , i) ;
        pro2(i) ;
        writeln('i = ' , i) ;
        writeln(fon1(i)) ;
        writeln('i = ' , i) ;
        writeln(fon2(i))
        writeln('i = ' , i) ;
    end.
```

Exercice 2

1°/ Ecrire un sous programme **test** qui, à partir d'un tableau d'entiers, vérifie que les valeurs contenues dans le tableau sont comprises entre 1 et 39, le résultat est un booléen.
2°/ Ecrire un sous programme **dessin** qui, à partir d'un tableau d'entiers compris entre 1 et 39, affiche à l'écran des lignes de caractères X.

Sur chaque ligne, les lignes étant ordonnées comme les indices du tableau, le nombre de caractères X est égal à la valeur contenue dans le tableau.
Exemple : { 5 ,2 , 4 } s'affiche en

```
XXXXX
XX
XXXX
```

3°/ Ecrire un sous programme **dessindouble** qui fait un affichage symétrique du dessin.
Exemple :

```
XXXXXXXXXX
   XXXX
  XXXXXXXX
```

4°/ Ecrire une fonction booléenne qui indique si la variance (voir S2E3) des éléments du tableau est inférieure à une valeur donnée en argument de la fonction.

5°/ Ecrire un sous programme **lissage** qui applique le sous programme **moyennage** (S3E1) autant de fois que nécessaire pour que la variance des éléments du tableau soit inférieure à une valeur donnée.
On affichera les évolutions successives du dessin.

6°/ Ecrire un programme, en utilisant les sous programmes déjà écrits qui permette de remplir un tableau et le seuil de lissage, d'afficher les dessins correspondants (dessin et double dessin), de lisser l'allure du dessin.

7°/ Dans le sous programme **moyennage** décrit (S3E1), le moyennage est réalisé sur trois termes consécutifs du tableau. On pourra modifier la programmation pour que le nombre de termes puisse être modifié.

Semaine n°5

Le but est d'introduire de nouveaux types de données.

type intervalle

```
T_ind = 1..500 ;
T_note = 0 .. 20  ;
T_tab = array [1 .. 500] of 0 .. 20 ;
ou mieux
T_tableau = array [T_ind] of T_note ;
```

type énuméré

suite ordonnée de noms de constantes ;
une variable de ce type ne peut pas être affichée, ni lue

```
T_semaine = ( lundi , mardi , mercredi , jeudi , vendredi , samedi , dimanche ) ;
```

succ(x) **pred**(x)
attention, dans l'exemple précédent **succ**(dimanche) et **pred**(lundi) n'existent pas.
avec les varaibles first et last ayant pour valeur lundi et dimanche

type ensemble

définition

```
Type   T_ensemble = set of integer ;
Var    E : T_ensemble ;
E := [ ] ;
E := E + [7] ;
```

opérateurs

Union	+
Soustraction	-
Intersection	*
Appartenance	in

un exemple

Travail à réaliser

Exercice 1

1°/ Déclarer un type énuméré nommé **t_semaine** comportant 7 termes correspondant aux 7 jours de la semaine
et une variable de ce type nommée **jour**

Déclarer un type de tableau d'entiers dont les indices sont les jours de la semaine, il porte le nom **t_tab**
et une variable de ce type de nom **compte**

Ecrire un sous-programme **saisie** qui permette de saisir au clavier les différentes valeurs d'un tableau tel que celui déclaré dans la question précédente.

Ecrire un programme utilisant le sous programme et les déclarations précédents, qui permette de saisir les éléments d'un tableau indicé par les jours de la semaine, et qui calcule et affiche la somme des nombres contenus dans les cases du tableau qui correspondent aux jours ouvrés (5 jours consécutifs du lundi au vendredi inclus).

Exercice 2

On dispose d'un tableau contenant la liste des numéros d'inscription (chaîne de caractères) des étudiants d'un groupe de taille N (N<30), d'un tableau contenant les notes de mathématiques, et d'un tableau contenant les notes d'informatique.

1°/ Créer un ensemble figurant ceux qui ont eu la moyenne en mathématiques.

2°/ Créer un ensemble figurant ceux qui ont eu la moyenne en informatique.

3°/ A partir des deux ensembles précédents, créer la liste de ceux qui ont la moyenne en mathématiques et en informatique, de ceux qui ont la moyenne en mathématiques et pas en informatique, de ceux qui n'ont pas la moyenne en mathématiques et la moyenne en informatique.

Semaine n°6

Le but est de maîtriser le type chaîne et révision sur les tableaux.

type prédéfini : string
var nom : string[20] ;

les composants élémentaires sont des caractères

accès à un élément et affectation de valeur nom[4]

relations : = > < <> >= <=

sous programmes standards

longueur	N := length(nom) ;
concaténation	identite := concat(nom , ' ' , prenom) ;
position	n := pos('ba' , 'remballage') ;
extraction de chaîne de longueur L à la position P	result := copy(chaine , P , L) ;
suppression de L caractères	delete (chaine , P , L) ;
insertion d'un motif à une position P	insert(motif , chaine , P) ;

Travail à réaliser

Exercice 1

Ecrire un programme qui, dans une chaîne saisie au clavier, remplace les occurrences de la sous-chaîne 'ce jour' par la date du jour écrite de manière littérale. Le résultat de la transformation sera affiché à l'écran, la date du jour doit être saisie au clavier.

Exercice 2

Ecrire un programme qui permet de transformer une chaîne contenant du texte littéral et des nombres écrits en chiffres, en texte littéral écrit uniquement en majuscules.
Pour cela, on écrira des sous programmes qui permettent (tous les sous programmes ne sont pas nécessaires à la résolution du problème énoncé).

- de vérifier qu'une chaîne de caractères saisie au clavier par l'utilisateur contient uniquement des chiffres.
- dans le cas d'une chaîne ne contenant que des chiffres, de calculer la valeur du nombre décimal correspondant.
- de transformer une chaîne pour que les lettres qui apparaissent soient des majuscules.
- si la chaîne de caractères ne contient pas uniquement des chiffres, d'extraire chaque sous-suite de chiffres formant un entier. (une chaîne de chiffres entourée du caractère espace)
- de transformer une valeur entière en chaîne de caractères ; on se limitera aux nombres inférieurs à vingt.

On peut alors écrire le programme principal.

Exercice 3

En utilisant un sous-programme de saisie d'un tableau déjà réalisé dans un TD et TP précédents, écrire un programme qui permet d'utiliser des sous programmes que vous écrirez aujourd'hui.
1°/ Connaissant un tableau et une valeur, écrire une fonction dont le résultat est un booléen qui indique si la valeur est contenue dans le tableau.
2°/ A partir de deux tableaux T1 et T2 contenant respectivement N1 et N2 éléments écrire des sous programmes qui permette de :

- construire un tableau contenant les N1+N2 éléments sans condition sur leurs valeurs,
- construire un tableau qui contient les éléments de T1 non présents dans T2,
- construire un tableau qui contient les éléments communs à T1 et T2

Semaine n°7

Le but est de maîtriser les enregistrements.

déclaration

```
TYPE t_membre = RECORD
                    nom : string[20] ;
                    ident : string[6] ;
                    licence : boolean ;
                    annee : integer
                    end ;
```

var membre1 : t_membre ;

accès au nom : membre1.nom

opérateurs

```
globalement
    égalité        =
    inégalité      <>
sur les champs
    ceux qui sont valides sur chacun des champs élémentaires
```

instruction with

