Sommaire des activités

- Activité 1: Manipulation des traitements alternatifs
- Activité 2: Manipulation des traitements itératifs
- Activité 3: Tableaux et chaines de caractères
- Activité 4: Fichiers (QCM)
- Activité 5: Procédures et fonctions
- Activité 6: Récursivité

Activité1

Traitements alternatifs

Description de l'activité

Compétences visées

- Maitrise des structures alternatives (utilisation et syntaxe)
- Choix de la meilleure structure alternative pour un problème donné

Recommandations clés

- Bonne compréhension du problème
- Maitrise des notions vues dans le cours

Etapes

1. Demander de lister les entrées d'un problème ainsi que leurs types
2. Demander de lister les résultats calculés pour la résolution du problème ainsi que leurs types
3. Questionner sur les formules permettant de déterminer ces résultats
4. Guider les stagiaires à écrire l'algorithme en se basant sur les résultats obtenus précédemment

Description de l'activité

Enoncé de l'exercice1:

Un organisme de location de voiture propose deux formules de location :

1. Location au kilomètre :

- pour les 100 premiers kilomètres : tarif t1 au km,
- pour les kilomètres de 101 à 1000 : tarif t2 au km,
- au delà de 1000 kilomètres : tarif t3 au km.

2. Forfait journalier : kilométrage illimité au prix t4 par jour.

Dans les deux cas, il convient d'ajouter une assurance comptabilisée par jour et dont le montant est donné.

Ecrire un algorithme qui lit le nombre total de kilomètres et le nombre de jours de location ainsi que les tarifs t1, t2, t3 et t4 calcule les coûts totaux des deux tarifications, affiche les 2 coûts et indique au client la solution la plus avantageuse

Description de l'activité

1. **Demander de lister les entrées du problème ainsi que leurs types**
2. **Demander de lister les résultats calculés pour la résolution du problème ainsi que leurs types**
3. **Questionner sur les formules permettant de déterminer ces résultats**
4. **Guider les stagiaires à écrire l'algorithme en se basant sur les résultats obtenus précédemment**

Description de l'activité

Enonvé de l'exercice2:

Écrire un algorithme qui étant donnés deux entiers représentant un mois et une année affiche le nombre de jours du mois de cette année. On prendra en considération l'année bissextile (divisible par 4 et ne se termine pas par 00 ou elle se termine par 00 et son quotient par 100 est divisible par 4).

1. **Demander de lister les entrées du problème ainsi que leurs types**
2. **Faire rappeler la formule permettant de vérifier si une année a est bissextile ou non**
3. **Questionner sur la possibilité d'utiliser la structure alternative Si_alors_sinon pour écrire l'algorithme**
4. **Demander de lister les variables sur lesquelles un test sera effectué**
5. **Guider les stagiaires à écrire l'algorithme en se basant sur les résultats obtenus précédemment**

Correction de l'activité

Exercice1:

1. **Demander de lister les entrées du problème ainsi que leurs types**

t1, t2, t3, t4 (réels) → les tarifs proposées

km(réel)→ le nombre de kilométrage parcouru

ass (réel) → tarif de l'assurance

Jour (entier)→nombre de jours de location

2. **Demander de lister les résultats calculés pour la résolution du problème ainsi que leurs types**

Tarif1→ tarif calculée pour une location au kilométrage

Tarif2→ tarif calculée pour un forfait journalier

Correction de l'activité

3. Questionner sur les formules permettant de déterminer ces résultats

Tarif 2?

Tarif 2= jour * (t4+ass)

Tarif 1?

Si km ≤ 100 → tarif1 := km * t1

Si km>100 et <1000 tarif1 := 100*t1 + (km-100)*t2

Si km>1000 tarif1 := 100*t1 + 900*t2 + (km-1000)*t3

Correction de l'activité

4. Guider les stagiaires à écrire l'algorithme en se basant sur les résultats obtenus précédemment

```
Tarif
Var  t1, t2, t3, t4, ass, km, tarif1, tarif2 : réel
      Jour : entier
Début
            Lire(t1,t2,t3,t4 ,ass,km,jour)
            Si  km ≤ 100
            Alors  tarif1 := km * t1
            Sinon  tarif1 := 100*t1
                        si  km ≤ 1000
                        alors  tarif1 := tarif1 + (km-100)*t2
                        sinon   tarif1 := tarif1 + 900*t2 + (km-
1000)*t3
                        fsi
            fsi
```

```
tarif1 := tarif1 + (jour * ass)
            tarif2 := jour * (t4+ass)
            écrire(''Le  coût selon la formule 1 est :'', tarif1)
            écrire(''Le  coût selon la formule 2 est :'', tarif2)
            si  tarif1 = tarif2
            alors  écrire(''Les  2 formules ont le même coût'')
            sinon  si  tarif1 < tarif2
                        alors     écrire(''La   formule   1   est   plus
avantageuse'')
                        sinon     écrire(''La   formule   2   est   plus
avantageuse'')
                        fsi
            fsi
Fin
```

