Travaux Dirigés 1

Actions paramétrées (Procédures et Fonctions)

Rappel de cours:

- **Déclaration et définition :** Pour les actions paramétrées, on suppose que déclaration et définition sont confondues. Ces déclarations sont placées avant l'algorithme principal (voir exercice 1).
- **Syntaxe :** (Exemples)
  - Une action paramétrée de type procédure avec ***par1*** entier transmis en entrée et ***par2*** entier transmis en sortie :
    Action *nomAction* (E/ ***par1*** :entier; S/ ***par2*** :entier) ;
    Debut … Fin ;
    (au lieu du mot clé **Action** on peut utiliser **Procedure**)
  - Une action paramétrée de type fonction retournant un entier avec ***par1*** entier transmis par valeur et ***par2*** entier transmis par variable :
    Fonction *nomFct* (E/ ***par1*** :entier; Ref/ ***par2*** :entier) : **entier *;***
    Debut … retourne(expression) ; Fin ;
- **Modes de transmission :** Quatre modes de transmission de paramètres : **Entrée** (ou par valeur), **Sortie**, **Mixte** et par **Référence** (ou par variable) spécifiés par E/, S/, ES/ et Ref/ respectivement. Pour expliquer le déroulement avec un schéma de RAM, on supposera qu'il y a création de variables locales de substitution dans l'espace pile pour les 3 premiers modes (E/, S/ et ES/).
  On suppose qu'un tableau peut être transmis par valeur.
- **Visibilité :**
  Les variables déclarées dans l'algorithme principal sont globales. Une action paramétrée peut appeler une autre si la définition de l'appelée est réalisée avant celle de l'appelante. Dans ce cas, les variables de cette dernière sont vues dans l'appelée.

Exercice 1 :

1. Ecrire une action paramétrée `MAXIMUM()` qui détermine le maximum de 2 entiers.
2. En utilisant `MAXIMUM()`, écrire un algorithme qui lit trois entiers et affiche leur maximum.

Exercice 2 :

1. Ecrire une action paramétrée `DIVISE()` qui vérifie si un nombre entier A divise un nombre entier B.

Soit tab un vecteur d'entiers de taille N<=10.

2. En utilisant `DIVISE()`, écrire un algorithme qui permet d'afficher tous les éléments du vecteur tab divisibles par une valeur val donnée.

Exercice 3 :

1. Ecrire une action paramétrée `VABS()` qui, étant donné un entier, calcule sa valeur absolue.

2. Soit T un vecteur de N valeurs entières quelconques (N<=10). Ecrire un algorithme qui détermine la valeur maximale de T, puis en utilisant la fonction précédente affiche la somme des valeurs absolues de tous les nombres de T (sauf la valeur maximale).

Exemple: Pour T=

6	-15	30	-9	140	28	-1

Somme = 6 + 15 + 30 + 9 + 28 + 1

Exercice 4 :

1. Ecrire une action paramétrée `FACT()` qui, étant donné un entier A, calcule sa factorielle.
2. Ecrire une action paramétrée `PUISS()` qui, étant donné un entier A, un entier N, calcule la puissance A^n
3. En utilisant ces deux actions paramétrées, écrire un algorithme qui calcule la valeur de la somme S, tel que : $S = X/1 ! + X^2/2 ! + X^3/3 ! + ... + X^n/n !$

Exercice 5 :

Ecrire une action paramétrée qui permet de vérifier si un mot de longueur L est palindrome. (un mot palindrome est un mot qui se lit indifféremment de gauche à droite ou de droite à gauche. Ex : le mot « radar » est un palindrome)

Exercice 6:

1. Écrire deux actions paramétrées à un argument entier et une valeur de retour entière permettant de préciser si l'argument reçu est multiple de 2 (pour la première fonction) ou multiple de 3 (pour la seconde fonction).
2. Utiliser ces deux actions paramétrées dans un algorithme principal qui lit un nombre entier et qui précise s'il est pair, multiple de 3 et/ou multiple de 6, comme dans cet exemple :

Exécution 1 :

```
Donnez un entier : 9
Il est multiple de 3
```

Exécution 2 :

```
Donnez un entier : 12
Il est pair
Il est multiple de 3
IL est divisible par 6
```

Exercice 7:

Ecrire une Action paramétrée `LongChaine()` qui renvoie la longueur d'une chaîne de caractères.

Exercice 8 :

Écrire une fonction récursive qui calcule la somme des n premiers nombres entiers positifs. Décrire son déroulement avec un schéma de RAM pour n=4.