```
with membre1 do
                 begin
                annee := 1 ;
                ident := 'A12345'
                end ;
```

Travail à réaliser

Exercice

Un garagiste se propose de gérer sa clientèle et aussi l'activité de son entreprise à l'aide d'un programme informatique. Pour chaque client sont connus son adresse, le nombre et le type des voitures qu'il possède, et si celles-ci sont neuves ou d'occasion, la date d'achat pour chacune. Le garage limite à 200 le nombre de clients et à 50 le nombre de modèles de voitures.

1°/ Il est dans un premier temps nécessaire de créer des types adaptés à la nature des données, l'un pour le client (on utilisera des tableaux pour le type de voiture, son prix, sa nature) et l'autre pour la voiture (couleur, puissance, nombre de portes, carburant, prix, occasion). Les données étant alors stockées dans deux tableaux.

Ecrire un programme permettant :

2°/ d'entrer les données : nombre effectif de clients et de voitures à vendre, ajout de clients et de voitures.

3°/ de connaître, pour un type de voiture donné, le nombre et la liste des clients possédant une voiture de ce type.

4°/ de modifier la fiche d'un client en proposant de modifier successivement les différents champs.

5°/ de savoir combien de voitures correspondant aux 5 critères sont disponibles et d'afficher leurs noms et leurs prix respectifs.

6°/ de calculer le nombre de voitures de chaque type vendues auprès de la clientèle, et d'afficher l'histogramme correspondant. On affichera un type de voiture par ligne, et sur la ligne un nombre de 'X' égal au nombre de voitures vendues une année donnée.

C1
XXXXXXXX
C2
XXXXX
C3
XXXXXXXXX
C4
XXX

Semaine n°8

Le but est de maîtriser les tableaux à deux dimensions.

déclaration

```
Const   lig = 10 ;
        col = 10 ;

TYPE    t_lig = 1..lig ;
        t_col = 1..col ;
        t_tab2 = array [t_lig , t_col] of integer ;

Var     A : t_tab2 ;
```

accès aux éléments
A[3,2]

A[1,1]	A[1,2]	A[1,3]		A[1,c]
A[2,1]	A[2,2]			
A[3,1]				
A[li,1]				A[li,c]

exemple

```
Procedure affiche (matri : t_tab2 ; nlig : t_lig , ncol : t_col) ;
Var i , j : t_ind ;
Begin
  for i:= 1 to nlig do
            begin
              for j := 1 to ncol do write (matri[i , j] : 4) ;
              writeln
            end
End ;
```

Travail à réaliser

exercice

Un carré magique de taille N est un arrangement en carré des nombres 1,2, …, N^2 tel que si l'on effectue la somme des éléments d'une ligne, d'une colonne ou de l'une des deux diagonales, on obtienne toujours la même valeur. Le dessin suivant représente un carré magique de taille 5 :

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

1°/ Formuler, par rapport à N, la valeur constante des sommes des lignes, colonnes et diagonales.

2°/ Ecrire une fonction Pascal qui détermine si un carré est magique.

3°/ Ecrire une procédure qui construit un carré magique de taille N, N impair, en plaçant les valeurs 1, 2, …, N^2 suivant le principe suivant :

on place la valeur 1 au milieu de la ligne 1,

puis on continue en montant en diagonale vers la gauche ;

- si cela conduit à déborder en haut ou à gauche, le nombre est placé dans la dernière ligne ou la dernière colonne.
  Par exemple, 2 est placé dans la dernière ligne, et 23 est placé dans la dernière colonne.
- Si on atteint une case déjà remplie, le nombre est placé en dessous du nombre précédent ; cette dernière situation se produit chaque fois qu'on vient de placer un multiple de N.
  Par exemple, 6 est placé sous 5 et 11 est placé sous 10.

4°/ Obtenir un nouveau carré magique par rotation du précédent.

Semaine n°9

Le but est d'assurer les connaissances acquises.

Travail à réaliser

Exercice

Un échiquier est un carré divisé en 64 cases alternativement blanches ou noires, chacune représentée par ses coordonnées (i,j). La case en haut à gauche a pour coordonnées (1,1).
Les cases sont numérotées de 1 à 64. La case de coordonnées (1,1) a le numéro 1, elle est blanche, la case (1,2) a le numéro 2, elle est noire, la case (2,1) a le numéro 9, elle est noire …
Pour effectuer un déplacement, partant d'une case (i,j), un cavalier a le choix entre les 8 cases (i+2,j+1),(i+2,j-1) ,(i+1,j+2),(i+1,j-2) ,(i-1,j+2),(i-1,j-2),(i-2,j-1),(i-2,j+1) si ces cases existent.
Aux coordonnées possibles des cases est associé un type intervalle **t_case**, aux numéros possibles des cases est associé un type intervalle **t_numero** et aux couleurs est associé un type énuméré **t_couleur**.

1°/ Ecrire un sous-programme pour transformer les coordonnées d'une case en son numéro.

2°/ Ecrire un sous programme pour transformer le numéro d'une case en ses coordonnées.

3°/ Ecrire une procédure **deplace** qui, pour 2 coordonnées **li** et **co** ainsi que pour **k**, l'un des 8 déplacements possibles, dit si le déplacement **k** est envisageable et si oui, donne le numéro de la case correspondante. L'ordre des 8 déplacements est choisi arbitrairement, par exemple celui donné plus haut dans l'énoncé.

4°/ Pour chaque case (de 1 à 64), on va rechercher les déplacements possibles du cavalier ainsi que le nombre de ces déplacements possibles. Ces informations seront rangées dans un tableau d'enregistrements. Chaque enregistrement contiendra les champs :

- **tab** est un tableau de longueur 8 contenant les numéros des cases correspondant à un déplacement possible
- **nb** est un entier donnant le nombre de déplacements possibles
- **color** indique la couleur de la case de départ.

Ecrire une procédure qui remplit le tableau précédent.

5°/ Ecrire un programme qui permet de vérifier la bonne écriture de ces sous-programmes.

Semaine n°10

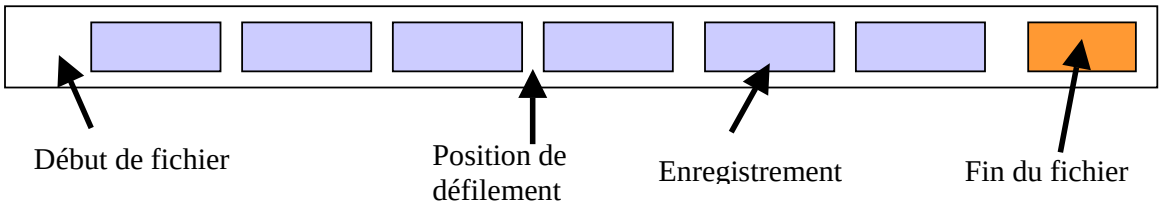
Le but est de maîtriser les fichiers.

fichier à accès séquentiel différent de fichier à accès direct
distinguer nom interne du fichier et nom physique sur le support
action :

- **Créer un fichier vide ou ouvrir un fichier existant**
- **Ajouter de nouveaux éléments en fin de fichier**

ou bien

- **Ouvrir un fichier existant**
- **Lire séquentiellement les éléments du fichier dans l'ordre**



déclaration
pour le nom interne du fichier, un type prédéfini :

file of ou bien **TEXT** pour les fichiers de lignes de caractères

déclaration du nom interne : file of *nom_de_type* ou bien **Var F : file of integer ; ou bien Var F : TEXT**

actions

lecture du premier élément stocké dans la variable art

```
Var     F : file of t_membre ;
        art : t_membre ;
```

- **Assign**(F , 'c:fichier.dat') ; **Rewrite**(F) ; ou bien **Rewrite**(F, 'fichier') ;
- **Reset**(F) ;
- Read(F , art) ;
- if (**Eof**(F)) then writeln('la fin du fichier est atteinte') ;
- **close**(F) ;

création et écriture d'une chaîne dans un fichier texte

```
Var     F : text ;
        S : string ;

  rewrite(F, 'essai.dat') ;
  writeln ('saisir du texte') ;
  readln(S) ;
  writeln(F,S) ;
  close(F)
```

lecture séquentielle d'un fichier texte

