Les Procédures

Les procédures

- Une procédure est une fonction qui ne retourne pas de valeur: elle ne calcule pas de résultat ou elle calcule plusieurs résultats à la fois
- Exemple:
  - Afficher la table de multiplication d'un entier saisie au clavier
  - Permuter les valeurs de deux variables
  - Afficher un triangle:

```
1
12
123
1234
12345
123456
```

Déclaration d’une procédure

Procedure nom_procedure (paramètres et leurs types)

Début

Instructions constituant le corps de la procédure

Fin

Exemple

```
Procedure table_multiplication (n:entier)
        Variables: i:entier
    Début
        Pour i allant de 1 à 10 faire
            Ecrire (n,"*", i , "=", n*i)
        FinPour
    Fin
```

L’appel d’une procédure

- L’appel de la procédure se fait dans le programme principal
- Contrairement à l’appel d’une fonction, on ne peut pas affecter la procédure à une variable ou l’afficher

→ **L’appel d’une procédure est une instruction autonome:**

Exemple complet

```
Algorithme afficher_table

Variables: x: entier

Procedure table_multiplication (n:entier)

        Variables: i:entier

    Début

        Pour i allant de 1 à 10 faire

            Ecrire (n,"*", i , "=", n*i)

        FinPour

    Fin

Début

Ecrire (" Entrer un réel   ")

Lire (x)

table_multiplication(x)

Fin
```

Paramètres d'une procédure

- Les paramètres servent à échanger des données entre le programme principale et la procédure appelée.
- Les paramètres placés dans la déclaration d'une procédure sont appelés **paramètres formels**. Ces paramètres peuvent prendre toutes les valeurs possibles mais ils sont abstraits (n'existent pas réellement)
- Les paramètres placés dans l'appel d'une procédure sont appelés **paramètres effectifs**. ils contiennent les valeurs pour effectuer le traitement
- Le nombre de paramètres effectifs doit être égal au nombre de paramètres formels. L'ordre et le type des paramètres doivent correspondre.

Les types de paramètres (1)

- Les paramètres d'une procédures sont de 3 catégories:
  - **Paramètres d'entrée**: ils correspondent aux valeurs que vous souhaitez transmettre à la procédure. Toute modification des paramètres à l'intérieur de la procédure n'aura aucune influence sur les variables d'origine.
  - Les paramètres d'entrée sont précédés par **« E »**

Les types de paramètres (2)

- **Paramètres de sortie**: ils correspondent aux valeurs retournés par la procédure au programme principal. Toute modification d'un paramètre par le code de la procédure entraine automatiquement la modification du paramètre d'origine
- Les paramètres de sortie sont précédés par **« S »**

Les types de paramètres

- **Paramètres d'entrée/Sortie**: les paramètres en entrée sont traités puis restitués en sortie
- Les paramètres d'Entrée/sortie sont précédés par **« ES »**

En langages de programmation

- On parle de:
  - Passage par valeur
  - Passage par adresse (ou par référence)

Algorithmique	Langage de Programmation
Paramètre d'entrée	Passage par valeur
Paramètre de sortie Paramètre d'entrée-sortie	Passage par adresse (par référence)

Exemple

```
Algorithme test
Variables m,n:entier
Procedure incrementer(E x:entier, E/S y: entier)
    Début
        x ←x+1
        y←y+1
        Ecrire( x,y)
    Fin
/**********************************/
Début
    n ←3
    m←3
    Incrementer(n,m)
    Ecrire(n,m)
Fin
```

4 4
3 4

Exercice 1

Ecrire un algorithme qui permet d’échanger les valeurs de a et b si a>b

Solution exercice 1

```
Algorithme Permutation
Variables a,b:entier
Procedure permuter(E/S x:entier, E/S y: entier)
        Variables: tmp:entier
    Début
        tmp ←x
        x←y
        y←tmp
     Fin
```

```
/**********************************/
Début
    Ecrire ("Entrez deux entiers")
    Lire(a,b)
    Si (a>b) alors
        permuter(a,b)
    Finsi
    Ecrire(a,b)
Fin
```

Exercice 2

Ecrire une procédure qui calcule la somme et le produit de 2 entier a et b

Solution exercice 2

```
Algorithme somme_produit
Variables a,b,S,P:entier
Procedure somme_produit(E x:entier, E y: entier, S Som entier, S Prod: entier )
        Début
                som←x+y
                Prod←x*y
        Fin
Début
        Ecrire ("Entrez deux entiers")
        Lire(a,b)
        somme_produit(a,b,S,P)
        Ecrire (la somme est : , S , Le produit est ,P)
Fin
```

Conclusion

Toute fonction peut être transformer en procédure en ajoutant un paramètre de sortie

3 - Variables locales et globales

- Les **variables locales** et les **variables globales** se distinguent par ce qu'on appelle leur **portée** (leur "champ de définition", leur "durée de vie")
- Une **variable locale** n'est connue qu'à l'intérieur de la fonction(procédure) où elle a été définie. Elle est créée à l'appel de la fonction (procédure) et détruite à la fin de son exécution
- Une **variable globale** est déclarée dans la partie déclaration du programme principal, elle est connue par toutes les procédures et les fonctions.

