Deug Mias-2, 2000-20001
Informatique
Institut Galilée

TD-1

Exercice 1 : passages de parametres

Simuler l'algorithme suivant :

```
Algorithme Alias
Var I: entier
Procédure IncrémenterDeI (var K:entier ; var L:entier)
          debut
          K <- K+I
          L <- L+1
          fin
debut
I<-1
IncrémenterDeI(I,I)
writeln(I)
fin
```

Correction Ex-1

{I=1}
{IncrementerDeI(I,I)}
{K est I, L est I (=1) /* passage par adresse : 'est' */
{K = 1+1=2}
{L = 2+1=3}
{}
{modifications : I = 3}
{affiche '3'}

Le jeu consiste à insister sur quelques implications simples du passage par variable.

Des noms de variables sont des alias s'ils sont différents et possèdent la même adresse. Par exemple, des éléments de tableau T[i] et T[j] sont des alias si i=j. Le passage de paramètres par adresse crée par définition un alias entre le paramètre formel et le paramètre effectif. Il permet aussi de créer des alias dans les conditions suivantes :

1- Une variable globale peut être visible dans le corps d'une procédure et désignée sous un autre nom par un paramètre formel.

2- Deux paramètres formels distincts peuvent désigner le même paramètre effectif.

L'algorithme de l'exercice 1 illustre les deux possibilités.

Exercice 2 passages de parametres- encore

Le phénomène d'interférence (modification de la valeur d'une variable qui change également la valeur des ses 'alias', cf. exercice 2) peut conduire à des erreurs de programmation. Soit l'algorithme :

```
Algorithme Factorielle?
Var  X,Y : entier
Procédure Fact (var N : entier ; var F : entier)
          Var  K : entier
          debut
          K <- 0
          F <- 1
          TantQue K≠N faire
               K <- K+1
               F <- K*F
          Fintq
          fin

debut
readln(X)
Fact(X,Y)
writeln(Y)
Fact(Y,Y)
writeln(Y)
fin
```

1) Quelles sont les valeurs affichées à l'écran quand la valeur de X lue est 3 ?

2) Comment modifier l'algorithme ci-dessus pour que les valeurs affichées à l'écran soient successivement 6 et 720 (=6!).

Correction ex-2

1) L'état de l'écran à la fin de l'exécution de l'algorithme est le suivant :

6

1

{lecture X=3}
{Fact(X,Y)}

{N est X (=3), F est Y }
{K=0, F=1}
{0 ≠ 3}
{K=1}
{F=1*1}
{1≠3}
{K=2}
{F=2*1=2}
{2≠3}
{K=3}
{F=3*2=6}
{3≠3 Faux}

{}
{Modifications X=3, Y=6}
{affiche 6}

{Fact(Y,Y)}

{N est Y (=6), F est Y }
{K=0, F=1 /*et donc N=1*/}
{0 ≠ 1}
{K=1}
{F=1*1}
{1≠1 Faux}

{}
{Modifications X=3, Y=1}

{affiche 1}

L'interférence entre les paramètres N et F peut être évitée en passant N par valeur.

```
Procédure Fact ( N : entier* ; var F : entier*)
```

{lecture X=3}
{Fact(X,Y)}

{N = 3 , F est Y } /* N est passé par valeur, il s'agit d'une variable locale */
{K=0, F=1}
{0 ≠ 3}
{K=1}
{F=1*1}
{1≠3}
{K=2}
{F=2*1=2}
{2≠3}
{K=3}
{F=3*2=6}
{3≠3 Faux}

{}
{Modifications Y=6}
{affiche 6}

{Fact(Y,Y)}

{N =6 , F est Y }
{K=0, F=1 /*mais ici N est locale et = 6 et ne sera pas confondue avec F (alias de Y)*/}
{0≠6}
{K=1}
{F=1*1}
{1≠6}
{K=2}
{F=2*1=2}
{2≠6}
{K=3}
{F=3*2=6}
……
{5≠6}
{K=6}
{F=6*120=720}
{6≠6 Faux}

{}

{modifications Y = 720}

{affiche 720}

Après modification, l'état de l'écran à la fin de l'exécution de l'algorithme est le suivant :

6

720

Exercice -3

Nous disposons d'un tableau T de N éléments. Un élément est constitué d'un champ Nom (une chaîne) et d'un champ Couleur (un caractère 'B' ou 'R'). Le problème est de réarranger les éléments de T de manière a ce que les éléments de couleur 'B' précèdent les éléments de couleur 'R'.