Exercice 9 :

Écrire une procédure récursive qui permet de trouver le mot miroir d'un mot donné. Ex : le mot miroir de PAVILLON est NOLLIVAP.

Exercices Supplémentaires (facultatifs)

Examen 2011/12

On veut écrire des actions paramétrées permettant d'effectuer des opérations concernant des matrices carrées et des vecteurs binaires (contenant des 0 et 1). On définit le poids d'un vecteur ou d'une matrice comme étant le nombre de 1 qu'il (ou elle) contient.

Ecrire une action paramétrée ***estVecteurBinaire*** qui prend en argument un entier n et un tableau d'entiers V (de taille n) et teste s'il s'agit d'un vecteur binaire (si V ne contient que des 0 ou des 1).

1. Ecrire une action paramétrée ***remplaceBitVecteur*** qui prend en argument un entier n et un vecteur binaire V (de taille n) et qui remplace toutes les occurrences de 1 avec 0 et vice-versa.
2. Ecrire une action paramétrée ***premierDernierBit1*** qui prend en argument un entier n, un vecteur binaire V (de taille n). Cette action paramétrée doit calculer les deux positions du premier et du dernier 1 contenu dans le vecteur. A la sortie, ces deux positions doivent être placées dans deux variables globales ***first*** et ***last***.
3. Ecrire une action paramétrée ***poidsVecteur*** qui prend en argument un entier n et un vecteur binaire V (de taille n) et renvoie le poids de V (un entier compris entre 0 et n) (vous devez utiliser l'action paramétrée ***premierDernierBit1***).
4. Ecrire le programme principale (main) qui :
   - 4.1. Demande de remplir une matrice 5x5.
   - 4.2. Vérifie que la matrice est binaire (pour cela mettez chaque ligne de la matrice dans un vecteur ***MatLigne***).

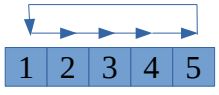
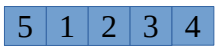
   Si la matrice est binaire :

   - 4.3. Calcule les poids de chaque ligne de la matrice. Ces poids seront stockés dans le tableau ***PoidsLignes***.
   - 4.4. Remplace les 1 de la dernière ligne de la matrice par 0 et vice-versa.
   - 4.5. Affiche la matrice et le tableau ***PoidsLignes***.

Examen 2014/2015

Écrire une action paramétrée **récursive** qui réalise un décalage circulaire à droite (i.e. chaque élément est déplacé vers la case suivante et on suppose que le suivant du dernier est le premier) des éléments d'un tableau d'entiers. L'action paramétrée ne doit pas utiliser un 2ieme tableau.

Exemples :

1. Le décalage du tableau Vect=  donne  .

2. Le décalage du tableau Vect=  donne  .

Travaux Dirigés 2

Enregistrements (Structures)

Rappel de cours:

- On déclare un *type* enregistrement comme suit :

```
Type   nomTypeEnreg = enregistrement
                          liste de champs ;
                     fin ;
```

- Pour en instancier une variable, on utilise :

```
Var   varEnreg : nomTypeEnreg;
```

Exemple : On veut déclarer deux variables d'une structure que nous appelons tPersonne contenant deux champs : nom, de type tableau de caractères, et age, de type entier :

```
Type  tPersonne=enregistrement
                    nom : tableau (20) caractere ;
                    age : entier;
                fin ;
Var   pers1,pers2 :tPersonne;
```

- On peut accéder à un champ d'une variable enregistrement par l'opérateur '.' comme suit : pers1.age
- Une telle variable peut être manipulée comme n'importe quelle variable du type de champ associé.
- Manipuler une variable enregistrement de façon globale n'est permis que pour l'affectation et la transmission de paramètres.

Exercice 1 :

Un nombre complexe `Z` est défini par ses parties réelle `a` et imaginaire `b`.

1. Ecrire un algorithme qui lit deux nombres complexes `Z1` et `Z2` et qui affiche ensuite leur somme et leur produit.
   - `(a+bi) + (c+id) = (a+b) + (b+d)i`
   - `(a+bi) * (c+id) = (ac-bd) + (ad+bc)i`

L'algorithme doit utiliser deux actions paramétrée `saisi()`, `affiche()`,`addition()` et `produit()` qui permettent respectivement de saisir, afficher, additionner et multiplier des nombres complexes.

2. Traduire votre algorithme en C.

Exercice 2 :

Ecrire un algorithme qui permet de gérer un ensemble d'étudiants (< = 100). Chaque étudiant est décrit par son identifiant, son nom, son prénom, son âge et sa moyenne.