Correction de l'activité

Enoncé de l'Exercice 2

Écrire un algorithme qui étant donnés deux entiers représentant un mois et une année affiche le nombre de jours du mois de cette année. On prendra en considération l'année bissextile (divisible par 4 et ne se termine pas par 00 ou elle se termine par 00 et son quotient par 100 est divisible par 4).

1. Demander de lister les entrées du problème ainsi que leurs types

m → mois (entier)

a → année (entier)

j → jour (entier)

2. Faire rappeler de l'expression permettant de vérifier si une année a est bissextile ou non

si (a mod 4 = 0) ET ((a mod 100 ≠ 0) OU (a mod 400 = 0)) alors a est bissextile sinon non

Correction de l'activité

3. Questionner sur la possibilité d'utiliser la structure alternative Si_alors_sinon pour écrire l'algorithme

Non car dans ce cas l'emboîtement de si en cascade est fastidieux à écrire. Il est préférable d'utiliser le schéma conditionnel généralisé

4. Demander de lister les variables sur lesquelles un test sera effectué

La variable a pour déterminer s'il s'agit d'une année bissextile et ainsi déterminer le nombre de jour pour le mois de février

La variable m pou déterminer le nombre de jour pour les autres mois.

5. Guider les stagiaires à écrire l'algorithme en se basant sur les résultats obtenus précédemment

Correction de l'activité

```
Nb-jours
Var m , a , j: entier
Debut
        lire( m , a )
        cas m de :
          1   :
          3   :
          5   :
          7   :
          8   :
          10 :
          12 :     j := 31
          4    :
          6    :
          9    :
          11  :   j := 30
          2     :  si  (a mod 4 = 0) ET ( (a mod 100 ≠ 0) OU (a
mod 400 = 0))
                    alors  j := 29
                    sinon j := 28
                        fsi
        default  :  j := 0
        fcas
        écrire(Le nombre de jours est :, j)
Fin
```

Activité2

Traitements itératifs

Description de l'activité

Compétences Visées

- Maitrise des structures itératives (utilisation et syntaxe)
- Choix de la meilleure structure itérative pour un problème donné

Recommandations clés

- Bonne compréhension de l'exercice
- Maitrise des notions vues dans le cours

Etapes

1. Questionner sur la solution possible pour vérifier que l'entier N saisi est >0
2. Demander de préciser le nombre de structures itératives à programmer ainsi que leurs types
3. Demander de proposer un algorithme pour la résolution de ce problème

Description de l'activité

Enoncé de l' Exercice1 (nombre d'itérations connu)

Écrire un algorithme qui lit un entier N > 0 puis saisit une suite de N quadruplets (un quadruplet est une suite de 4 entiers). Pour chacun des quadruplets, l'algorithme doit afficher le minimum et le maximum des 4 nombres. Enfin, on affichera le plus petit et le plus grand de tous les nombres.

1. **Questionner sur la solution possible pour vérifier que l'entier N saisi est >0**
2. **Demander de préciser le nombre de structures itératives à programmer ainsi que leurs types**
3. **Demander de proposer un algorithme pour la résolution de ce problème**

Correction de l'activité

Exercice1:

Écrire un algorithme qui lit un entier N > 0 puis saisit une suite de N quadruplets (un quadruplet est une suite de 4 entiers). Pour chacun des quadruplets, l'algorithme doit afficher le minimum et le maximum des 4 nombres. Enfin, on affichera le plus petit et le plus grand de tous les nombres.

