Algorithmes et programmation II : La récursivité

Souheib Baarir[1]

[1]Université Paris Ouest Nanterre La Défense.
Laboratoire d'informatique de Paris 6.
Souheib.Baarir@u-paris10.fr

Licence Mia - 2010/2011

Grandes lignes du cours

Introduction sur la récursivité

Récursivité sur les nombres

C'est quoi ?

Définition

Une fonction récursive est une fonction qui s'appelle elle-même.

C'est quoi ?

Définition

Une fonction récursive est une fonction qui s'appelle elle-même.

Exemple :

```
int sommeN(int n){
  if(n == 1)
    return 1;
  else
    return (n + sommeN(n-1));
}
```

Principe de la récursivité

- **Décomposer le problème en un problème plus simple**
  $\Rightarrow$ réduire la taille du problème considéré.

- **Pour la récursion sur des entiers** : la taille du problème est définie par un entier, on réduit la valeur de cet entier à chaque appel récursif.

- **Pour la récursion sur les tableaux** :
  - Soit on considère la taille du tableau, on réduit la taille du tableau considéré à chaque appel récursif
  - Ou bien on utilise un ou des indices qui varient à chaque appel pour tendre vers la condition d'arrêt (dépendant des valeurs des indices).

Définir une fonction récursive

- Une fonction récursive est définie par :
  - au moins un *cas de base* et,
  - au moins un *cas général*.

Définir une fonction récursive

- Une fonction récursive est définie par :
  - au moins un *cas de base* et,
  - au moins un *cas général*.
- **Cas de base** : on décrit les cas pour lesquels le résultat de la fonction est simple à calculer : la valeur retournée par la fonction est directement définie.

Définir une fonction récursive

- Une fonction récursive est définie par :
  - au moins un *cas de base* et,
  - au moins un *cas général*.
- **Cas de base** : on décrit les cas pour lesquels le résultat de la fonction est simple à calculer : la valeur retournée par la fonction est directement définie.
- **Cas général** : la fonction est appelée récursivement et le résultat retourné est calculé en utilisant le résultat de l'appel récursif. A chaque appel récursif, la valeur d'au moins un des paramètres (effectifs) de la fonction doit changer.

Définir une fonction récursive

- Une fonction récursive est définie par :
  - au moins un *cas de base* et,
  - au moins un *cas général*.
- **Cas de base** : on décrit les cas pour lesquels le résultat de la fonction est simple à calculer : la valeur retournée par la fonction est directement définie.
- **Cas général** : la fonction est appelée récursivement et le résultat retourné est calculé en utilisant le résultat de l'appel récursif. A chaque appel récursif, la valeur d'au moins un des paramètres (effectifs) de la fonction doit changer.

Attention !

Il faut toujours s'assurer que chaque cas général converge vers un cas de base.

Définir une fonction récursive : exemple

- Somme des n premiers entiers : $\sum_{i=1}^{n} i$

Définir une fonction récursive : exemple

- Somme des n premiers entiers : $\sum_{i=1}^{n} i$
  - Cas général : $\sum_{i=1}^{n} i = n + \sum_{j=1}^{n-1} j$
  - Cas de base : $\sum_{i=1}^{n} i = 1$, pour $n = 1$

Définir une fonction récursive : exemple

- Somme des n premiers entiers : $\sum_{i=1}^{n} i$
  - Cas général : $\sum_{i=1}^{n} i = n + \sum_{j=1}^{n-1} j$
  - Cas de base : $\sum_{i=1}^{n} i = 1$, pour $n = 1$
- Factoriel de n : $n!$

Définir une fonction récursive : exemple

- Somme des n premiers entiers : $\sum_{i=1}^{n} i$
  - Cas général : $\sum_{i=1}^{n} i = n + \sum_{j=1}^{n-1} j$
  - Cas de base : $\sum_{i=1}^{n} i = 1$, pour $n = 1$
- Factoriel de n : $n!$
  - Cas général : $n! = n * (n-1)!$
  - Cas de base : $n! = 1$, pour $n \in \{0, 1\}$