```
Var     F : text ;
        S : string ;

  reset(F, 'essai.dat') ;
  repeat
    readln(F,S) ;
    writeln(S) ;
  until ( eof(F)) ;
  close(F)
```

Travail à réaliser

Exercice

Image1.img est un fichier text qui comporte une suite de lignes toutes de même longueur. De plus les caractères utilisés sont des '1' et des '0' uniquement.

1°/ Ecrire un sous programme **lire** qui permet de transférer le contenu d'un tel fichier dans un tableau d'entiers de type **t_image** à définir. On n'oubliera pas de gérer la taille de l'image.

2°/ Ecrire un sous-programme **affiche** pour afficher une telle image à l'écran. Les '1' correspondront à l'écriture d'un 'X' et les '0' seront traduits par un espace.

3°/ Ecrire un sous-programme **ecrire** pour stocker dans un fichier dont le nom sera indiqué, le contenu d'un tableau de type t_image.

On se propose de faire quelques transformations sur une image. Ces transformations sont de type local. Elles dépendent de l'état du voisinage de chaque point de l'image. Les transformations s'effectuent à partir d'un tableau de type t_image. On s'aidera d'un autre tableau auxiliaire.

4°/ Ecrire un sous programme **bruit** qui supprime les points isolés, c'est-à-dire qui remplace un '1' par un '0' quand le '1' est entouré de 8 '0' et qui remplace un '0' par un '1' quand le '0' est entouré de 8 '1'.

5°/ Ecrire un sous programme **ouverture** qui conserve dans l'image tous les groupes de 4 '1' en carré. On utilisera un autre tableau pour stocker le résultat.

6°/ Ecrire un sous programme **érosion** qui conserve à '1' tous les points dont les 8 voisins sont '1' et qui met à '0' les autres points.

7°/ Ecrire un sous programme **dilatation** qui remplace chaque '0' par un '1' dès qu'un de ses 8 voisins est à '1'.

8°/ Ecrire un programme qui permet de vérifier la bonne écriture de ces sous-programmes et de montrer leur intérêt sur l'image proposée.

exemple de fichier image1.txt

```
00000000000000000000000000000000000000000000
00100000111111000100001000000111111000000
00000011111111100000100000111111011110000
00001111000111000000000001111111111111100
00011100000111000100001111110000000001110
00111000000111000000011110000000000001010
01110000000111100000111100000000000011110
01100000000111000010011110000010000111100
01000100000111000000001000000000001111000
00000000000111000000000000000000000111100000
00000000000111100000000100000000111100000
00100010001111000000000000000011100000000
00000000000111000010000000001111000010000
00000000000111000000000000111100000000000
00010000000111000000000001111000010000110
00000000001101100000011111111111111111110
00000000001111100001111111111111111111110
00000000000000000000000000000000000000000000
```

Semaine n°11

Le but est de maîtriser la notion de fichier.

Travail à réaliser

Exercice 1

Dans un programme contenant un menu écrire des sous-programmes qui permettent :

1°/ de créer un fichier texte contenant des noms écrits chacun sur une ligne

2°/ d'ajouter un nom en fin de fichier

3°/ de lire les mots contenus dans un fichier texte et de les classer dans l'ordre alphabétique

4°/ de créer un fichier où les mots sont classés par ordre alphabétique

5°/ de rechercher si 3 fichiers de telle structure, classés ou non dans l'ordre alphabétique, contiennent un élément commun. Ce dernier sous-programme doit rendre un booléen et le nom commun s'il existe.

Exercice 2

Pour gérer une bibliothèque on crée un fichier de lecteurs. Pour chaque lecteur on note son numéro, son nom (sur 20 caractères), le nombre de livres empruntés, la liste des identifiants (entier) des livres empruntés.

Ecrire des sous-programmes qui permettent de

1°/ créer un fichier

2°/ vérifier si un lecteur est déjà inscrit dans la bibliothèque grâce à son unméro

3°/ remplir une nouvelle fiche

4°/ supprimer une fiche de lecteur dont on connaît le numéro

5°/ mettre à jour une fiche

6°/ compter le nombre de lecteurs ayant emprunté un livre

7°/ trouver le lecteur ayant emprunté un livre d'identifiant donné

8°/ un menu correspondant à une utilisation dans le cadre d'une bibliothèque.

Semaine n°12

Le but est de réviser en fonction de la demande des étudiants.

ANNALES

2005 - 2006

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2005-2006
21 février 2006

Programmation 2
Contrôle continu 1

aucun document autorisé

1. On considère les instructions suivantes :

```
x :=0 ; y := 0 ; aux := true ;
While (x < 5) do
   begin
     x := x + 1 ; y := y + 1 ;
     if (y mod 2 = 0 and aux)
         then begin y := y + 2 ; aux := not(aux) ; end
   end ;
writeln (x , '          ' , y) ;
if aux then writeln(x) else writeln(y) ;
```

   Quel est l'affichage ?

2. Transformer la structure itérative précédente en utilisant une "boucle repeat"

3.
```
program principal ;
const N = 3 ;
var A,B,C,N : integer ;
procedure exemple(X1,X2 : integer;var Y:integer);
var P,I : integer ;
     begin     P := 1 ;
          for I := 1 to N do P := P * X1 ;
          X1 := P ; X2 := 3*X2 ; Y := X1+X2 ;
          writeln(X1,X2,Y) ;
     end ;
begin
     A := 2 ; B := 3 ; C := 0 ;
     exemple(A,B,C) ; writeln(A,B,C)
end.
```

   Quel est l'affichage ?

4. Ecrire une procédure nommée **echange** qui permette d'échanger les contenus de deux variables entières. On veillera à écrire correctement les paramètres.

5.1 Déclarer un type énuméré nommé **t_semaine** comportant 7 termes correspondant aux 7 jours de la semaine
et une variable de ce type nommée **jour**

5.2 Déclarer un type de tableau d'entiers dont les indices sont les jours de la semaine, il porte le nom **t_tab**
et une variable de ce type de nom **compte**

Tournez S.V.P.

5.3. Ecrire un sous-programme **saisie** qui permette de saisir au clavier les différentes valeurs d'un tableau tel que celui déclaré dans la question précédente.

.

5.4 Ecrire un programme utilisant le sous programme et les déclarations précédents, qui permette de saisir les éléments d'un tableau indicé par les jours de la semaine, et qui calcule et affiche la somme des nombres contenus dans les cases du tableau qui correspondent aux jours ouvrés (5 jours consécutifs du lundi au vendredi inclus).

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2005-2006
21 mars 2006

Programmation 2
Contrôle continu 2

aucun document autorisé

1. Ecrire ce qui apparaît sur l'écran lorsqu'on exécute le programme suivant :