1. Questionner sur la solution possible pour vérifier que l'entier N saisi est >0

Utiliser la structure répéter jusqu'à

Répéter

Lire(N)

Jusqu'a N > 0

2. Demander de préciser le nombre de structures itératives à programmer ainsi que leurs types

Correction de l'activité

3 structures itératives:

1 structure itérative pour vérifier que N est positif (répéter—jusqu'à)

1 structure itérative pour la saisie de N (nombre de répétitions N) (Pour)

1 structure itérative pour la saisie quadriplets (nombre de répétitions 4) (Pour)

3. Demander de proposer un algorithme pour la résolution de ce problème

Correction de l'activité

```
Quadruplets
Var  N, I, j, x , min, max, minimum, maximum :
entier
Debut
            Répéter
            Lire( N )
            Jqa  N > 0
      minimum := +∞
      maximum := -∞
      Pour  I  de  1  à  N  faire
                        min := +∞
                        max := -∞
                        Pour  j  de  1  à  4  faire
                                    Lire ( x )
                                    Si  x  <  min   alors
min := x  fsi
                                    Si  x  >  max   alors
max := x  fsi
                        Finpour
```

```
Ecrire ( ''le minimum du'',i,''ème quadruplet est :'',min)
                        Ecrire ( ''le maximum du'',i,'ème
quadruplet est :'',max)
                        Si  min  <  minimum  alors
minimum := min  fsi
                        Si  max  >  maximum  alors
maximum := max  fsi
      Fpour
      Ecrire ( ''le minimum de tous les nombres
est :'',minimum)
            Ecrire ( ''le maximum de tous les nombres
est :'',maximum)
Fin
```

Activité3

Tableaux et chaines de caractères

Description de l'activité

Compétences visées
• Maitrise de la manipulation des tableaux vecteurs et des matrices (lecture, saisie, remplissage, parcours, etc)
Recommandations clés
• Maitrise des structures itératives et alternatives
Etapes
1. Proposer de définir une variable T de type tableau 2. Demander de préciser le type de la structure itérative à utiliser pour la résolution de ce problème 3. Donner une indication: Ajouter une variable booléenne Test qui est initialisée à vrai et prend faux si T[i]>T[i+1] 3. Demander de proposer un algorithme pour la résolution de ce problème en se basant sur les étapes précédentes

Description de l'activité

Enoncé de l'Exercice 1:

Ecrire un algorithme qui étant donné un tableau contenant N entiers (N ≤ 100) (on suppose le tableau est rempli) retourne :

- Vrai : si le tableau est ordonné dans un ordre croissant
- Faux : si le tableau n'est pas ordonné

Exemple : Si le tableau contient les 7 valeurs suivantes : [5 , 8 , 9, 10, 15, 20 , 50]

La fonction retournera Vrai

Par contre si le tableau contient les 7 valeurs : [5 , 8 , 10, 9, 15, 20 , 50]

La fonction retournera Faux.

Description de l'activité

1. Proposer de définir une variable T de type tableau

2. Demander de préciser le type de la structure itérative à utiliser pour la résolution de ce problème

3. Donner une indication: Ajouter une variable booléenne Test qui est initialisée à vrai et prend faux si T[i]>T[i+1]

3. Demander de proposer un algorithme pour la résolution de ce problème en se basant sur les étapes précédentes

Description de l'activité

Enoncé de l'exercice 2:

Écrire un algorithme qui lit 2 entiers N et P (> 0 et ≤ 100) puis saisit N*P entiers et les stocke dans une matrice N lignes P colonnes ligne par ligne.

Ensuite, l'algorithme affiche le contenu de la matrice ligne par ligne et alternativement de gauche à droite puis de droite à gauche.

Si la matrice est la suivante

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Les éléments seront affichés dans cet ordre :

1, 2, 3, 4

8, 7, 6, 5

9, 10, 11, 12

16, 15, 14, 13

Description de l’activité

1. **Demander de proposer un algorithme qui permet de saisir une matrice**
2. **Demander de proposer un algorithme qui permet de permuter deux variables**
3. **Demander d’écrire un algorithme permettant un affichage accordéon de la matrice**

Correction de l'activité

Enoncé de l'exercice1:

1. Proposer de définir une variable T de type tableau

Tab_entier = tableau[1..100] d'entier

T: Tab_entier

2. Demander de préciser le type de la structure itérative à utiliser pour la résolution de ce problème

Répéter….. jusqu'à ou la structure Tanque puisque le nombre d'itérations n'est pas connu à l'avance

3. Donner une indication: Ajouter une variable booléenne Test qui est initialisée à vrai et prend faux si T[i]>T[i+1]

4s. Demander de proposer un algorithme pour la résolution de ce problème en se basant sur les étapes précédentes

