Algorithmique et structures de données 2017-18

Contenu du cours 5 du 19.10.2017

1. Tri à bulles

- Algorithme appliqué au tableau

4	**7**	**6**	**2**	**1**

Principe : on parcours le tableau depuis et on permute les éléments successifs s'il s sont dans le désordre

4	**7**	6	2	1
4	**7**	**6**	2	1
4	6	**7**	**2**	1
4	6	2	**7**	**1**
4	6	2	1	7

A la fin de cette traversée, le plus grand élément se trouve en dernière position.

- On applique à nouveau ce principe, mais sur le tableau allant de la 1ère à l'avant-dernière case du tableau

4	**6**	2	1	7
4	**6**	**2**	1	7
4	2	**6**	**1**	7
4	2	1	6	7

- Et ainsi de suite …
- En 4 étapes nécessitant 4, puis 3, puis 2, et finalement 1, opérations de comparaison-échange, on obtient un tableau trié. Donc en 4+3+2+1 = 5·4/2 = 10 opérations, on a un tableau trié. Plus généralement, un tableau à N éléments se trie en N(N-1)/2 opérations avec le tri à bulles.
- Pseudo-code

```
type T_Stat is array(Positive range <>) of Float;
procedure Echange(A,B : in out Float) is …;
procedure Max_En_Fin(X : in out T_Stat) is …;
procedure Tri_Bulles(X : in out T_Stat) is
begin
   for J in reverse X'first+1..X'last loop
      Max_En_Fin(X(X'first..J));
   end loop;
end Tri_Bulles;
```

2. Notion de visibilité des variables

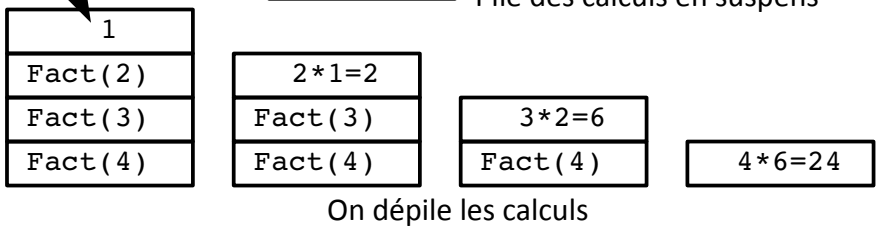
```
procedure Visible is
   N : Integer := 1;
   procedure Quoi(N : out Integer) is
   begin
      -- Par défaut N désigne le N local
      -- Visible.Quoi.N désigne le paramètre N de la fonction
      -- Visible.N désigne la variable globale
      -- de la procédure principale
      Put_Line("Visible.Quoi.N =" & Integer'Image(N));
      Put_Line("Visible.N =" & Integer'Image(Visible.N));
      N := N+2; -- ici N est donc le paramètre de la fonction
      Put_Line("Visible.Quoi.N =" & Integer'Image(N));
      Put_Line("Visible.N =" & Integer'Image(Visible.N));
   end Quoi;
begin
   Quoi(N);
   Put_Line("N =" & Integer'Image(N));
end Visible;
```

3. Récursivité

- Exemple de la factorielle : n ! = n·(n-1)·(n-2)·… ·3·2·1 = n·(n-1) !

  Donc Fact(n) = n·Fact(n-1) (récursivité) et Fact(1) = 1 (condition d'arrêt)

```
procedure Recursif is
   function Fact(N : Positive) return Positive is
   begin
      if N > 1 then
         return N*Fact(N-1);
      else
         return 1;
      end if;
   end Fact;
   F : Positive;
begin
   F := Fact(4);
end Recursif;
```



Pile des calculs en suspens

On dépile les calculs

- Exemple du PPCM

```
function PPCM(N1,M1,N,M : Positive) return Positive is
begin
   if N1 = M1 then
      return N1;
   elsif N1 > M1 then
      return PPCM(N1,M1+M);
   else
      return PPCM(N1+N,M1);
   end if;
end PPCM;
```

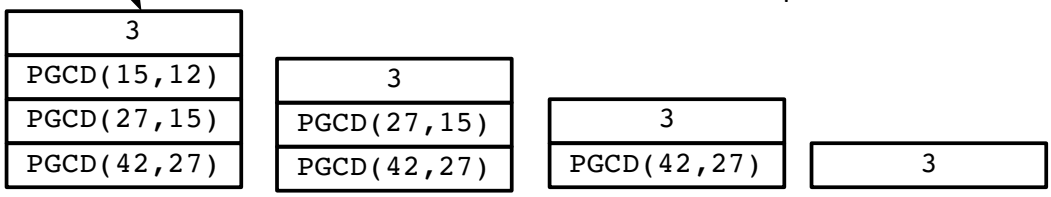
- Exemple du PGCD

  Algorithme d'Euclide pour le PGCD de 42 et 27

```
42 = 27 * 1 + 15
27 = 15 * 1 + 12
15 = 12 * 1 + 3
12 =  3 * 4 + 0
```

PGCD(42,27)=PGCD(27,15)=PGCD(15,12)=PGCD(12,3)=3

```
function PGCD(N,M : Positive) return Positive is
begin
   if N rem M > 0 then
      return PGCD(M,N rem M);
   else
      return M;
   end if;
end PGCD;
PGCD(42,27);
```

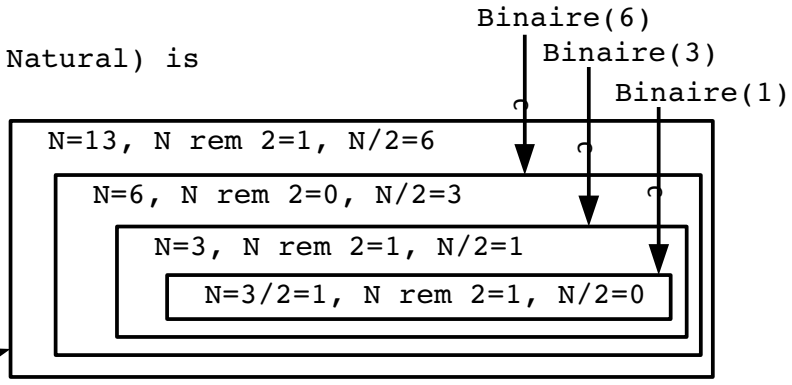


Pile des calculs en suspens

On dépile les calculs

- Exemple de l'écriture binaire

```
procedure Binaire(N : Natural) is
begin
   Put(N rem 2);
   if N/2 /= 0 then
      Binaire(N/2);
   else
      New_Line;
   end if;
   -- Put(N rem 2);
end Binaire;
Binaire(13); -- affiche 1 0 1 1 puis un retour à la ligne
```



$2^0\ 2^1\ 2^2\ 2^3$

Que se passe-t-il si on décommente le deuxième `Put` ?

- Exemple de calcul de la puissance a^b avec $a, b \geq 0$

```
function Puissance(A : Integer; B : Natural) return Integer is
begin
   if B = 0 then
      return 1;
   elsif b mod 2 =0 then
      return Puissance(A,B/2)*Puissance(A,B/2);
   else
      return Puissance(A,B-1)*A;
   end if;
end Puissance;
```

$a^b = a^{b/2} \cdot a^{b/2}$ si b est pair,

$a^b = a^{b-1} \cdot a$ si b est impair

$a^0 = 1, 0^0 = 1$