Exemple:

(Toto,B)(Titi,R),(Tutu,B)(Tata,R), (Tyty,R), (Tete,B)

est réarrangé par exemple en:

(Toto,B),(Tete,B),(Tutu,B) , (Tata,R), (Tyty,R),(Titi,R)

Il faut de plus respecter les contraintes suivantes: 1) On n'utilisera pas de tableau auxilliaire et on procedera uniquement par echanges de deux éléments, 2) on ne regardera pas deux fois la couleur d’un élément.

Il est suggéré d'utiliser la méthode suivante: On considère trois zones successives dans le tableau: la zone des 'B', la zone des éléments de couleur non identifiée, la zone des 'R'. Au début les zones 'B' et 'R' sont vides. A la fin la zone des éléments non identifiés est vide, et le problème est résolu.

Pour cela on notera b, la position dans le tableau du premier élémént suivant la zone des 'B', et r la position dans le tableau du premier élément précédant la zone des 'R'.

On fera ainsi pas à pas augmenter b ou decroître r, et eventuellement échanger les éléments d'indice b et r jusqu'à ce que le problème soit résolu.

Exemple: au début

(Toto,B)	(Titi,R),	(Tutu,B)	(Tata,R),	(Tyty,R),	(Tete,B)
b=1					r=6

à la fin

(Toto,B),	(Tete,B),	(Tutu,B) ,	(Tata,R),	(Tyty,R),	(Titi,R)
		r=3	b=4		

1) Ecrire une procédure Ranger(T) réarrangeant le tableau T.

Correction ex-3-1

```
Const N = ?
Type Element =Enregistrement
      Nom: Chaîne
      Couleur: Caractère
FinEnregistrement
Type Tab = Tableau [1..N] de Element

Procédure Ranger(Var T: Tab)
    Var b,r :entier
    {}
    début
    b <--1
    r<--N
    TantQue b ≤ r Faire
          Si T[b].Couleur ='B' Alors
               b <-- b+1
          Sinon {en b la couleur est 'R'}
               Echange(T,b,r)
               r<-- r-1
          Finsi
    Fintq
    {la zone Inconnue est Vide car les zones R et B sont en
    contact}
```

```
    fin
```

```
NB :
Procédure Echange(var T : tab ; i: entier ; j : entier)
    Var E :Elément
    Début
    E<- T[i]
    T[i]<-T[j]
    T[j]<- E
    fin
```

2) Ecrire une procédure Ranger3(T) traitant le même problème dans le cas où il y a 3 couleurs 'B', 'R', 'V'. Il faudra distinguer 4 zones et utiliser un indice supplementaire v représentant la position du premier élément précédant la zone des 'V'. Par exemple en cours d'exécution on peut rencontrer la configuration suivante:

< B >	< ? >		< R >		< V >
(Toto,B)	(Titi,R),	(Tutu,B)	(Tata,R),	(Tyty,R),	(Tete,V)
	b=2	r=3		v=5	

Correction ex-3-2

```
Procédure Ranger3(Var T: Tab)
    Var b,r,v :entier*
    {}
    début
    b <-1
    r <-N
    v <-N
    TantQue b ≤ r Faire {zone inconnue non vide}
          Si T[b].Couleur ='B' Alors
               b <-- b+1 {zone inconnue diminue}
          Sinon {en b la couleur est 'R' ou 'V'}
               Si T[b].Couleur ='R' Alors
                    Echange(T,b,r)
                    r<-- r-1
               Sinon {en b la couleur est 'V'}
                    Echange(T,b,v) {en v la couleur est 'V'}
                    v <- v-1
                    SI r= v+1 Alors {zone rouge vide}
                         r<-r-1 {zone inconnue diminue}
                    Sinon {l'élément échangé était rouge
donc il y a un rouge en b}
                         Echange(T,b,r)
                         r<-- r-1
                    Finsi
```

```
            Finsi
        Finsi
Fintq
{la zone Inconnue est Vide car les zones R et B sont en
contact, et la boucle se termine car a chaque iteration
soit b augmente, soit r diminue}}
fin
```