```
Program opaque ;
Var a , b , c : integer ;
    Logic : boolean ;
Procedure P1(x,y:integer ; var z:integer ; test:boolean) ;
Begin
    If test then
       begin if x<y then z:=x else z:=y end
       else
       begin if x<y then z:=y else z:=x end ;
     test := not(test)
end ;
Procedure P2(x,y:integer ; var z:integer ; var test:boolean) ;
Begin
    Test := not(test) ;
     if x<y then z:=x else z:=y
end ;
begin
     a:=0 ; b:=5 ; c:=10 ; logic:=false ;
     P1(a,b,c,logic) ; writeln(a,'  ',b,'  ',c) ;
     P2(a,b,c,logic) ; writeln(a,'  ',b,'  ',c) ;
     P1(a,b,c,logic) ; writeln(a,'  ',b,'  ',c) ;
     P2(a,b,c,logic) ; writeln(a,'  ',b,'  ',c) ;
     If logic then writeln('a=' , a) else writeln('b=' , b)
end.
```

2. On veut exploiter une petite forêt (de moins de 10 000 arbres). Les arbres de cette forêt sont un ensemble de sapins, d'épicéas, de chênes, de hêtres, de frênes, d'érables, de châtaigniers. Les noms des arbres sont, dans cet ordre, les éléments d'un tableau **nom** de type **T_t0** à une dimension qui contient les chaînes de caractères correspondantes.
   Les renseignements nécessaires à l'exploitation seront stockés dans quatre tableaux.
   Le tableau de totalisation **total**, de type **T_t1**, à 2 dimensions, dans lequel l'indice de ligne varie de 1 à 7, 7 étant le nombre d'espèces des arbres. L'indice de colonne varie de 1 à 4. Pour chaque espèce, on note le nombre d'arbres, la circonférence moyenne, la hauteur moyenne et le cubage (volume en m^3) moyen.
   Le tableau d'identification **identification** de type **T_t2**, tableau à 2 dimensions. Le premier indice varie de 1 à 10 000 et le second de 1 à 2. Le tableau contient, pour chaque arbre, son numéro d'identification et son espèce, sous forme d'une chaîne de caractères. Le tableau est initialisé par des chaînes vides.

   Le tableau des données **don** de type **T_t3**, tableau à 2 dimensions. Le premier indice varie de

1 à 10 000 et le second de 1 à 3. Le tableau contient la circonférence, la hauteur et le cubage de chaque arbre. Le tableau est initialisé par des réels nuls.
Le tableau d'abattage **abat** de type **T_t4**, tableau à 1 dimension, de booléens, indique si un arbre doit ou non être abattu (vrai indique un arbre à abattre; faux indique un arbre à conserver). L'indice varie de 1 à 10 000.
On admettra que la procédure **init** permet d'initialiser et de remplir ces 5 tableaux. Les paramètres de cette procédure sont donc au nombre de 5, de types respectifs T_t0, T_t1, T_t2, T_t3 et T_t4. La procédure initialise **abat** à faux.

2.1 Ecrire les déclarations de type correspondant aux 5 types introduits.

2.2 Ecrire un sous-programme qui affiche le nombre de résineux (sapins ou épicéas) et le nombre de feuillus (non résineux) de la forêt.

2.3 Le calcul du cubage C d'un arbre se fait comme suit : $C = \left[\frac{circonference}{4}\right]^2 * hauteur$

Ecrire un sous-programme **cube** qui, pour une circonférence et une hauteur données, calcule le cubage.

2.4 Ecrire un sous-programme qui a pour résultat le tableau de type T_t4 où on écrit que les arbres de cubage supérieur à une valeur C0 sont à abattre.

2.5 Après une campagne d'abattage,
Les tableaux doivent être remis à jour. Les arbres à abattre sont supposés abattus,
Leurs éléments d'identification doivent devenir des chaînes vides,
Les circonférences, les hauteurs, et les cubages sont mis à 0
Le contenu du tableau de totalisation est recalculé.
Ecrire un sous-programme correspondant à ces spécifications.

2.6 Ecrire un sous-programme qui permette de supprimer les lignes vides dans les différents tableaux. Les lignes pleines des 3 tableaux d'identification, de données et d'abattage seront regroupées au début des tableaux.

2.7 Ecrire la définition d'un type permettant de décrire un arbre comprenant son identification, la circonférence et la hauteur mesurées à différentes dates (au plus 10) et la date où il a été abattu.

Les questions sont indépendantes

---

nom

1	‘Sapin’
2	‘Epicéa’
3	‘Chêne’
7	‘Châtaignier’

total

	Nombre d'arbres 1	Circonférence moyenne 2	Hauteur moyenne 3	Cubage moyen 4
1	230	1.4	15.3	2.1
2	1150	1.2	13.5	1.5
3	2100	2.1	19.6	6
7	300	1.9	15.5	3.8

identifaction **don** **abat**

	élément d'identification 1	espèce 2	circonférence	hauteur	cubage	
1	‘sa127’	‘Sapin’	1.9	19.1	4.3	True
2	‘’	‘’	0	0	0	False
3	‘he1n32o’	‘Hêtre’	2.3	18.3	6.1	False
4	‘sa285’	‘Sapin’	0.8	5.1	0.2	False
	…	…	…			…
100	‘che12n9o’	‘Chêne’	5	24.6	38.4	True
	...	...	..			...
10 000	‘er4n22o’	‘Erable’	1.8	8.2	5.3	False

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2005-2006
10 mai 2006

Programmation 2
Contrôle continu 3

aucun document n'est autorisé

1. On admet que le développement suivant :

$$d_n = 2\left[\frac{x-1}{x+1} + \frac{1}{3}*\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}*\left(\frac{x-1}{x+1}\right)^5 + \cdots + \frac{1}{2n+1}*\left(\frac{x-1}{x+1}\right)^{2n+1}\right]$$

est une approximation de ln(x), l'erreur de méthode vérifiant :

$$\varepsilon_n = \left|\ln(x) - d_n\right| < \frac{1}{2n+3} * \left|\frac{x-1}{x+1}\right|^{2n+3} * \frac{(x+1)^2}{2x}$$

Ecrire un sous-programme permettant le calcul de la valeur approchée de ln(x), à partir des données suivantes : x un réel positif, epsilon un réel positif.
Les résultats seront la valeur approchée de ln(x), à epsilon près et le nombre de termes utilisés pour ce calcul.
Note : remarquer qu'il existe une relation simple entre deux termes consécutifs du développement, d_n et d_{n+1}.

$$d_{n+1} = d_n + \frac{2}{2n+3} * \left(\frac{x-1}{x+1}\right)^{2n+3}$$

2. On rappelle les sous-programmes sur les chaînes qui peuvent être utilisés dans l'exercice :
   Concat(chaine1 , chaine2 , … , chaineN) : string
   Positionde(chaine, sous_chaine) : integer
   RemplaceCar(chaine1 , i , c) : string {le i^e caractère est remplacé par c}
   extraireChaine(Chaine, i,n) : string {chaîne composée d'un nombre de caractères égal à n pris dans chaine à partir de la position i}
   Longueur(chaine) : integer

On dispose de deux fichiers texte. L'un contient un texte contenant des marques spéciales, l'autre un dictionnaire de mots. Le fichier dictionnaire contient un mot par ligne. On se propose de remplacer dans le texte les marques spéciales par les mots du dictionnaire. Chaque marque est de la forme : le signe $ suivi par un, deux ou trois chiffres, et d'un signe $ pour indiquer la fin de la marque. Chaque chiffre est un des caractères '0' , '1', '2' , '3' , '4' , '5' , '6' , '7' ; '8' , '9'. Le signe $ n'est utilisé que pour les marques
Par exemple, à partir du texte :

Allons voir un 2 de 3
Mais il faut 1
Attendre qui ?
Attendre que le 3 se couche
Antoine de Saint-Exupéry

Et d'un dictionnaire contenant les mots suivants :

Attendre
Coucher
Soleil

Nous souhaitons obtenir le texte :

Allons voir un coucher de soleil
Mais il faut attendre
Attendre qui ?
Attendre que le soleil se couche

Antoine de Saint-Exupéry

2.1 On demande d'écrire une fonction de conversion permettant le passage d'une marque (exemple 185) vers la forme numérique entière correspondante. Son en-tête est le suivant :

convertChaineVersEntier(ch : string) : integer ;

2.2 Réaliser un sous programme qui, à partir de la donnée de deux fichiers texte correspondant aux exemples 1 et 2, crée un troisième fichier correspondant à l'exemple 3 .

3. Tournez S.V.P.

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2005-2006
13 juin 2006

Programmation 2
Session 2

aucun document n'est autorisé

1. La suite définie par la formule de récurrence.

$$\begin{cases} \mathbf{U_{n+1}} = \sqrt{\mathbf{1} + (\mathbf{U_n})^2} \\ \mathbf{U_1} = \mathbf{a} \end{cases}$$ pour n strictement positif

1.1 Ecrire un sous-programme **iter** qui, à partir de la valeur d'un élément de la suite, permet de calculer la valeur de l'élément de la suite ayant l'indice strictement supérieur.

1.2 Ecrire une fonction **calcule** qui, à partir de la valeur d'un indice n, permet de calculer la valeur de l'élément U_n de la suite.

Rq. Aucune variable globale ne devra être utilisée.

1.3 Ecrire un programme qui demande à l'utilisateur la valeur d'un entier positif et affiche la valeur du premier élément de la suite supérieur à la valeur de cet entier s'il existe ou un message indiquant qu'il n'existe pas de tel élément dans les 500 premiers termes de la suite.

2. Les clients d'un magasin sont connus par leur nom, leur numéro d'inscription qui correspond à l'ordre d'inscription, le code postal de leur lieu d'habitation, la somme dépensée dans le magasin jusqu'à ce jour.

2.1 Ecrire le type enregistrement correspondant à ces données.

2.2 On suppose qu'un fichier appelé "client.dat" sur le disque "D" contient l'ensemble des clients.
Ecrire un sous-programme qui permet de lire un fichier de ce type et qui permet de mettre à jour le champ "somme dépensée". Cette donnée doit être un argument du sous programme, par contre le nom du client doit être lu dans le sous programme.

2.3 Ecrire un sous-programme qui calcule le pourcentage de clients qui ont dépensé une somme supérieure à une valeur donnée, celle-ci devra figurer comme argument du sous programme. Ce sous programme doit pouvoir être utilisé pour tout fichier ayant la même structure que le fichier "client.dat" mais un nom différent.

ANNALES

2006 - 2007

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2006-2007
13 mars 2007

Programmation 2
Contrôle continu 1

aucun document autorisé

1. On considère les instructions suivantes :