Récursivité sur les nombres

Récursivité sur les nombres : exemple (1/4)

Factoriel de n : $n!$

- Cas général : $n! = n * (n-1)!$
- Cas de base : $n! = 1$, pour $n \in \{0, 1\}$

Récursivité sur les nombres : exemple (1/4)

Factoriel de n : $n!$

- Cas général : $n! = n * (n-1)!$
- Cas de base : $n! = 1$, pour $n \in \{0, 1\}$

```
int factorielle(int n){

  /* Le cas de base */
  if((n == 0) || (n == 1))
    return 1 ;

  /* Le cas général */
  else
    return (n * factorielle(n-1)) ;
}
```

Récursivité sur les nombres : exemple (2/4)

```
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    return (n * factorielle(n-1)) ;
}

int main(int arv,char * arg[]){
  printf("Test factorielle : \n");
  printf("1 -> %d \n ",factorielle(1));
  printf("5 -> %d \n ",factorielle(5));
  printf("9 -> %d \n ",factorielle(9));
  return 0;
}
```

Récursivité sur les nombres : exemple (2/4)

```
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    return (n * factorielle(n-1)) ;
}

int main(int arv,char * arg[]){
  printf("Test factorielle : \n");
  printf("1 -> %d \n ",factorielle(1));
  printf("5 -> %d \n ",factorielle(5));
  printf("9 -> %d \n ",factorielle(9));
  return 0;
}
```

Test factorielle :

$1 \rightarrow 1$

$5 \rightarrow 120$

$9 \rightarrow 362880$

Récursivité sur les nombres : exemple (3/4)

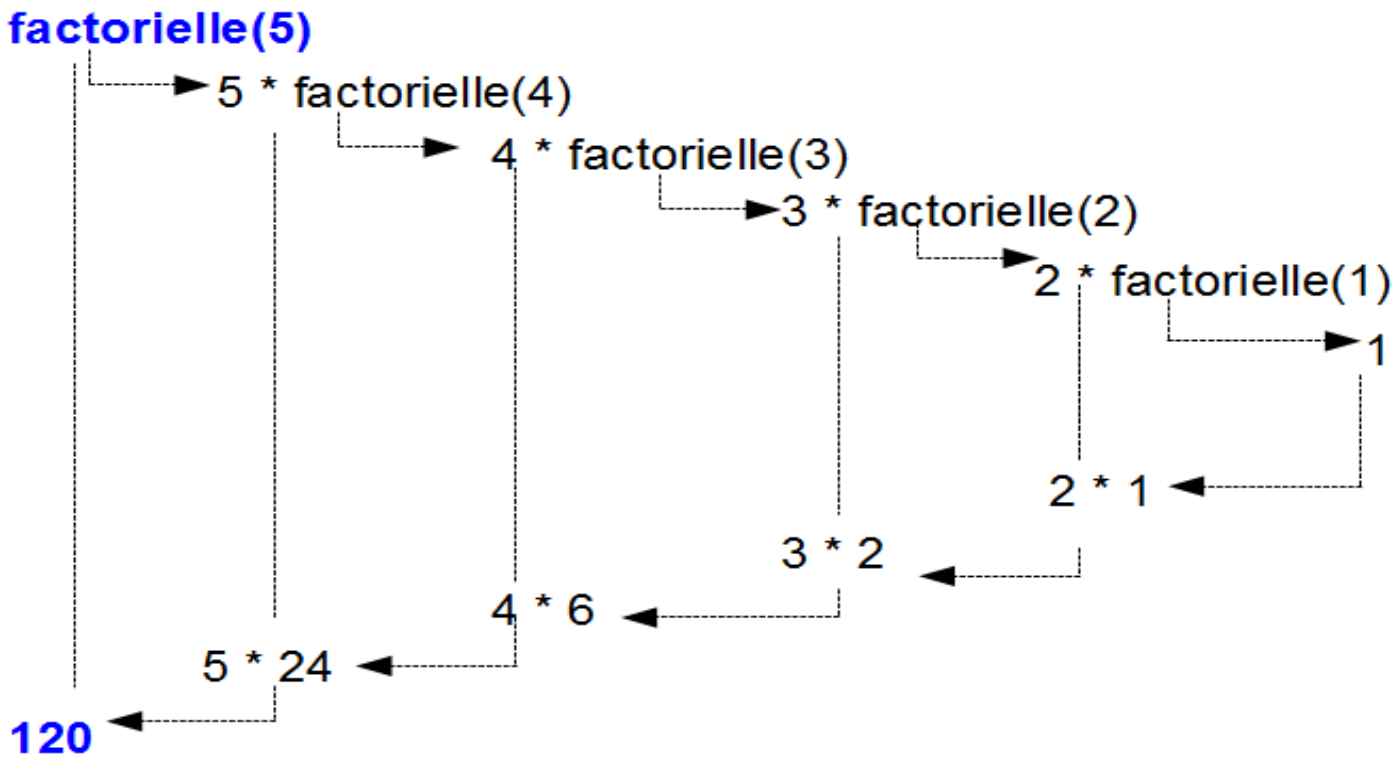
```c
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    return
    (n * factorielle(n-1)) ;
}
```

```c
int main(int arv,
         char * arg[]){
  int res=factorielle(5);
  printf("%d",res) ;
  return 0;
}
```