Ecrire un algorithme qui permet de :

1. Saisir les informations concernant des étudiants.
2. Afficher les informations saisies et le résultat concernant chaque étudiant sachant que le résultat est soit ADM ou AJN (selon que la moyenne est < 10 ou non).
3. Afficher la moyenne générale des étudiants.

Exercice 3 :

1. Définir un type enregistrement `TEMPS` qui contient les champs `heure`, `minute`, `seconde`.
2. Ecrire une action paramétrée qui réalise la somme `T` de deux durées `T1` et `T2` de type `TEMPS`.
3. Ecrire une action paramétrée `transform()` qui transforme un temps T de type `TEMPS` en un entier S qui exprime ce temps en secondes.

Exemple : pour `T` = 2 heures 10 minutes 37 secondes, `S` = 7837 secondes.
Ecrire une procédure `decompos()` qui décompose un temps `S` exprimé en secondes en un temps `T` de type `TEMPS`.
Exemple : pour `S` = 7837 secondes, `T` = 2 heures 10 minutes 37 secondes.
Etant donnés deux temps `T1` et `T2` de type `TEMPS`, écrire un algorithme qui calcule le temps `T` somme des temps `T1` et `T2` (`T, T1` et `T2` sont de type `TEMPS`) en utilisant les actions `transform()` et `decompos()`.

Exercice 4 :

Écrire un programme qui introduit deux temps en (heure/minute/seconde) et calcule la différence entre eux. Le programme utilisera les actions paramétrées suivantes :

- `struct TEMPS lecture()` // Lecture de temps en HH:MM:SS
- `void affiche(struct TEMPS t)` // Affichage de temps en HH:MM:SS
- `struct TEMPS diff(struct TEMPS start,struct TEMPS end)` // calcule la différence entre les temps `start` et `end`.
- `struct TEMPS decompos(int s)` // Convertit un nombre de secondes en HH:MM:SS

L'affichage doit être comme suit :

```
Donnez le premier temps
 Donnez les heures :4
 Donnez les minutes :51
 Donnez les secondes :20
 4 : 51 : 20
Donnez le deuxieme temps
 Donnez les heures :10
 Donnez les minutes :3
 Donnez les secondes :44
 10 : 3 : 44
Difference entre les deux temps 5 : 12 : 24
Difference en secondes : 18744
```

Exercice 5 :

Considérons les enregistrements suivants :

```
Type
Date = Enregistrement
     Jour, mois, annee : Entier ;
Fin;
Adresse = Enregistrement
                  Numero : entier ;
                  Rue    : chaine [50] ;
                  Ville  : chaine [20] ;
                  Wilaya : chaine [20] ;
```

```
Fin;
Enseignant = Enregistrement
                  Nom, prenom          : chaine [20] ;
                  Date_naiss           : date ;
                  Residence            : Adresse ;
                  Grade                : chaine [10] ;
                  Annee_de_recrutement : entier ;
Fin;
```

Ecrire un algorithme qui permet de :

1. Remplir un vecteur `T` de `N` enseignants (`N ≤ 100`).
2. Afficher à partir de `T` les noms et adresses des enseignants nés avant une date donnée.
3. Afficher les noms et les prénoms des enseignants résidents de la ville de `'Boudouaou'` de la wilaya de `'Boumerdes'`.
4. Afficher le nombre d'enseignants par année de recrutement à partir d'une année donnée.

Exercice 6 :

On souhaite gérer une base de données d'inscriptions pour l'organisation d'un congrès qui dure une journée. Les organisateurs proposent aux participants de s'inscrire pour des repas, ainsi que pour l'hébergement en hôtel. Un participant peut s'inscrire indépendamment aux 2 repas proposés : déjeuner (1500 DA) et/ou dîner (3500 DA) ou aucun. Il n'est pas obligé de prendre un hôtel. S'il en prend un, il peut choisir parmi 2 types d'hôtels différents : 2 étoiles (7500 DA) ou 3 étoiles (10000 DA).

Un participant peut venir accompagné de son conjoint. Dans ce cas, la réservation d'hôtel est identique mais lorsqu'un repas est sélectionné alors il faut en compter 2.

1. Créer un enregistrement `Participant` qui inclut son nom, son prénom ainsi que toutes les autres informations nécessaires à son inscription selon les critères définis ci-dessus.