Correction de l'activité

```
Est_ordonné
Var
   i : entier
   test :  booléen
  T : tableau[1..100] d'entier
Début
    test := vrai
    i := 1
    Tant que  i  <  N  ET   test    faire
              Si  T[ i ] > T[ i+1 ]
              Alors  test := faux
              Fsi
              i := i+1
    Fait
    Si test = vrai alors ecrire (''tableau ordonné)
    Sinon ecrire (''tableau non ordonné'')
Fin
```

Correction de l'activité

Enoncé de l'exercice 2:

1. Demander de proposer un algorithme qui permet de saisir une matrice

```
Saisie_matrice
M: tableau[1..100,1..100] d'entiers
N,P : entiers
Debut
Lire (N)
Lire (P) // Il faut faire des tests sur les valeurs de N et P saisies
        Pour i de 1 à N faire
                Pour j de 1 à P faire
                        Lire( M[ i , j ] )
                Fpour
        Fpour
Fin
```

Correction de l'activité

2. Demander de proposer un algorithme qui permet de permuter deux variables

```
Permutter
Var
V,fin, deb: entier
Début
   Lire(v)
   Lire(fin)
   v := deb
   deb := fin
   fin := v
Fin
```

Correction de l'activité

3. Demander d'écrire un algorithme permettant un affichage accordéon de la matrice

```
Affiche_accordéon
Var
M: tableau[1..100,1..100] d'entiers
N,P : entiers
 i , j, deb, fin, v : entier
Debut
Lire (N)
Lire (P)
            Pour  i  de 1 à N  faire
                        Pour  j  de  1  à  P  faire
                                    Lire( M[ i , j ] )
                        Fpour
            Fpour
```

```
                deb := 1
                fin := P
                Pour  i  de 1 à N  faire
                            Pour  j  de  deb  à  fin  faire
                                        écrire( M[ i , j ] )
                            Fpour
                            v := deb
                            deb := fin
                            fin := v
                Fpour
Fin
```

Activité4

Fichiers (QCM)

Description de l'activité

Compétences visées
• Maitrise des notions acquises dans le cours Fichiers
Recommandations clés
• Révision convenable du cours

Description de l'activité

1. Quelle affirmation concernant l'accès à un fichier séquentiel est correcte?

A. L'accès est direct

B. l'accès se fait en utilisant un rang

C. l'accès à l'article de rang n, il est nécessaire de parcourir les (n-1) articles précédents.

2. La fonction FDL(NomFichier) permet de

A. Fermer un fichier

B. Vérifier si le pointeur a atteint la fin d'une ligne

C. Lire une ligne du fichier

Description de l’activité

3. Un fichier textuel ne peut être traité que caractère par caractère

A. Vrai

B. Faux

4. Les fichiers sont stockés en mémoire secondaire comme les variable

A. Vrai

B. Faux

Correction de l'activité

1. Quelle affirmation concernant l'accès à un fichier séquentiel est correcte?

C. l'accès à l'article de rang n, il est nécessaire de parcourir les (n-1) articles précédents.

2. La fonction FDL(NomFichier) permet de

B. Vérifier si le pointeur a atteint la fin d'une ligne

3. Un fichier textuel ne peut être traité que caractère par caractère

A. Vrai

4. Les fichiers sont stockés en mémoire secondaire comme les variable

B. Faux

Activité5

Procédures et fonctions

Description de l'activité

Compétences visées

- Maitrise de la syntaxe d'écriture d'une procédure et d'une fonction
- Différencier entre procédure et fonction
- Maitriser l'appel d'une procédure ou une fonction dans un programme principal

Recommandations clés

- Maitrise de toute la partie précédente du cours

Etapes

1. Demander l'écrire une fonction/procédure permettant de retourner un entier positif saisi au clavier
2. Demander une définition d'une fonction permettant de retourner la somme des diviseurs d'un entier positif
3. Demander l'écriture de l'algorithme principal faisant appel aux procédures et fonctions définies précédemment

Description de l'activité

Enoncé de l'exercice1

Ecrire un algorithme qui permet de lire deux entiers X et Y strictement positifs, affiche "X et Y sont AMIS" ou "X et Y ne sont pas AMIS".

X et Y sont dits nombres AMIS si SX = Y et SY = X, avec:

- SX est la somme des diviseurs de X excepté lui-même,

- SY est la somme des diviseurs de Y excepté lui-même.