Récursivité sur les nombres : exemple (3/4)

```c
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    return
    (n * factorielle(n-1)) ;
}
```

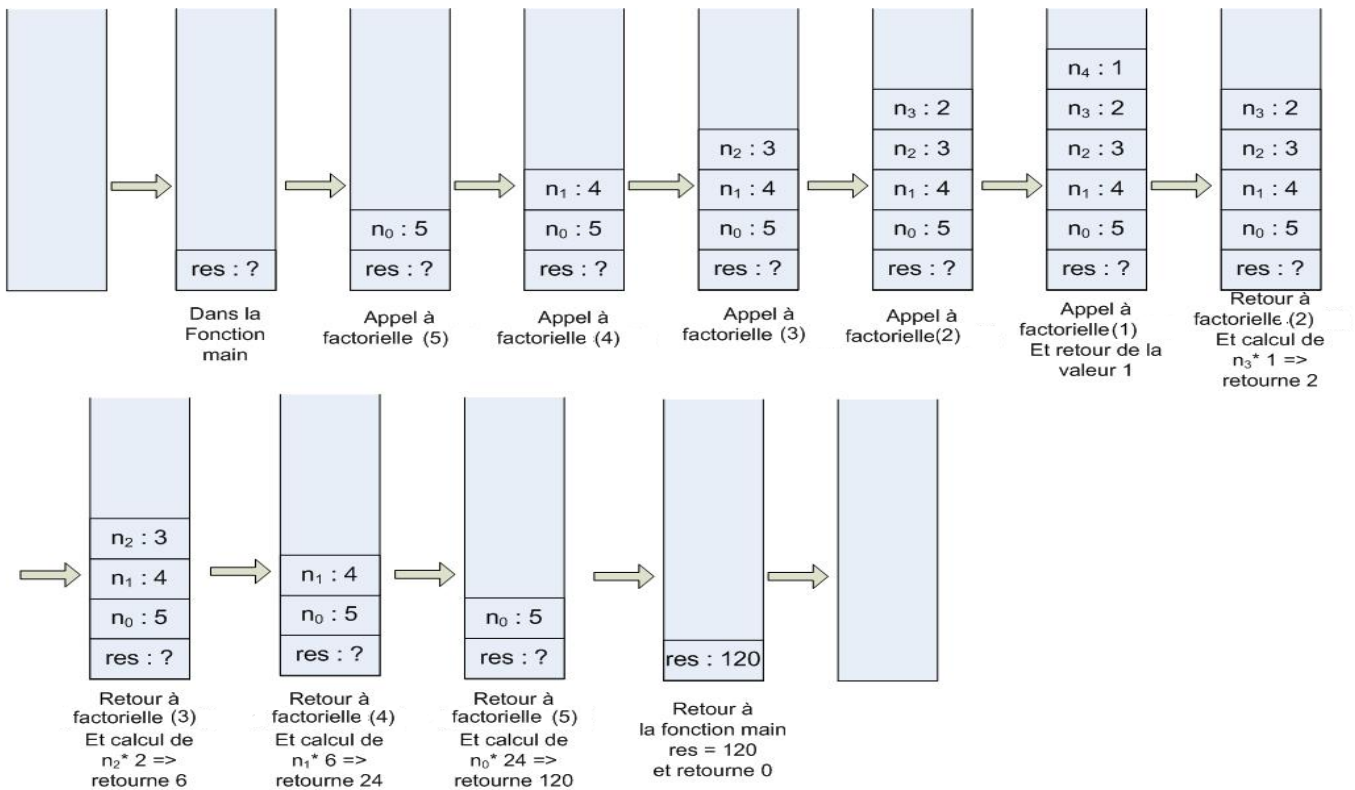
```c
int main(int arv,
         char * arg[]){
  int res=factorielle(5);
  printf("%d",res) ;
  return 0;
}
```