On privilégiera un enregistrement contenant un nombre minimal de champs. L'enregistrement contient donc les champs suivants :

- `Nom et prenom`
- `Dejeuner` : `Faux` = pas de Déjeuner, `Vrai` = avec Déjeuner
- `Diner` : `Faux` = pas de Diner, `Vrai` = avec Diner
- `Hotel` : `1` = pas d'Hôtel, `2` = 2 étoiles, `3` = 3 étoiles
- `Seul` : `Faux` = avec conjoint, `Vrai` = seul

2. Créer une variable `TabPart` qui est un tableau de 100 enregistrements de type `Participant` (Donnez la déclaration de cette variable).
3. Créer une action paramétrée `liste2Etoiles()` qui reçoit un tableau `TabPart` de type `participant` et le nombre de participants pour afficher les nom et prénoms des personnes qui ont choisi de réserver un hôtel 2 étoiles.
4. Créer une action paramétrée `NbDej()` qui accepte comme argument un tableau `TabPart` de type `Participant` et le nombre de participants, puis retourne le nombre de déjeuners à prévoir.
5. Créer une action paramétrée `Montant()` qui calcule, pour un participant donné en argument, le montant de sa facture.

Exercices supplémentaires

Examen 2014 - 2015

Un pharmacien veut gérer les opérations de vente et d'achat de `N` médicaments différents. Pour cela, il associe à chaque médicament les informations : `NomMedicament` de 20 caractères, `NbBoites` de type entier, `PrixUnitaire` de type réel, `DatePeremption` de type date et `Fournisseur` de 20 caractères. Les dates étant définies à l'aide des informations : `annee`, `mois`, `jour` qui sont de type entier.

1. Proposer les structures `tDate` et `tMedicament` pour contenir de telles informations.
2. Ecrire une fonction pour remplir un tableau `TabMedicament` de structures `tMedicament`.
3. Ecrire une fonction qui affiche tous les médicaments livrés par un fournisseur donné.
4. Ecrire une fonction qui retourne le prix total des médicaments périmés à la date : 04 septembre 2015.
5. Ecrire une fonction qui permet au pharmacien de mettre à jour le tableau `TabMedicament` suite à une opération de vente ou à une opération d'achat d'un médicament. On distinguera ces deux opérations à l'aide d'un paramètre en entrée, de type caractère, qui vaut `'v'` pour la vente et `'a'` pour l'achat.

Remarque : On suppose que chaque médicament est livré par un seul fournisseur.

Rattrapage 2019 - 2020

La Fédération Algérienne de Football (FAF) veut automatiser la gestion des matchs du championnat. Chaque match est représenté par les noms des deux équipes qui l'ont disputé et le nombre de buts marqués par chaque équipe (voir exemple d'exécution). Pour simplifier, on supposera que le nombre de matchs **n** du championnat ne dépasse pas 50.

1. Proposer une déclaration d'un type enregistrement (ou structure C) **tMatch** qui permet de représenter les données d'un match du championnat.
2. Écrire une action paramétrée **fillvMatch()** qui permet de remplir les données de **n** matchs (n<=50) dans un tableau **vMatch** d'enregistrements **tMatch**.
3. Écrire une action paramétrée **bestStrikeTeam()** qui détermine l'équipe ayant la meilleure attaque (i.e ayant le plus grand nombre global de buts marqués) dans le championnat.

Exemple d'exécution pour la question 2 (ici l'affichage des matchs et en plus, il n'est pas demandé) :

```
Donnez le nombre de matchs :3
***** Saisie match Numero 1 ****
Donnez nom equipe 1  :ESS
Donnez nom equipe 2  :CSC
Donnez nombre but(s) ESS  :3
Donnez nombre but(s) CSC  :2
***** Saisie match Numero 2 ****
Donnez nom equipe 1  :MOB
Donnez nom equipe 2  :JSK
Donnez nombre but(s) MOB  :2
Donnez nombre but(s) JSK  :2
***** Saisie match Numero 3 ****
Donnez nom equipe 1  :JSMB
Donnez nom equipe 2  :MOB
Donnez nombre but(s) JSMB  :1
Donnez nombre but(s) MOB  :3

Les matchs que vous avez introduits sont: !!!(cet affichage est en plus)!!!
********** Match  1 ********
ESS 3 - CSC 2
********** Match  2 ********
MOB 2 - JSK 2
********** Match  3 ********
JSMB 1 - MOB 3
```

Travaux Dirigés 3

Fichiers

Rappel de cours:

On s'intéresse aux fichiers de type texte, structurés et à accès séquentiel.