1. **Demander l'écrire une fonction/procédure permettant de retourner un entier positif saisi au clavier**
2. **Demander une définition d'une fonction permettant de retourner la somme des diviseurs d'un entier positif**

Description de l’activité

3. Demander l’écriture de l’algorithme principal faisant appel aux procédures et fonctions définies précédemment

Correction de l'activité

Enoncé de l'exercice1

1. **Demander l'écrire une fonction/procédure permettant de retourner un entier positif saisi au clavier**

```
lire_ent( ) : entier
Var  x : entier
Début
        Répéter
                Lire( x )
        Jusqu'à  x > 0
        lire_ent := x
Fin
```

2. **Demander une définition d'une fonction permettant de retourner la somme des diviseurs d'un entier positif**

Correction de l'activité

```
Somme_div( X : entier ) : entier
Var   S, I, M : entier
Debut
            S := 0
            M := X div 2
            Pour  I de 1 à M  faire
                        Si  X mod I = 0
                        Alors  S := S+I
            Finpour
            Somme_div := S
Fin
```

Partie 2: Appel des procédures et fonctions dans un programme principal

3. Demander l'écriture de l'algorithme principal faisant appel aux procédures et fonctions définies précédemment

Correction de l'activité

```
Amis
Var  x, y, sx, sy : entier
Début
        x := lire_ent( )
        y := lire_ent( )
        sx := Somme_div( x )
        sy := Somme_div( y )
        Si  sx = y  ET  sy = x
        Alors  écrire(''Les 2 entiers sont amis'')
        Sinon  écrire(''Les 2 entiers ne sont pas amis'')
        Fsi
Fin
```

Activité6

Récursivité

Description de l'activité

Compétences visées

- Maitrise de toutes les étapes d'écriture d'une solution récursive (Condition d'arrêt et formule de récursivité)

Recommandations clés

- Maitrise de toute la partie précédente du cours

Etapes

1. Questionner sur la (les) condition(s) d'arrêt
2. Demander la formule de récurrence
3. Demander la définition d'une fonction récursive

Description de l'activité

Enoncé de l'exercice 1:

Ecrire une fonction récursive qui teste si une suite d'entiers stockée dans un tableau est croissante ou non.

1. **Questionner sur la (les) condition(s) d'arrêt**
2. **Demander la formule de récurrence**
3. **Demander la définition de la fonction récursive Croissante**

Description de l'activité

Enoncé de l'exercice2:

Ecrire une fonction récursive qui teste si une suite d'entiers stockée dans un tableau est croissante ou non.

1. **Questionner sur la (les) condition(s) d'arrêt**
2. **Demander la formule de récurrence**
3. **Demander la définition de la fonction récursive recherche_Dicho**

Correction de l'activité

Partie1: Vérification si un tableau est trié ou non

1. Questionner sur la (les) condition(s) d'arrêt

Cas particulier: suite vide (deb>fin) ou suite contenant un seul élément (deb=fin), le résultat est évident la suite est croissante.

2. Demander la formule de récurrence

Suite Edeb , Edeb+1 , …., Efin est croissante si et seulement si :

Edeb $\leq$ Edeb+1 et Edeb+1 , …., Efin est croissante

3. Demander la définition de la fonction récursive Croissante

Correction de l'activité

```
Croissante ( T : tableau[1..Max] de entier ; deb, fin : entier) : booléen
Début
    Si  deb ≥ fin  alors  Croissante := vrai
    sinon  Si  T[deb] > T[deb+1]
                Alors Croissante := faux
                Sinon  Croissante := Croissante ( T, deb+1, fin)
                fsi
    fsi
Fin
```

Correction de l'activité

Enoncé de l'exercice 2:

1. **Questionner sur la (les) condition(s) d'arrêt**

2 cas particuliers :

- suite vide (deb>fin) l'élément n'existe pas dans le tableau

- l'élément du milieu du tableau est x.

2. **Demander la formule de récurrence**

Selon la comparaison entre le milieu du tableau et x on oriente la recherche vers la partie à gauche ou la partie à droite (Si x < T[m] on effectue une recherche à gauche sinon à droite)

Correction de l'activité

3. Demander la définition de la fonction récursive recherche_Dicho

```
Recherche_dicho (T: tableau[1..Max] de entier ; deb, fin, x : entier)
Var  m : entier
Début
si deb > fin  alors Recherche_dicho := 0
sinon       m := (deb+fin) div 2
                        si  T[m] = x  alors  Recherche_dicho := m
                        sinon       si  x < T[m]
                                    alors  Recherche_dicho := Recherche_dicho (T, deb, m-1, x)
                                    sinon Recherche_dicho := Recherche_dicho (T, m+1, fin, x)
                                    fsi
                        fsi
            fsi
Fin
```