Récursivité sur les nombres : exemple (4/4)

```
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    return
    (n * factorielle(n-1)) ;
}
```

```
int main(int arv,
         char * arg[]){
  int res=factorielle(5);
  printf("%d",res) ;
  return 0;
}
```



Récursivité sur les nombres : cas de base jamais atteint !

```c
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    /* return (n * factorielle(n-1)) */
    return (n * factorielle(n)) ;
}

int main(int arv,char * arg[]){
  printf("Test factorielle : \n");
  printf("1 -> %d \n ",factorielle(1));
  printf("5 -> %d \n ",factorielle(5));
  printf("9 -> %d \n ",factorielle(9));
  return 0;
}
```

Récursivité sur les nombres : cas de base jamais atteint !

```c
int factorielle(int n){
  if((n == 0) || (n == 1))
    return 1 ;
  else
    /* return (n * factorielle(n-1)) */
    return (n * factorielle(n)) ;
}

int main(int arv,char * arg[]){
  printf("Test factorielle : \n");
  printf("1 -> %d \n ",factorielle(1));
  printf("5 -> %d \n ",factorielle(5));
  printf("9 -> %d \n ",factorielle(9));
  return 0;
}
```

Pas d'erreur à la compilation mais le programme ne s'arrête jamais !

Exercice

Écrire le programme qui calcule le plus grand commun dénominateur (pgcd) de deux entiers a et b.

Méthode des différences :
Si a et b sont multiples de d alors $a - b$ également.
Réciproquement, si d divise b et $a - b$ alors il divise également $(a - b) + b = a$.

Exercice

Écrire le programme qui calcule le plus grand commun dénominateur (pgcd) de deux entiers a et b.

Méthode des différences :
Si a et b sont multiples de d alors $a - b$ également.
Réciproquement, si d divise b et $a - b$ alors il divise également $(a - b) + b = a$.

Calcul du pgcd :
pgcd(a, 0) = a
pgcd(0, b) = b
pgcd(a, b)= pgcd(a, b-a) si a < b
pgcd(a, b) = pgcd(a-b, b) sinon

Solution

pgcd(m,n) :

- 2 cas de base :
  - Si a == 0 alors pgcd(b,a) = b
  - Si b == 0 alors pgcd(b,a) = a
- 2 cas généraux :
  - Si b < a alors pgcd(b, a)= pgcd(b, a-b)
  - Sinon pgcd(b, a) = pgcd(b-a,a)

```
int pgcd (int b, int a){
    if(a == 0) return b;
    if(b == 0) return a;

    if(b < a)
      return pgcd(b, a-b);
    return pgcd(b-a, a);
}
```