- **Déclaration** : les mots-clé sont soulignés

```
type       type_enregistrement=enregisterment
                champs1 : type_champs1 ;
                ...
                champsn : type_champsn ;
var        nom_fichier_ram : fichier type_enregistrement ;
```

- **Ouverture** : `nom_fichier_ram ← ouvrirFich(nom_fichier_phy, mode) ;`

on se contente de trois modes 1. `LECTURE` : lecture seule , 2. `ECRITURE` : écriture seule avec tête d'écriture en début de fichier et 3. `AJOUT` : écriture seule avec tête d'écriture en fin de fichier.

- **Fermeture** : `fermerFich(nom_fichier_ram) ;`
- **Prédicat de détection de fin de fichier** : `finFich(nom_fichier_ram);` retourne vrai si la tête de lecture a atteint la fin du fichier faux sinon.
- **Lecture dans un fichier texte** : `lireFich(nom_fichier_ram, variable) ;` ignore les séparateurs au début puis lit une suite de caractères jusqu'au premier séparateur ou la fin de fichier et met la valeur convertie dans `variable`.
- **Lecture d'un caractère depuis un fichier texte** : `lireCarFich(nom_fichier_ram, variable_car);` lit un seul caractère et déplace la tête de lecture vers le caractère suivant.
- **Écriture dans un fichier texte** : `ecrireFich(nom_fichier_ram, expression) ;` écrit dans le fichier une suite de caractères correspondant a la conversion de l'évaluation de `expression` en chaîne de caractères.
- **Changement de nom de fichier** : `renommer(nom_fichier_ext_old, nom_fichier_ext_new);` affecte au fichier physique `nom_fichier_ext_old` le nouveau nom `nom_fichier_ext_new`.
- **Séparateurs** : Deux constantes prédéfinies : `SDC` (séparateur de champ) pour séparer les champs et `FDL` la constante fin de ligne pour séparer les enregistrements.

Exemple : Un algorithme qui crée un fichier texte structuré (nom + age) puis affiche son contenu.

```
Algorithme GestionPersonnel;
Type
  tPersonne=enregistrement
              nom:chaine(15);
              age:entier;
  fin;
var
  fich: fichier tPersonne;
  nomFichPhy: chaine(30);
  pers:tPersonne;
  i,nbrPers:entier;
debut
  {Partie I: Créer et remplir le fichier}
  lire(nomFichPhy);
  fich ← ouvrirFich(nomFichPhy,ECRITURE);
  lire(nbrPers);
  Pour i ← 1 a nbrPers faire
      lire(pers.nom);
      ecrireFich(fich,pers.nom,SDC);
      lire(age.nom);
      ecrireFich(fich,age.nom,FDL);
  fpour;
  fermerFich(fich);
  {Partie II: afficher le contenu du fichier}
  fich ← ouvrirFich(nomFichPhy,LECTURE);
  tant que non(finFich(fich)) faire
       lireFich(fich,pers.nom);
       ecrire(pers.nom);
       lireFich(fich,age.nom);
       ecrire(age.nom);
  ftq;
  fermerFich(fich);
Fin.
```

Exercice 1 :

On veut gérer un groupe d'étudiants en utilisant un fichier «`etudiant.txt`» dont les informations sont structurées selon la manière suivante:

- Numéro de matricule (entier)
- Nom (chaîne de caractères)
- Prénom (chaîne de caractères)
- Moyenne (réel)

En utilisant les actions paramétrées, écrire un algorithme qui permet de :

1. Créer le fichier «`etudiant.txt`». Le nombre d'étudiants est à fournir par l'utilisateur.

2. Afficher les informations des étudiants ainsi que la moyenne des notes du groupe.

2. Ajouter un nouvel étudiant au groupe.

3. Insérer un nouvel enregistrement dans «`etudiant.txt`» en supposant que le fichier est trié relativement au champ Nom.

4. Supprimer dans «`etudiant.txt`» tous les étudiants dont la moyenne est inférieure à 10.

ETLD 2013-2014 :

Une entreprise veut automatiser la gestion de carrière de ces employés. Elle dispose sur papier d'informations ayant la forme suivante (dans ce qui suit, on ne représente qu'une partie des listes):

`Liste des employés`			
Nom	Prénom	Poste	Date de recrutement
Benmalki	Ali	Directeur	23/05/2000
Mahmoudi	Zaki	Agent	20/04/2010
Bousri	Sami	Mécanicien	01/06/2005
Ouafi	Aissa	Agent	04/08/1998
Dardour	Imed	Chauffeur	05/03/2001
Sidhoum	Rima	Secrétaire	12/07/2001
---	---	---	---
---	---	---	---

`Liste des postes`	
Poste	Salaire
Agent	20000
Directeur	55000
Mécanicien	25000
Chauffeur	22000
Secrétaire	25000
---	---
---	---

1. Proposer une structure (enregistrement) pour stocker une ligne de `Liste des employés` et une autre pour stocker une ligne de `Liste des postes`.