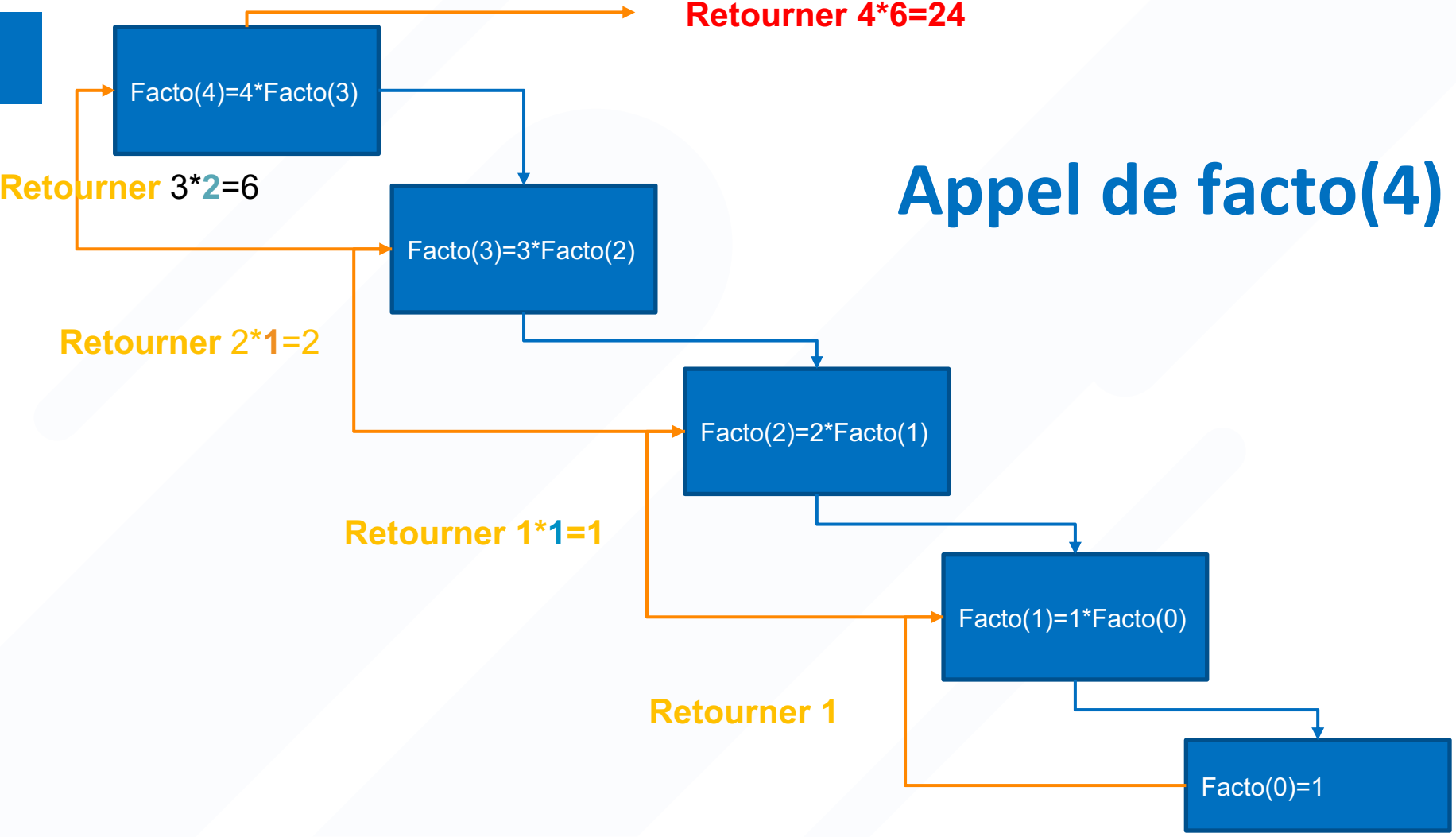
Les fonctions récursives

Récursivité

- On appelle fonction (procédure) **récursive** toute fonction (procédure) qui s'appelle elle même.
- Exemple : la fonction factorielle : n!=n*(n-1)!

```
FONCTION factorielle (n:entier):entier
 DEBUT
    SI (n=0) ALORS
        RETOURNER (1)
    SINON
    RETOURNER (n*factorielle(n-1))
 FIN
```

Appel de facto(4)



Remarques importantes

- La définition d'une fonction récursive nécessite la définition d'une condition d'arrêt
- Le processus récursif remplace le processus itératif
- Toutes fonction récursive peut être transformé en fonction itérative
- Toutes les fonctions ne sont pas récursives !!!!

Exemple 2 : suite de Fibonacci

$$F_0 = 0$$
$$F_1 = 1$$
$$F_i = F_{i-1} + F_{i-2} \qquad \text{pour i > 1}$$

```
FONCTION fibo (n:entier):entier
 DEBUT
    SI (n=0) ou (n=1) ALORS
        RETOURNER (n)
    SINON
    RETOURNER (fibo(n-1)+fibo(n-2))
 FIN
```