```
x :=0 ; y := 0 ; aux := true ;
repeat
     x := x + 2 ; y := y + 1 ;
     if (y mod 2 = 0 and aux)
         then begin y := y + 2 ; aux := not(aux) ; end
until (x > 5) ;
writeln (x , '          ' , y) ;
if aux then writeln(x) else writeln(y) ;
```

   Quel est l'affichage ?

2. Transformer le bloc précédent en utilisant une "boucle while"

3.
```
program principal ;
procedure exemple1(X1,X2,N : integer;var Y:integer);
var I : integer ; P : longint ;
     begin     P := 1 ;
          for I := 1 to N do P := P * X1 ;
          X1 := P ; X2 := 3*X2 ; Y := X1+X2 ;
          writeln(X1,X2,Y) ;
     end ;
procedure exemple2(N : integer;var Z1,Z2,Z:integer);
var aux : boolean ;
     begin     aux := true ;
          repeat
               N := N + Z1 ; Aux := not(aux) ;
               Z1 := N ; Z2 := 3*Z2 ; Z := Z1+Z2 ;
          until (aux or (N > 10)) ;
          writeln(Z1,Z2,Z) ;
     end ;

var A,B,C,N : integer ;
begin
     N := 3 ; A := 2 ; B := 3 ; C := 0 ;
     Exemple1(A,B,N,C) ; writeln(A,B,C,N) ;
     N := 1 ; A := 2 ; B := 3 ; C := 0 ;
     Exemple2(A,B,N,C) ; writeln(A,B,C,N)
end.
```

   Quel est l'affichage ?

Tournez S.V.P.

4. Déclarer un type énuméré nommé **t_ufr** comportant 5 termes correspondant aux 5 années de la scolarité

et une variable de ce type nommée **année**

5. Déclarer un type de tableau d'entiers dont les indices sont les années de scolarité, L1, L2, L3, M1, M2 ; il porte le nom **t_tab**
et une variable de ce type de nom **A2002**

6. Ecrire un sous-programme **saisie** qui permette de saisir au clavier les différentes valeurs d'un tableau tel que celui déclaré dans la question précédente et qui calcule la somme des éléments du tableau, c'est à dire l'effectif de l'ufr.

7 Ecrire un programme utilisant le sous programme et les déclarations précédents, qui permette de saisir les éléments de 6 tableaux indicés par les années de licence, **A2002**, **A2003**, **A2004**, **A2005**, **A2006**, **A2007**. Les effectifs de l'ufr seront stockés dans un tableau **archive** dont les indices varient de 2002 à 2007. Le programme affichera l'année de plus fort effectif global.

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2006-2007
3 avril 2007

Programmation 2
Contrôle continu 2

aucun document autorisé

Pour un client, on connaît, le nom, le prénom, le genre, la taille et la liste des montants de ses 10 derniers achats. Le nom et le prénom sont des chaînes de longueur maximum 25 caractères, le genre est un énuméré à 3 valeurs **M**, **Mme**, **Melle**, la taille est un entier et la liste est un tableau à deux dimensions qui indique le prix des achats en fonction de l'ordre des achats (1 correspond au plus ancien et 10 au plus récent) et de leur catégorie (elle est indiquée par un entier compris entre 1 et 20).
L'ensemble des clients est connu dans un tableau nommé **clients** et le nombre de clients présents dans le tableau est **N**. Pour chaque client le tableau des achats comporte au plus dix éléments non nuls. La fin du tableau est indiquée par des éléments vides ou nuls.

Il est rappelé qu'aucune variable globale ne devra être utilisée.

1. Ecrire un type **t_personne** qui permet de décrire un individu.

2. Dans un courrier on veut introduire une personnalisation qui dépend du nom, du prénom et du genre de la personne à laquelle est destiné le courrier.
   A partir d'une variable de type **t_personne** connue pour un individu, écrire une fonction **perso** qui construit une chaîne dont un exemple serait "Monsieur Olivier DUPONT".

3. Ecrire une fonction **remise** qui calcule la somme correspondant à 10% des achats déjà effectués par un individu. On fournira à la fonction le tableau des achats correspondants.

4. Ecrire une fonction **ligne_vide** qui analyse un tableau comme celui contenant les achats. La fonction a pour résultat le numéro de la première ligne vide du tableau si elle existe et 0 sinon.

5. Ecrire une procédure **achat** qui, étant donnés un individu, une catégorie d'achat, et un montant du nouvel achat, met à jour le tableau des achats de cette personne. Si le nombre d'achats est inférieur à 10 on ajoutera le nouvel achat, par contre, si le tableau contient déjà 10 achats, on veillera à ne faire figurer dans le tableau que les 10 derniers achats.

	1	2		19	20
1	0	54,3		0	0
2	0	0		0	10,2
9	0	0		0	0
10	0	0		0	0

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2006-2007
9 mai 2007

Programmation 2
Contrôle continu 3

aucun document autorisé

Exercice 1 Aucune variable globale ne devra être utilisée.
La suite définie par la formule de récurrence.

$$\begin{cases} U_{n+1} = 5\,U_n - 3 \\ U_1 = a \end{cases}$$ pour n strictement positif

2.1 Ecrire un sous-programme **iter** qui, connaissant la valeur d'un élément de la suite, permet de calculer la valeur de l'élément de cette suite ayant l'indice immédiatement supérieur.

2.2 Ecrire une fonction **calcule** qui, à partir de la valeur d'un indice n, et de a, la valeur du premier élément, permet de calculer la valeur de l'élément U_n de la suite.

Exercie 2

Un fichier texte nommé "liste.dat" est stocké sur le disque D. Il est constitué par une liste de noms, chaque nom figurant sur une ligne. L'objet de cet exercice est de créer un nouveau fichier "ordonne.dat" sur le disque D, ce nouveau fichier doit avoir la même structure mais les noms sont maintenant stockés par ordre alphabétique.

1.1 Ecrire un sous programme **lire** qui lit un ficher texte connu par son nom et qui met le contenu de ce fichier dans un tableau, chaque élément du tableau contient une chaîne.
On commencera par écrire la définition du type du tableau utilisé : **t_tab**. sachant qu'on ne peut considérer plus de 500 éléments dans le tableau.

1.2 Ecrire une fonction **recherche** qui recherche si un nom donné est contenu dans un tableau du type précédent, ce tableau sera passé en paramètre de la fonction. Le résultat de la fonction est un booléen vrai si le nom a pu être trouvé et faux s'il n'a pas pu être trouvé.

1.3 Ecrire une procédure **classe** qui à partir d'un tableau de type t_tab crée un tableau de même type mais qui ne contient aucun doublon (les noms ne sont écrits qu'une seule fois).

1.4 Ecrire un programme **CC3** qui permet de lire un fichier, de tester autant de fois qu'on veut la présence d'un nom, de trier le fichier, et de stocker dans un fichier.

Université René Descartes
UFR Math-Info
Licence 1 – S2

année 2006-2007
juin 2007

Programmation 2
Session 2

aucun document autorisé

Exercie 1 Aucune variable globale ne devra être utilisée.

On considère la suite numérique définie par la formule de récurrence.

$$\begin{cases} U_{n+1} = \begin{vmatrix} 3U_n - 2U_{n-1} + 2^n & si\ n\ est\ pair \\ 3U_{n-1} - 2U_n + 3^n & si\ n\ est\ impair \end{vmatrix} \\ U_1 = a \end{cases}$$ pour n strictement positif

1.1 Ecrire une fonction **calcul** dont l'en-tête est le suivant
calcul(a,b,c : integer) : integer ;
et dont le résultat est égal à 3a-2b+c.

1.2 Ecrire une procédure **formule** qui permette le même calcul à partir des mêmes données.

1.3 Ecrire une fonction **puissance** qui, pour une valeur donnée et un entier indiquant l'exposant, calcule la valeur de la puissance.

1.4 Ecrire un programme **suite1** qui permette de lire deux entiers, d'une part le premier terme a de la suite, d'autre part l'indice n du terme que l'on veut calculer, et qui permette d'afficher la valeur de U_n. Ce programme doit utiliser la fonction **calcul**.

1.5 Ecrire un second programme **suite2** qui ait la même fonctionnalité mais qui utilise la procédure **formule**.

Tournez S.V.P.

Exercice 2 Aucune variable globale ne devra être utilisée.

un type **t_mat** est défini par :
t_mat = (info , algo , alge , analyse , phys , angl , eco) ;
On considère un enregistrement qui comprend deux champs : un tableau à deux dimensions et le nombre de lignes du tableau utilisées. Les indices de ligne du tableau sont des entiers et l'indice de colonne est de type t_mat. Un tableau **promo** de ce type contient sur chaque ligne les notes obtenues par un étudiant dans chaque matière, toutes les matières sont obligatoires.

2.1 Ecrire la définition du type **t_donnee** correspondant à l'enregistrement décrit précédemment.

2.2 Ecrire un sous programme qui permet à partir du nom d'une matière, d'afficher d'une part le nombre d'étudiants ayant la moyenne dans cette matière et d'autre part la moyenne générale obtenue dans cette matière par l'ensemble de la promotion.

2.3 Ecrire un sous programme qui permet de déterminer l'indice de l'étudiant classé premier de la promo, c'est-à-dire celui qui a la meilleure moyenne, toutes les matières étant affectées du coefficient 1.

ANNALES

2007 - 2008

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2007-2008
18 mars 2008

Programmation 2
Contrôle continu 1

aucun document autorisé

1.1 On considère la fonction suivante :