2. Écrire une action paramétrée pour stocker ces informations dans deux tableaux de structures : Le premier pour `Liste des employés` et le deuxième pour `Liste des postes`.

3. En utilisant les tableaux de la question 2, écrire une action paramétrée qui détermine le salaire d'un employé donné.

Afin de préserver ces informations, cette entreprise décide d'utiliser des fichiers au lieu de tableaux.

4. Écrire une action paramétrée pour stocker ces informations dans deux fichiers : Le premier pour `Liste des employés` et le deuxième pour `Liste des postes`.

5. En utilisant ces deux fichiers uniquement, écrire une action paramétrée qui affiche les employés ayant un salaire inférieur strictement à 25000 DA.

Exercices supplémentaires

Exercice 3 :

1. Écrire un programme qui permet d'enregistrer dans un fichier un nombre quelconque de valeurs entières positives fournies au clavier.

2. Écrire un programme qui permet de retrouver, dans le fichier précédent, une valeur de rang donné. On fournira un message dans le cas où le rang demandé dépasserait la taille du fichier.

Exercice 4 :

Écrire un programme qui lit 10 valeurs entières d'un fichier de 10 entiers, les place dans un tableau, trie le tableau dans l'ordre croissant et place ce tableau trié dans le même fichier à la place des 10 valeurs de départ.

Exercice 5 :

1. Écrire une fonction qui permet de créer un fichier contenant une succession d'informations fournies par l'utilisateur. Chaque information correspond à un point d'un plan défini par un nom formé d'une lettre, et par deux coordonnées entières. L'exécution de cette fonction donnerait par exemple :

Nom du fichier à créer : courbe.txt

Donner le nom (* pour finir) et les coordonnées des points :

```
C    12   34
F    8    121
Z    152  95
X    25   74
*
```

2. Écrire une fonction qui permet de lister le contenu du fichier formaté créé précédemment.

3. Écrire une fonction qui permet de produire, à la demande de l'utilisateur, les informations de n'importe quel point dont le rand est fournit. Exemple d'exécution :

```
Nom du fichier a consulter : courbe.txt
Votre fichier comporte 4 points
Numéro du point cherché  (0 pour finir) ? : 4
Point numéro 4 est : X 25 74
Numéro du point cherché (0 pour finir) ? : 2
```

```
Point numéro 2 est : F 8 121
Numéro du point cherché  (0 pour finir) ? : 6
Numéro incorrect
Numéro du point cherché (0 pour finir) ? : 0
A bientôt !
```

ETLD 2014-2015 :

Une ménagère fait ses courses depuis deux supérettes : une à Alger et l'autre à Blida. Elle dispose sur papier de deux listes d'historique d'achats, une pour chaque supérette, contenant les informations : article, prix, et date d'achat en jour/mois/année, ordonnées par la date d'achat. Voici un exemple de ses deux listes :

Liste d'achat supérette d'Alger		
article	Prix en DA	date
Pain	50	25/04/2014
Tricot	900	25/04/2014
Cuisinière	35000	10/01/2015
Viande	1500	12/05/2015

...

Liste d'achat supérette de Blida		
article	Prix en DA	date
Légumes	500	25/02/2014
Chemise	1800	11/03/2014
Fruits	450	08/07/2014
Viande	2000	12/06/2015

...

Cette ménagère veut automatiser la gestion de ses dépenses.

1. Proposer les structures de donnée pour représenter les informations d'un achat.
2. Proposer une fonction ou procédure pour remplir un tableau de liste d'achats, puis donner les appels à cette fonction pour remplir chacun des deux tableaux `listAchatAlger` et `listAchatBlida`.
3. Proposer une fonction pour déterminer à partir des tableaux `listAchatAlger` et `listAchatBlida` le montant global des dépenses entre deux dates données.
4. Proposer une fonction ou procédure pour sauvegarder les informations des deux tableaux `listAchatAlger` et `listAchatBlida` dans deux fichiers `fichAchatAlger` et `fichAchatBlida`.
5. Proposer une fonction ou procédure pour afficher a partir des fichiers `fichAchatAlger` et `fichAchatBlida`, les mois où les dépenses dépassent 30000 DA.

