Définition

Définition

En informatique, on dit qu'une fonction est récursive,

Définition

En informatique, on dit qu'une fonction est récursive, lorsque cette fonction fait appel à elle-même.

Définition

En informatique, on dit qu'une fonction est récursive, lorsque cette fonction fait appel à elle-même.

Remarques

Définition

En informatique, on dit qu'une fonction est récursive, lorsque cette fonction fait appel à elle-même.

Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon itérative (avec des boucles) ou de façon récursive (en s'appelant elle-même).

Définition

En informatique, on dit qu'une fonction est récursive, lorsque cette fonction fait appel à elle-même.

Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon itérative (avec des boucles) ou de façon récursive (en s'appelant elle-même).
- Une fonction récursive doit toujours contenir une condition d'arrêt, dans le cas contraire elle s'appelle elle-même à l'infini et le programme ne se termine jamais.

Définition

En informatique, on dit qu'une fonction est récursive, lorsque cette fonction fait appel à elle-même.

Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon itérative (avec des boucles) ou de façon récursive (en s'appelant elle-même).
- Une fonction récursive doit toujours contenir une condition d'arrêt, dans le cas contraire elle s'appelle elle-même à l'infini et le programme ne se termine jamais.
- Les valeurs passées en paramètres lors des appels successifs doivent être différents, sinon la fonction s'exécute à l'identique à chaque appel et donc boucle à l'infini.

Exemple : les puissances positives

En mathématiques, pour un nombre quelconque a et un entier positif n, on définit a puissance n par :
$a^n = a \times a \times \cdots \times a$, et on convient que $a^0 = 1$

Exemple : les puissances positives

En mathématiques, pour un nombre quelconque a et un entier positif n, on définit a puissance n par :
$a^n = a \times a \times \cdots \times a$, et on convient que $a^0 = 1$

- Définir une fonction `puissance` qui prend en argument un flottant a et un entier n et renvoie a^n en effectuant ce calcul de façon itératif

Exemple : les puissances positives

En mathématiques, pour un nombre quelconque a et un entier positif n, on définit a puissance n par :
$a^n = a \times a \times \cdots \times a$, et on convient que $a^0 = 1$

- Définir une fonction `puissance` qui prend en argument un flottant a et un entier n et renvoie a^n en effectuant ce calcul de façon itératif
- Recopier et compléter : $a^n = \cdots \times a^{\cdots}$

Exemple : les puissances positives

En mathématiques, pour un nombre quelconque a et un entier positif n, on définit a puissance n par :
$a^n = a \times a \times \cdots \times a$, et on convient que $a^0 = 1$

- Définir une fonction `puissance` qui prend en argument un flottant a et un entier n et renvoie a^n en effectuant ce calcul de façon itératif
- Recopier et compléter : $a^n = \cdots \times a^{\cdots}$
- En déduire une version récursive de la fonction calculant les puissances

Exemple : les puissances positives

- Puissance : version itérative

```
1  float puissance_iteratif(float a, int n):
2      p=1
3      for (int k=1;k<n;k=k+1){
4          p=p*a;}
5      return p;
```

Exemple : les puissances positives

- Puissance : version itérative

```
1  float puissance_iteratif(float a, int n):
2      p=1
3      for (int k=1;k<n;k=k+1){
4          p=p*a;}
5      return p;
```

- $a^n = a \times a^{n-1}$

Exemple : les puissances positives

- Puissance : version itérative

```
1  float puissance_iteratif(float a, int n):
2      p=1
3      for (int k=1;k<n;k=k+1){
4          p=p*a;}
5      return p;
```

- $a^n = a \times a^{n-1}$
- Puissance : version récursive

```
1  float puissance_recursif(float a, int n):
2      if n==0:
3          return 1;
4      else:
5          return a*puissance_recursif(a,n-1);
```

Exemple : factorielle

1. Ecrire une fonction itérative en C permettant de calculer $n!$.

Exemple : dessin d'un triangle

Exemple : factorielle

1. Ecrire une fonction itérative en C permettant de calculer $n!$.
2. Ecrire la relation liant $n!$ et $(n-1)$.

Exemple : dessin d'un triangle

Exemple : factorielle

1. Ecrire une fonction itérative en C permettant de calculer $n!$.
2. Ecrire la relation liant $n!$ et $(n-1)$.
3. Donner une fonction récursive permettant de calculer $n!$

Exemple : dessin d'un triangle

Exemple : factorielle

1. Ecrire une fonction itérative en C permettant de calculer $n!$.
2. Ecrire la relation liant $n!$ et $(n-1)$.
3. Donner une fonction récursive permettant de calculer $n!$

Exemple : dessin d'un triangle

1. Ecrire une fonction itérative en C qui prend en argument un entier n et dessine un triangle de caractères *comme ci-dessous $(n = 4)$:

```
*
**
***
****
```

Exemple : factorielle

1. Ecrire une fonction itérative en C permettant de calculer $n!$.
2. Ecrire la relation liant $n!$ et $(n-1)$.
3. Donner une fonction récursive permettant de calculer $n!$

Exemple : dessin d'un triangle

1. Ecrire une fonction itérative en C qui prend en argument un entier n et dessine un triangle de caractères *comme ci-dessous $(n = 4)$:

```
*
**
***
****
```

2. Donner une version récursive de cette fonction.