Algorithmes et programmation II : La récursivité

Souheib Baarir¹

¹Université 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

Introduction sur la récursivité

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));
}
```

4 / 15

Principe de la récursivité

- ▶ Décomposer le problème en un problème plus simple ⇒ 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).

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

- 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.

- 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.

- 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.

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

7 / 15

- 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

- 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*!

- 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

Introduction sur la récursivité

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 $n \in \{0, 1\}$

9 / 15

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));
```

9 / 15

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 \rightarrow %d \setminus n ", factorielle(1));
  printf("5 \rightarrow %d \n ", factorielle(5));
  printf("9 \rightarrow %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 \rightarrow %d \setminus n ", factorielle(1));
  printf("5 -> %d \n ", factorielle(5));
  printf("9 \rightarrow %d \n ", factorielle(9));
  return 0:
Test factorielle:
1 \rightarrow 1
5 \rightarrow 120
9 \to 362880
                                         ◆□▶ ◆□▶ ◆三▶ ◆三▶ ● ◆○○
```

La récursivité

L2 Mia - 2010/2011 10 / 15

S. Baarir (Paris10/LIP6)

Récursivité sur les nombres : exemple (3/4)

```
int factorielle(int n){
                                int main(int arv ,
  if ((n == 0) | | (n == 1))
                                         char * arg[]){
    return 1 ;
                                  int res=factorielle(5);
  else
                                  printf("%d", res) ;
    return
    (n * factorielle(n-1));
                                  return 0;
```

Récursivité sur les nombres : exemple (3/4)

S. Baarir (Paris10/LIP6)

```
int factorielle(int n){
                                   int main(int arv ,
  if((n == 0) || (n == 1))
                                             char * arg[]){
    return 1
                                     int res=factorielle(5);
  else
                                     printf("%d", res);
    return
                                     return 0;
    (n * factorielle(n-1));
   factorielle(5)
      5 * factorielle(4)
                  4 * factorielle(3)
                              ➤3 * factorielle(2)
                                        2 * factorielle(1)
          5 * 24
   120
```

La récursivité

L2 Mia - 2010/2011

11 / 15

Récursivité sur les nombres : exemple (4/4)

```
int factorielle(int n){
                                                                           int main(int arv ,
     if ((n == 0) | | (n == 1))
                                                                                                 char * arg[]){
         return 1
                                                                                int res=factorielle(5);
    else
                                                                                printf("%d", res);
         return
                                                                               return 0:
         (n * factorielle(n-1));
                                                                                         n4:1
                                                                            n<sub>2</sub>: 2
                                                                                         no: 2
                                                                                                       no: 2
                                                               no: 3
                                                                            no: 3
                                                                                         no: 3
                                                                                                       no: 3
                                                               n<sub>1</sub>:4
                                                                                         n<sub>1</sub>:4
                                                  n<sub>1</sub>:4
                                                                            n<sub>1</sub>:4
                                                                                                       n<sub>1</sub>:4
                                     no: 5
                                                  no:5
                                                               no:5
                                                                            no:5
                                                                                         n<sub>0</sub>:5
                                                                                                       no:5
                       res:?
                                    res:?
                                                 res:?
                                                              res:?
                                                                            res:?
                                                                                         res:?
                                                                                                      res:?
                                                                                         Appel à
                       Dans la
                                                               Appel à
                                    Appel à
                                                 Appel à
                                                                            Appel à
                                                                                                   factorielle (2)
                                                                                      factorielle(1)
                       Fonction
                                                            factorielle (3)
                                                                         factorielle(2)
                                 factorielle (5)
                                                                                                     Et calcul de
                                                                                       Et retour de la
                                                                                                      nx* 1 =>
                                                                                         valeur 1
                                                                                                     retourne 2
                 no:3
                               n<sub>1</sub>:4
                 n_0:5
                               no: 5
                                             no:5
                               res:?
                 res:?
                                             res:?
                                                           res: 120
                              Retour à
                                            Retour à
                                                           Retour à
              factorielle (3)
                             factorielle (4)
                                           factorielle (5)
                                                         la fonction main
               Et calcul de
                             Et calcul de
                                           Et calcul de
                                                           res = 120
                no* 2 =>
                              n.* 6 =>
                                            no* 24 =>
                                                          et retourne 0
               retourne 6
                             retourne 24
                                           retourne 120
```

La récursivité

L2 Mia - 2010/2011

Récursivité sur les nombres : cas de base jamais atteint!

```
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 \rightarrow %d \n ", factorielle(1));
  printf("5 \rightarrow %d \n ", factorielle(5));
  printf("9 \rightarrow %d \n ", factorielle(9));
  return 0:
```

Récursivité sur les nombres : cas de base jamais atteint!

```
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 \rightarrow %d \n ", factorielle(1));
  printf("5 \rightarrow %d \n ", factorielle(5));
  printf("9 \rightarrow %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 :

```
\frac{pgcd(a, 0) = a}{pgcd(0, b) = b}

pgcd(a, b) = pgcd(a, b-a) \text{ si } a < b

pgcd(a, b) = pgcd(a-b, b) \text{ sinon}
```



Solution

```
pgcd(m,n):
  2 cas de base :
       \triangleright Si a == 0 alors pgcd(b,a) = b

ightharpoonup Si b == 0 alors pgcd(b,a) = a
  2 cas généraux :
       Si b < a alors pgcd(b, a) = pgcd(b, a-b)</p>

ightharpoonup 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);
```