```
function exemple(N : integer;P : integer) : integer;
var aux : integer ;
     begin     aux := 1 ; C := 1 ;
          repeat
               aux := aux * N ; C := C + 1 ;
          until C > P ;
          exemple := aux ;
     end ;
```

Transformer ce sous-programme en procédure.

1.2 On considère le programme suivant :

```
program principal ;
const L := 10 ;
type t_TAB = ARRAY[1..15] of integer ;
function exemple(N : integer;P : integer) : integer;
var aux : integer ;
     begin     aux := 1 ; C := 1 ;
          repeat
               aux := aux * N ; C := C + 1 ;
          until (C > P) ;
          exemple := aux ;
     end ;

var I, X : integer ;
    T : t_TAB ;
begin
     X := 825 ;
     for I := 1 to L do T[I] := exemple(2,I) ;
     I := 1 ;
     while ((X div T[I]) > 0 ) do I := I+1 ;
     writeln (I-1) ;
end.
```

Quel est l'affichage ?
Quelle est l'interprétation de cette valeur ?

1.3 Réécrire le corps de ce programme en utilisant la procédure que vous avez écrite à la question précédente.

Tournez S.V.P.

2.1 Déclarer un type énuméré nommé **t_animal** comportant 5 termes correspondant aux différents animaux domestiques les plus communs : chien, chat, hamster, poisson et oiseau.
et une variable de ce type nommée **animal**

2.2. Déclarer un type de tableau d'entiers dont les indices sont du type t_animal défini à la question précédente; il porte le nom **t_tab**
et une variable de ce type de nom **A1.**

2.3 Ecrire un sous-programme **saisie** qui permette de saisir au clavier les différentes valeurs d'un tableau tel que celui déclaré dans la question précédente.

2.4 Ecrire un sous-programme **somme** qui permette de calculer la somme des éléments d'un tableau de type t_tab

.

2.5 Ecrire un programme utilisant le sous programme et les déclarations précédents, qui

- permette de saisir les éléments de 2 tableaux de type **t_tab** qui représentent le nombre des animaux recensés dans deux villes. Les variables utilisées seront nommées **A1** et **A2**.
- permette de remplir un tableau **Atotal** de type **t_tab** qui contienne le total des animaux dans chaque catégorie,
- permette d'afficher pour chaque ville le pourcentage des différentes catégories d'animaux.

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2007-2008
17 avril 2008

Programmation 2
Contrôle continu 2

aucun document autorisé

1
```
program principal ;

procedure calcul1(X,N:integer; test:boolean;var Y:integer);
var I : integer ;
     begin
          Y := 0 ;
          if (test) then X := X + 1 ;
          for I := 1 to N do Y := Y + X
          writeln(X ,' ', Y)
     end ;

procedure calcul2(N : integer ; var X,Y : integer);
var aux : boolean ;
     I : integer ;
     begin
          aux := ((N mod 2) = 0 ) ;
          if (aux) then X := X + 1 ;
          for I := 1 to N do Y := Y + X ;
          writeln(X, ' ' , Y)
     end ;

var A,B,P : integer ;
begin
     P := 3 ; A := 2 ; B := 3 ;
     calcul1(B,P,P=B,A) ;
     writeln(A, ' ', B, ' ', P) ;
     P := 2 ; A := 2 ; B := 2 ;
     calcul2(P,A,B) ;
     writeln(A, ' ', B, ' ', P)
end.
```

1.1 Préciser, pour chaque sous-programme, à quelle variable du programme principal correspond chaque paramètre.

1.2 Quel est l'affichage ?

1.3 Transformer en fonction la procédure calcul1
et réécrire le programme principal en utilisant cette fonction

Tournez S.V.P.

2 Un médecin doit gérer l'ensemble de ses clients, il connaît, leur **nom**, leur **prenom**, le **poids**. Le nom et le prénom sont des chaînes de longueur maximum 25 caractères, le poids est un tableau de 20 entiers au plus.
On n'oubliera pas de gérer en permanence le nombre des éléments utiles dans les tableaux qui interviennent.
L'ensemble des patients est connu dans un tableau nommé **patients.**
On admettra qu'il existe une procédure **saisie** qui permette de remplir la totalité du tableau des patients

2.1 Ecrire un type **t_patient** qui permet de décrire un patient et un type t_cab qui permet de traiter l'ensemble des patients d'un médecin.

2.2 écrire l'entête du sous-programme **saisie.**

2.3 On se propose d'afficher l'évolution du poids d'un patient. Pour cela on écrira

a/ une fonction permettant de calculer le maximum d'un tableau contenant les poids d'un patient,

b/ une procédure permettant de normaliser les valeurs du tableau en utilisant une transformation linéaire de sorte que le maximum du nouveau tableau soit égal à 20,
par exemple si un tableau T se présente :

70	72	80	80	78

on devra obtenir le tableau normalisé T2 suivant :

$17{,}5 = \frac{70 \cdot 20}{80}$

17,5	18	20	20	19,5

c/ un sous-programme permettant d'afficher, à raison d'une ligne par poids, un nombre de X égal à la partie entière du poids normalisé,
pour l'exemple on aura l'affichage suivant :

