

corrigé TD6TEC6111991 algorithmes paramétrés ;définitions et appels

EX1-a) l'algorithme <premier> appelle la procédure <permuter> avec les paramètres I et J transmis par valeur et restent donc locaux à la procédure <permuter> afficher (I, J) donne le résultat suivant

1 2

b) si on remplace l'en-tête de la procédure <permuter> par procédure permuter (var I, AUX : entier) la transmission des paramètres se fait donc par adresse et tout traitement fait sur ces paramètres a le même effet sur la procédure appelante <premier> d'où le résultat affiché suivant :

2 1

c) les modifications à apporter sont :

supprimer l'instruction $J \leftarrow 2$ et faire seulement $AUX \leftarrow I ; I \leftarrow J ; J \leftarrow AUX ;$ et déclarer procédure permuter(var I, J : entier).

EX2-

- a) la procédure calcule la somme de 4 valeurs A, B, C, D dans S.
- b) pour faire la somme de trois (3) variables X, Y, Z dans la variable TOTAL on utilise la procédure comme suit :
- c) l'effet de l'appel SOMME1(K, L, M, N, K) donne

SOMME1(0, X, Y, Z, TOTAL)

$K \leftarrow 0 ;$
 $K \leftarrow K+K ;$
 $K \leftarrow K+L ;$
 $K \leftarrow K+M ;$
 $K \leftarrow K+N$

Cet appel calcule dans K la somme de L, M, N.

Pour pouvoir faire la somme de K, L, M, et N dans K nous proposons la séquence suivante :

$J \leftarrow K ;$ SOMME1(J, L, M, N, K).

d) SOMME2(X, Y, Z, T)

$X \leftarrow X+Y ;$
 $X \leftarrow X+Z ;$
 $X \leftarrow X+T$

e) pour calculer la somme de six(6) variables nous utilisons les deux procédures SOMME1 et SOMME2 comme suit :

SOMME2(N1, N2, N3, N4) ; SOMME1(N1, N5, N6, 0, CUMUL).

f) on peut quadrupler la valeur de X par SOMME2(X, X, X, X) ou bien par SOMME1(X, X, X, X, S).

EX3-

procedure <lecture d'une ligne>

ligne : packed array[1..80] of char ;N : integer;
begin I:=1;

```
for n := 1 to 100 do begin read(ligne); (*on suppose avoir 100 lignes de 80 caractères ,à
traiter une par une *)
```

```
    IL := 1
    repeat
    examen; if com = 0 then reconnait (* com est l'indicateur de
commentaires si 0 il s'agit d'un mot à identifier *)
    until IL > 80; (*avec la procédure
reconnait*)
    end ;
```

```
end ;
```

```
procedure <examen> (* repère chacun des mots de la ligne ,ceux-ci étant séparés par
des blancs ou commentaires*)
```

```
I := IL ; longueurmot:=0 ; J := 1 ;com := 0 ;
```

```
while(ligne[I] <> ' ' and ligne[I] <> '(' do begin
```

```
    motexamine[I] := ligne[I];
```

```
    J := J+1 ; I := I+1;
```

```
end;
```

```
if ligne[I] = '(' then if ligne[I+1] = '*' then begin (* elimination des
commentaries*)
```

```
    I := I+2; etiq: while ligne[I] <> '*' do
```

```
I := I+1;
```

```
    I := I+1;
```

```
    if ligne[I] = ')' then com := 1 else goto
```

```
etiq
```

```
end
```

```
else longueurmot := J; IL := I+1 (* mot prochain*)
```

```
procedure <reconnait1> (* reconnaît si motexamine est un identificateur ou non *)
```

```
begin
```

```
I := 2 ; while ( motexamine[I] in ['A'..'Z', 'a'..'z',','..'9'] and I < longueurmot) do I :=
I+1;
```

```
if I < longueurmot then indice := 0 (*sortir du while sous la condition I<longuermot
indique un symbole illégal *)
```

```
    else indice := 1;
```

```
end;
```

```
procedure< reconnait2> (* reconnaît si motexamine est un numérique ou non *)
```

```
begin
```

```
case motexamine[1] of
```

```
    '+', '-' : begin I := 2;
```

```
    et2: while ( motexamine[I] in ['0'..'9'] and I < longueurmot ) do I:= I+1;
```

```
    if I < longueurmot then if motexamine[i] = '.' then begin
```

```
        I := I+1;
```

```
        while motexamine[I]
```

```
in ['0'..'9'] do I := I+1
```

```
if I < longueurmot
```

```
then indice :=0
```

```

indice := 2
else
end
else indice := 2;
end;
'0'..'9' : if examine[2] = '.' then begin I := 3
longueurmot do I := I+1
while( motexamine[I] in ['0'..'9'] and I <
indice := 2
if I < longueurmot then indice := 0 else
end
else goto et2
end;

```

procedure <reconnait>

```

const: reserve = ['begin', 'end', 'var', 'read', .....];
      bool = ['true', 'false'];
      carac = ['A'..'Z', 'a'..'z', '0'..'9']
      operateur = ['+', '-', '/', '*', 'div', 'mod', ...]
begin

```

```

if motexamine in reserve then indice := 5
else if motexamine in bool then indice := 3
else if motexamine in carac then
indice := 4
else if motexamine in operateur then indice := 6

```

```

case motexamine[1] of
reconnait2
reconnait1
end ;

```

la syntaxe des mots à traiter par les procédures est définie par les diagrammes de Conway suivants :