Remarque : On suppose que les informations des champs sont sans espace.

UMBB, Faculté des Sciences, Département Informatique, MI, Première année, Semestre 2. Matière : Algo. et structures de données 2

Travaux Dirigés 4

Pointeurs & Variables dynamiques

Rappel:

- *Pointeurs*
  - **Déclaration** : var variable_pointeur : pointeur type_pointé ;
    Exemple : var ptr : pointeur entier ;
  - **Opérateurs** : adresse &, contenu *, affectation ←, addition +, soustraction −;
  - **Relation avec les tableaux** : Si p est un pointeur vers un type T et vect un tableau d'éléments de type T et i un entier alors l'affectation p←&vect[0] permet
    1. d'utiliser p[i] et vect[i] de manière indifférente ;
    2. de rendre p+i et &p[i] égaux ;
    3. de donner un sens a la condition (p<p+i) qui est vraie ssi i>0.

    (dans cette série, on supposera que les indices de tableaux varient de zéro à la taille maximale -1)
  - **Constante prédéfinie** : la constante pointeur nul est notée NUL.
- *Variables dynamiques :*
  - **Allocation** : allouer(*ptr*);
  - **Liberation** : liberer(*ptr*) ;

Exercice 1

En utilisant un schéma RAM, déterminer les valeurs de t a la fin des instructions suivantes :

```
Algorithme manipPointeur ;
var
   t : tableau[5] entier ;
   p1 : pointeur entier ;
   p2 : pointeur pointeur entier ;
   pt : tableau (2) pointeur entier;
debut
   <code pour initialiser t par les valeurs 1 à 5>
   p1 = &t[2];
   p2 = &p1;
   pt[0] = &t[3];
   (*p2) ← (*p2) +1;
   **p2 = 5;
   pt[1] = t + 1;
   (*p1) ← (*p1) +1;
   t[t[t[2]]-2] ← t[t[t[2]]-2]+1 ;
   *pt[1] = *(pt[0]+1);
   *(pt[1]-1)=2**(*p2-1);
Fin.
```

indication : 6,6,3,6,6.

Exercice 2

Écrire un algorithme qui calcule la somme des éléments d'un vecteur d'entiers. Le parcours du tableau sera effectué par un pointeur.

Exercice 3

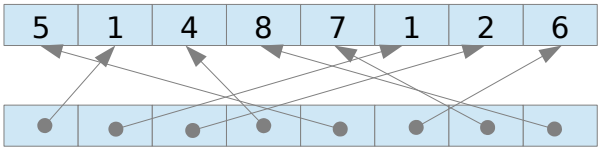
Écrire un algorithme qui permet de permuter le contenu de deux pointeurs d'entiers. Chaque pointeur pointe sur une variable entière statique à lire. Cet algorithme devra utiliser les actions paramétrées pour réaliser cette permutation.

Exercice 4

Étant donné un tableau d'entiers **`vect`**. On veut définir les valeurs d'un tableau de pointeurs d'entiers **`ptrVect`** qui permettra d'afficher les valeurs de **`vect`** de façon triée sans changer les valeurs ni l'ordre dans **`vect`**.

Exemple :

Pour **`vect`** =



`ptrVect` sera . En parcourant **`ptrVect`** et affichant les valeurs pointées, on obtient les valeurs de vect triées 1 1 2 4 5 6 7 8.

Exercice 5

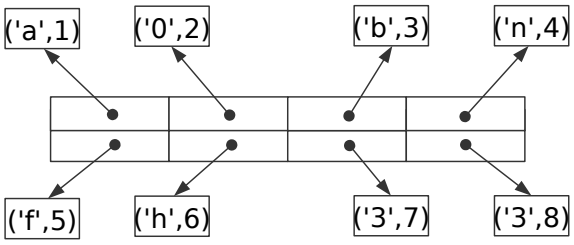
En utilisant les variables dynamiques, écrire un algorithme qui :

- crée deux variables dynamiques en utilisant deux pointeurs ptr1, ptr2 sur des entiers;
- demande à l'utilisateur de saisir les valeurs pointées par ptr1 et ptr2;
- enregistre dans une troisième variable dynamique pointée par ptr3, la somme des deux valeurs saisies précédemment.