```
XXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX
```

d/ un sous-programme mettant en œuvre ces différents sous-programmes pour afficher l'évolution du poids d'un patient.

e/ un appel à ce sous-programme pour afficher l'évolution du poids du patient d'indice 23.

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2007-2008
14 mai 2008

Programmation 2
Contrôle continu 3

aucun document autorisé

1 Aucune variable globale ne devra être utilisée. Il est interdit de stocker le contenu du fichier dans un tableau puis de raisonner sur le tableau. Cette solution n'est pas envisageable car nous ne donnons aucune limite à la quantité de données.
On veut stocker les informations relatives aux logements qui ont été confiés à la location à une agence immobilière. Les informations concernant les logements à la location comprennent l'adresse, le nombre de pièces de l'appartement, la présence d'un balcon, l'exposition du séjour (Nord, Sud, Est, Ouest), le prix proposé, la zone géographique qui est un entier (le numéro de l'arrondissement) et un booléen indiquant si l'appartement est actuellement occupé ou non.
Les informations sont stockées dans un fichier **appart.dat**.

1.1 Ecrire un type **t_appart** qui permet de décrire l'appartement selon la description indiquée plus haut.

1.2 Ecrire un sous-programme qui permet, en utilisant un fichier d'appartements comme appart.dat, d'indiquer le nombre d'appartements disponibles ayant un nombre de pièces donné, dans un secteur géographique donné.

1.3 Ecrire une fonction booléenne qui indique si un appartement d'un nombre de pièces donné est disponible à la location dans un secteur géographique donné.

1.4 Ecrire une procédure qui permet de créer un fichier (par exemple appart1.dat) qui contient uniquement les informations contenant le descriptif des appartements situés dans une zone géographique donnée.

1.5 Ecrire une fonction qui permet de calculer la moyenne des prix de location des appartements de nombre de pièces donné.

1.6 Ecrire une fonction booléenne qui permet de vérifier s'il existe des appartements disponibles dont le prix est inférieur à une valeur donnée.

1.7 Ecrire une procédure **occupe** qui permet de rendre faux le booléen indiquant la disponibilité d'un appartement dont l'adresse et le nombre de pièces est connu dans un fichier décrivant les appartements.

Tournez S.V.P.

2 Un tableau d'entiers à 9 lignes et 9 colonnes a été déclaré par le type **t_tab**.
On dispose d'une procédure **initial** dont l'en-tête est le suivant :
procedure **initial** (var T : t_tab) ;
qui permet de disposer d'un tableau de valeurs nommé T
On dispose d'une procédure **somme_col** dont l'en-tête est le suivant :
procedure **somme_col**(col : integer , var Ncol : integer) ;
qui permet de calculer la somme Ncol des nombres contenus dans la colonne col du tableau.
On dispose d'une procédure **somme_lig** dont l'en-tête est le suivant :
procedure **somme_lig**(lig : integer , var Nlig : integer) ;
qui permet de calculer la somme Nlig des nombres contenus dans la ligne lig du tableau.
On dispose d'une fonction **somme_pave** dont l'en-tête est le suivant :
function **somme_pave**(lig1 , lig2 , col1 , col2 : integer) : integer ;
qui permet de calculer la somme des nombres contenus dans le bloc limité par les lignes lig1 et lig2 et les colonnes col1 et col2 du tableau.
On se propose de vérifier qu'un tableau déclaré de type **t_tab** est un sudoku valide.

2.1 Ecrire une fonction booléenne vérifiant que la somme des éléments de chaque ligne du tableau est égale à 36.

2.2 Ecrire une fonction booléenne vérifiant que la somme des éléments des 9 blocs 3x3 du tableau est égale à 36.

2.3 Ecrire un sous-programme mettant en œuvre ces différents sous-programmes pour afficher si un tableau T de type t_tab est bien un sudoku valide.

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2007-2008
11 juin 2008

Programmation 2
Session 2

aucun document autorisé

Aucune variable globale ne sera utilisée.

1 Un fichier d'enregistrements de nom **fichier1.dat** contient une liste d'enregistrements de type t_element. ce type est défini par

```
t_element = record
              nom : string[20] ;
              taille : integer
            end ;
```

1.1 Ecrire une fonction qui calcule la moyenne des tailles contenues dans le fichier.

1.2 Ecrire une fonction qui calcule le nombre d'individus de taille supérieure à 178.

1.3 Ecrire une fonction booléenne qui indique si un individu de nom donné est contenu dans le fichier.

1.4 Ecrire une fonction qui indique la taille d'un individu de nom donné.

2. La suite définie par la formule de récurrence.

$$\begin{cases} \mathbf{U_{n+1}} = \sqrt{2 + 3.(\mathbf{U_n})^2} \\ \mathbf{U_1} = \mathbf{a} \end{cases}$$ pour n strictement positif

2.1 Ecrire une procédure **itere** qui, à partir de la valeur d'un élément de la suite (U_n), permet de calculer la valeur de l'élément de la suite ayant l'indice strictement supérieur (U_{n+1}).

2.2 Ecrire une fonction **calcule** qui, à partir de la valeur d'un indice n, permet de calculer la valeur de l'élément U_n de la suite, la valeur du premier terme de la suite sera un paramètre du sous programme.

Exercice 1

Ceci n'est pas un exercice de mathématiques, mais la traduction informatique d'un résultat qui a pu être obtenu par ailleurs. On admet que le développement suivant :

$$d_n = 2\left[\frac{x-1}{x+1} + \frac{1}{3}*\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}*\left(\frac{x-1}{x+1}\right)^5 + \cdots + \frac{1}{2n+1}*\left(\frac{x-1}{x+1}\right)^{2n+1}\right]$$

est une approximation de ln(x), l'erreur de méthode vérifiant :

$$\varepsilon_n = \left|\ln(x) - d_n\right| < \frac{1}{2n+3} * \left|\frac{x-1}{x+1}\right|^{2n+3} * \frac{(x+1)^2}{2x}$$

Ecrire un programme permettant de saisir un nombre x réel positif et epsilon un réel positif et de calculer la valeur approchée de ln(x). On indiquera aussi le nombre de termes utilisés pour ce calcul.

Note : remarquer qu'il existe une relation simple entre deux termes consécutifs du développement, d_n et d_{n+1}.

Exercice 1

On se propose d'afficher l'évolution du poids d'un patient. Pour cela on écrira

1°/ une fonction permettant de calculer le maximum d'un tableau contenant les poids d'un patient,

2°/ une procédure permettant de normaliser les valeurs du tableau en utilisant une transformation linéaire de sorte que le maximum du nouveau tableau soit égal à 20,

par exemple si un tableau T se présente :

70	72	80	80	78

on devra obtenir le tableau normalisé T2 suivant :

17,5	18	20	20	19,5

$17{,}5 = \frac{70 \cdot 20}{80}$

3°/ un sous-programme permettant d'afficher, à raison d'une ligne par poids, un nombre de X égal à la partie entière du poids normalisé,

pour l'exemple on aura l'affichage suivant :

XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX

4°/ un sous-programme mettant en œuvre ces différents sous-programmes pour afficher l'évolution du poids d'un patient.

5°/ un programme faisant appel à ces sous-programme pour afficher l'évolution du poids d'un patient d'indice saisi au clavier.

ANNALES

2008 - 2009

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2008-2009
17 mars 2009

Programmation 2
Contrôle continu

aucun document autorisé

1.1 On considère l'entête de la fonction :

```
function cadeau(X : integer; B : boolean) : integer;
```

qui permet de calculer le carré de X si B est vrai et son cube si B est faux.

Ecrire l'entête d'une procédure qui aurait le même effet et que l'on nommera **`gift`**

1.2 Ecrire un sous-programme **`terme`** utilisant la fonction **`cadeau`** que l'on supposera connue et qui permet de calculer la valeur de u_n définie par :

$$u_n = \begin{cases} (n+3)^2 + (2n)^3 & si\ n\ est\ pair \\ (n+4)^3 & si\ n\ est\ impair \end{cases}$$

pour une valeur n entière donnée positive ou nulle.

1.3 Transformer le sous-programme écrit à la question précédente pour créer un sous-programme **`element`** qui utilise la procédure **`gift`** et qui a le même effet que le sous-prgramme **`terme`**.

1.4 Ecrire un programme qui contient les sous-programmes précédents et qui, au choix de l'utilisateur, calcule la somme de N premiers termes de la suite $(u_n)_n$. L'utilisateur devra pouvoir recommencer sa demande autant de fois qu'il le veut, sans sortir du programme.

2 On considère le bloc suivant où le type des variables utilisées sont des entiers :

```
S := 0 ;
for i := 1 to 10 do S := S + i ;
writeln(S) ;
```

a/ Quel est l'affichage ?
b/ Ecrire les lignes équivalentes en utilisant une boucle repeat
c/ Ecrire les lignes équivalentes en utilisant une boucle while

Tournez S.V.P.

3 Soit le programme suivant :

```
Program examen ;

Function F1(N : integer; var X : real; B : boolean ; P : integer) : boolean ;
begin
if (B) then X := X + N ;
B := (X+1) < P ;
Writeln(X) ;
F1 := B ;
End ;