Exercice 6 (ETLD2013)

On considère la matrice de pointeurs du schéma ci-contre :

1. proposer une déclaration pour une telle structure.
2. proposer des instructions pour initialiser cette structure de données avec les valeurs illustrées (les constantes de type caractère sont a lire).



Exercices supplémentaires

Exercice 7

Dans cet exercice, on utilisera les pointeurs pour le parcours. Écrire un algorithme qui inverse les éléments d'un tableau d'entiers (le 1^{ier} devient dernier, le 2^{ieme} devient avant dernier et ainsi de suite) en utilisant :

- deux tableaux ;
- un unique tableau.

Exercice 8 (ETLD2013)

Pour le code suivant, compléter la colonne « valeur » du tableau suivant par les valeurs affichées à l'écran. On suppose que les adresses mémoires 2548 et 2544 (%X définit un format hexadécimal mais pour simplifier on peut supposer le format décimal) sont attribuées par le compilateur et que la taille d'un int est de quatre octets.

```
0  include ...
1  void main()
2  {
3      int a=3 , b=5 ;
4      int *ptr1,*ptr2 ;
5      ptr1=&a ;ptr2=NULL ;
6      printf(" a: %d\n",a) ;
7      printf(" &a : %X\n",&a) ;
8      printf(" b: %d\n",b) ;
9      printf(" &b : %X\n",&b) ;
10     printf(" ptr1: %X\n",ptr1) ;
11     printf(" *ptr1: %d\n",*ptr1) ;
12     a=8 ;
13     printf(" a : %d\n",a) ;
14     printf(" &a : %X\n",&a) ;
15     printf(" ptr1 : %X\n",ptr1) ;
16     printf(" *ptr1 : %d\n",*ptr1) ;
17     *ptr1=12 ;
18     printf(" a : %d\n",a) ;
19     printf(" &a : %X\n",&a) ;
20     printf(" ptr1 : %X\n",ptr1) ;
21     printf(" *ptr1 : %d\n",*ptr1) ;
22     ptr2=&b ;
23     printf(" b : %d\n",b) ;
24     printf(" &b : %X\n",&b) ;
25     printf(" ptr2 : %X\n",ptr2) ;
26     printf(" *ptr2 : %d\n",*ptr2) ;
27     ptr2=b+ptr1 ;
28     printf(" b : %d\n",b) ;
29     printf(" &b : %X\n",&b) ;
30     printf(" ptr2 : %X\n",ptr2) ;
31     printf(" *ptr2 : %d\n",*ptr2) ;
32     b=2*b+*ptr1 ;
33     a=b+10 ;
34     ptr1=ptr2 ;
35     ptr2=&a ;
36     printf(" a : %d\n",a) ;
37     printf(" &a : %X\n",&a) ;
38     printf(" ptr1 : %X\n",ptr1) ;
39     printf(" *ptr1 : %d\n",*ptr1) ;
40     printf(" b : %d\n",b) ;
41     printf(" &b : %X\n",&b) ;
42     printf(" ptr2 : %X\n",ptr2) ;
43     printf(" *ptr2 : %d\n",*ptr2) ;
44     b=a-*ptr1 ;
45     b=2*a+10 ;
46     ptr2=ptr1 ;
47     ptr1=ptr2 ;
48     printf(" a : %d\n",a) ;
49     printf(" &a : %X\n",&a) ;
50     printf(" ptr1 : %X\n",ptr1) ;
51     printf(" *ptr1 : %d\n",*ptr1) ;
52     printf(" b : %d\n",b) ;
53     printf(" &b : %X\n",&b) ;
54     printf(" ptr2 : %X\n",ptr2) ;
55     printf(" *ptr2 : %d\n",*ptr2) ;
56 }
```

N° de ligne	Message	Valeur
6	a:	
7	&a :	2548
8	b:	
9	&b :	2544
10	ptr1:	
11	*ptr1:	
13	a :	
14	&a :	
15	ptr1 :	
16	*ptr1 :	
18	a :	
19	&a :	
20	ptr1 :	
21	*ptr1 :	
23	b :	

24	&b :	
25	ptr2 :	
26	*ptr2 :	
28	b :	
29	&b :	
30	ptr2 :	
31	*ptr2 :	
36	a :	
37	&a :	
38	ptr1 :	
39	*ptr1 :	
40	b :	
41	&b :	
42	ptr2 :	
43	*ptr2 :	
48	a :	
49	&a :	
50	ptr1 :	
51	*ptr1 :	
52	b :	
53	&b :	
54	ptr2 :	
55	*ptr2 :	