Var X : real ;
      M : integer ;
      Y , Test : boolean ;
Begin
X := 28 ; M := 25 ; test := X<M ;
If test then writeln('on commence le processus') else writeln('on est mal
parti') ;
X := 24 ; test := X<M ;
Y := F1(3 , X , test , M) ;
If test then writeln('ABC') else writeln('DEF');
If Y then writeln('GHI') else writeln('JKL') ;
X := 2 ; test := X<M ;
If test then writeln('on commence le processus') else writeln('on est mal
parti') ;
Y := F1(3 , X, test , M) ;
If test then writeln('MNO') else writeln('PQR');
If Y then writeln('STU') else writeln('VWX') ;
End.
```

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2008-2009
7 avril 2009

Programmation 2

Contrôle continu 2

aucun document autorisé

1.1 Déclarer un type énuméré nommé **t_spec** comportant 6 termes correspondant aux différentes options proposées aux étudiants : systèmes multi-agents, apprentissage, signal, parole, image, réseaux.
et une variable de ce type nommée **choix.**
Déclarer un type de tableau nommé **t_promo** dont les indices sont entiers et dont les éléments sont des noms d'options
et une variable de ce type nommée **promo_choix**.

1.2 Déclarer un type de tableau d'entiers dont les indices sont de type t_spec défini à la question précédente, il porte le nom **t_tab**
et une variable de ce type nommée **C2009**

1.3 Ecrire un sous-programme **bilan** qui, à partir d'un tableau de type t_promo, permet de remplir un tableau de type t_tab qui récapitule le nombre d'étudiants qui ont effectué leur choix dans chaque option.

1.4 Ecrire un sous programme **effectif** qui permet d'afficher l'option la plus souvent choisie et les options qui ont obtenu moins de 15 candidats.

Tournez S.V.P.

2 On considère dans cet exercice des tableaux de chaînes. Dans toutes les questions, il est demandé d'écrire des sous programmes. Sauf mention du contraire, vous choisirez la nature de ce sous programme. Vous veillerez à la définition des arguments en respectant l'hypothèse que le programme principal ne comporte aucune variable globale. Aucune saisie au clavier n'est prévue dans cet exercice.

2.1 Ecrire la déclaration de type des tableaux de chaînes, **t_tabmot**, elle sera utilisée dans toute la suite de l'exercice.

2.2 Ecrire un sous-programme **pluriel** qui ajoute la lettre **s** à la fin de tous les éléments d'un tableau.

2.3 Ecrire un sous-programme **present** qui, pour un tableau et un mot donnés, fournit quand il existe, l'indice du tableau où ce mot est présent. Le sous programme doit comporter un argument ou un résultat booléen indiquant si le mot est effectivement présent dans le tableau.

2.4 Ecrire un sous-programme **fusion** qui, à partir de deux tableaux de chaînes permet de construire un tableau représentant la fusion des deux tableaux et ne comprenant qu'une seule fois les mots communs aux deux tableaux.

'de'
'sur'
'plus'

'sur'
'avec'
'sur'

devient

'de'
'sur'
'plus'
'avec'

2.5 Ecrire une fonction **ftest** puis une procédure **ptest** qui, à partir d'un tableau de chaînes, teste la présence d'un mot ou de sa forme plurielle (avec ajout d'un s) dans le tableau, le résultat est un booléen.

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2008-2009
18 mai 2009

Programmation 2

Contrôle continu 5

aucun document autorisé

1.1 Déclarer un type nommé **t_ville** comportant 4 champs correspondant à un nom de ville (une chaîne de caractères), à sa population (un entier), son département (un entier compris entre 1 et 95) et la région représentée par son indicatif téléphonique (un entier compris entre 1 et 5). Puis déclarer une variable de ce type nommée **ville.**
Déclarer un type de tableau nommé **t_pays** dont les indices sont entiers et dont les éléments sont de type t_ville. Puis déclarer une variable de ce type nommée **pays**.
Déclarer un type de fichier nommé **tf_pays** qui contient la description de villes. Puis déclarer une variable de ce type nommée **f_pays**.
Déclarer un type tableau nommé **t_tab** à deux dimensions dont les indices et les éléments sont des entiers. Puis déclarer une variable de ce type nommée **lien**.

1.2 Ecrire une procédure **tri** qui permet de classer un tableau de type t_pays par ordre décroissant de la population. On veillera à ne pas détruire les données contenues dans le tableau à trier.

1.3 Le fichier **France.dat** contient les informations associées à un ensemble de villes.
Ecrire une procédure **lecture** qui permet de transférer le contenu d'un fichier de ce type dans un tableau.

1.4 Ecrire un sous programme **bilan** qui à partir d'un tableau de type t_pays permet de calculer la moyenne des populations des villes.

1.5 Ecrire un sous programme **gdvilles** qui permet de créer, à partir d'un fichier de villes, un nouveau fichier contenant les villes dont la population est supérieure à un seuil T. Les noms des 2 fichiers ainsi que la valeur du seuil seront des paramètres du sous programme.

Tournez S.V.P.

1.5 Ecrire un sous programme **region** qui, à partir d'un tableau de type t_pays permet de calculer la région qui est la plus peuplée.

1.6 Ecrire un sous programme **accu** qui permet de compter dans un tableau de type t_tab et à partir d'un tableau de type t_pays le nombre des villes en fonction de la région et du département.

1.7 Ecrire un sous-programme **histo** qui, à partir d'un tableau de type t_tab et d'une région, permet d'afficher pour chaque département de cette région, sur des colonnes consécutives, des X correspondant au nombre de villes dans le département.

Le résultat attendu est du type :

Région 1
75 X
77 XXX
78 XX
91 XXXX
92 XXXXX
93 XXX
94 XX
95 XXXX

Université Paris Descartes
UFR Math-Info
Licence 1 – S2

année 2008-2009
22 juin 2009

Programmation 2

Session 2

aucun document autorisé

1.1 Le traitement ou la transformation d'une image sont réalisés dans une structure qui est un tableau à deux dimensions. La position d'un élément d'image est repéré par son indice de ligne et son indice de colonne. L'indice de ligne indique la position verticale l'indice de colonne indique la position horizontale. Dans une image couleur chaque élément d'image est caractérisé par trois entiers compris entre 0 et 255 qui indiquent respectivement la quantité de Rouge, Vert et Bleu qui entrent dans la composition de la couleur. Dans une image à niveaux de gris, chaque élément d'image est caractérisé par un entier compris entre 0 et 255. La valeur est d'autant plus proche de 0 que l'élément d'image est foncé.

- Déclarer un type enregistrement nommé **t_couleur** comportant 3 champs correspondant aux niveaux de chaque couleur.
- Déclarer un type de tableau nommé **t_cimage** à deux dimensions dont les indices sont entiers et dont les éléments sont de type t_couleur.
- Déclarer un type tableau nommé **t_nvgimage** à deux dimensions dont les indices sont des entiers et dont les éléments indiquent un niveau de gris.

1.2 Ecrire une fonction **moyen** qui permet de calculer la valeur moyenne des niveaux de gris d'un tableau de type t_nvgimage.

1.3 Ecrire une fonction **ctonvg** qui, à un élément de type t_couleur, permet d'associer un niveau de gris en calculant la moyenne des niveaux associés aux 3 couleurs.

1.4 Ecrire une procédure **lecturecouleur** qui permet de lire un fichier d'image couleur dont on connaît le nom, le nombre de lignes et de colonnes et qui permet de remplir un tableau de type t_cimage.

1.5 Ecrire une procédure **transfert** qui permet de lire une image couleur connue par son nom, le nombre de lignes et de colonnes et qui permet de stocker dans un autre fichier connu par son nom, l'image à niveaux de gris correspondante.

Tournez S.V.P.

1.6 Ecrire une fonction booléenne qui indique, pour une image de type t_nvgimage qu'une linge d'indice donné ne comporte que des valeurs supérieures à un seuil T donné.

1.7 Ecrire un sous programme **lignebl** qui permet à partir d'une image en niveaux de gris, de créer un nouveau tableau qui représente une image en niveaux de gris dans laquelle on a supprimé les lignes dont toutes les valeurs sont supérieures à un seuil T donné. On supprime ainsi les lignes très claires.

23	32	54	237
214	230	248	245
14	25	46	250

M[2,3] = 248 T = 200

Dans ce contexte la ligne 2 est considérée comme claire.

1.8 Ecrire une fonction **moyencouleur** qui permet de calculer la valeur moyenne des niveaux d'une couleur donnée d'un tableau de type t_cimage. (1 pour Rouge, 2 pour Vert, 3 pour Bleu)

1.9 Ecrire un sous programme **tendance** qui permet de déterminer la couleur dominante déterminée par la plus grande valeur moyenne des couleurs. La réponse sera donnée sous la forme d'un nombre compris entre 1 et 3. (1 pour Rouge, 2 pour Vert, 3 pour Bleu)

Dans tous les cas on utilisera les sous-programmes précédents chaque fois que cela est possible.