Puissance nième d'un nombre [rc02] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 21 mai 2018

Table des matières

C++ - Puissance nième d'un nombre (Solution)

Mots-Clés Récursivité des actions ■
Requis Schéma itératif ■
Difficulté • • ◦ (30 min) ■

Objectif
Cet exercice calcule récursivement la puissance x^n d'un réel x par un entier $n \geq 0$ de plusieurs manières. Dans le même ordre d'idées, l'exercice @[Fonction produit] calcule récursivement le produit $a \cdot n$ d'un réel a par un entier $n \geq 0$ de plusieurs manières.

1 Puissance nième d'un nombre / pgpuissance

1.1 Puissance naïve

Soient un réel x et un entier $n \geq 0$.

L'idée la plus simple pour le calcul de x^n consiste à utiliser :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ x \cdot x^{n-1} & \text{sinon} \end{cases}$$

Écrivez une fonction récursive `puiss1(x,n)` qui calcule et renvoie la puissance d'un réel `x|` `(avec~\lstinline`n@ entier positif) à partir de la définition par récurrence.

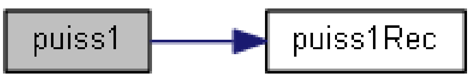
Validez votre fonction avec la solution.

Solution C++ @[pgpuissance.cpp]

```cpp
/**
  Puissance naive terminale
  @param[in] x - un réel
  @param[in] n - un entier
  @param[in] y - le cumul
  @return Puissance x^n
*/

double puiss1Rec(double x, int n, double y)
{
  return (n == 0 ? y : puiss1Rec(x,n - 1,x * y));
}

/**
  Puissance naive (suppose n >= 0)
  @param[in] x - un réel
  @param[in] n - un entier
  @return Puissance x^n
*/

double puiss1(double x, int n)
{
  return puiss1Rec(x,n,1.0);
}
```

Solution commentée

La fonction `puiss1Rec` est récursive terminale.



Combien y a-t-il d'appels récursifs ?

Solution simple
Le nombre d'appels récursifs est n.

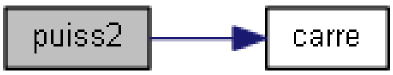
1.2 Stratégie basée sur la parité

La propriété suivante accélère le calcul :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ (x^{n \text{ div } 2})^2 & \text{si } n \text{ pair} \\ x \cdot x^{n-1} & \text{sinon} \end{cases}$$

Écrivez une fonction `carre(x)` qui renvoie le carré de `x` (réel).

Écrivez une fonction récursive `puiss2(x,n)` qui calcule et renvoie la puissance d'un réel `x` (avec `n` entier positif) comme décrit ci-avant.



Validez votre fonction avec la solution.

Solution C++ @[pgpuissance.cpp]

```
/**
  Puissance basée sur la parité de n (suppose n >= 0)
  @param[in] x - un réel
  @param[in] n - un entier
  @return Puissance x^n
*/

double puiss2(double x, int n)
{
  if (n == 0)
  {
    return 1.0;
  }
  else if (n % 2 == 0)
  {
    return carre(puiss2(x,n / 2));
  }
  else
  {
    return x * carre(puiss2(x,(n - 1) / 2));
  }
}
```

1.3 Autre stratégie basée sur la parité

Une autre façon d'accélérer significativement le calcul de la puissance (en le ramenant à au plus $2\log_2 n$) est la propriété suivante :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ (x^2)^{n \text{ div } 2} & \text{si } n \text{ pair} \\ x \cdot x^{n-1} & \text{sinon} \end{cases}$$

Par exemple, on calcule x^{10} en quatre multiplications au lieu de 9 :

$$x^{10} = (x^2)^5 = x^2\left((x^2)^4\right) = x^2((x^2)^2)^2$$

Écrivez une fonction récursive `puiss3(x,n)` qui calcule et renvoie la puissance d'un réel `x` (avec `n` entier positif) en appliquant la relation ci-dessus.

Validez votre fonction avec la solution.

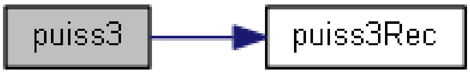
Solution C++ @[pgpuissance.cpp]

```
/**
  Puissance terminale
  @param[in] x - un réel
  @param[in] n - un entier
  @param[in] y - le cumul
  @return Puissance x^n
*/

double puiss3Rec(double x, int n, double y)
{
  if (n == 0)
  {
    return y;
  }
  else if (n % 2 == 0)
  {
    return puiss3Rec(x * x,n / 2,y);
  }
  else
  {
    return puiss3Rec(x,n - 1,x * y);
  }
}

/**
  Puissance basée sur la parité de n (suppose n >= 0)
  @param[in] x - un réel
  @param[in] n - un entier
  @return Puissance x^n
*/

double puiss3(double x, int n)
```

```
{
  return puiss3Rec(x,n,1.0);
}
```

Solution commentée

La fonction `puiss3Rec` est récursive terminale.



Donnez la suite des transformations de `(x,n,y)` pour le calcul de 5^8 puis le calcul de 5^7. Concluez.

Solution simple

Le cas bénéficiant de la plus forte accélération est celui où l'exposant est une puissance de 2. Voici la suite des transformations de `(x,n,y)` pour le calcul de 5^8 (en trois opérations au lieu de 8) :

```
(5,8,1) -> (25,4,1) -> (625,2,1) -> (390625,0,1)
```

La situation est moins favorable quand l'exposant n'est pas une puissance de 2 : le calcul de 5^7 se fait en 5 opérations, soit moins de $2\log_2 7$:

```
(5,7,1) -> (5,6,5) -> (25,3,5) -> (25,2,125) -> (625,1,125) ->
    (625,0,78125)
```

Cet algorithme est décrit dans le *Chandah Sutra d'Acharya Pingala* (écrit avant 200 ans avant J.C.).

1.4 Fonction itérative

La fonction du problème précédent étant récursive terminale,

Écrivez une fonction itérative `puiss4(x,n)`, équivalente à la version récursive terminale `puiss3(x,n)`, en remplaçant la liste des paramètres des appels récursifs par des affectations appropriées.

Validez votre fonction avec la solution.

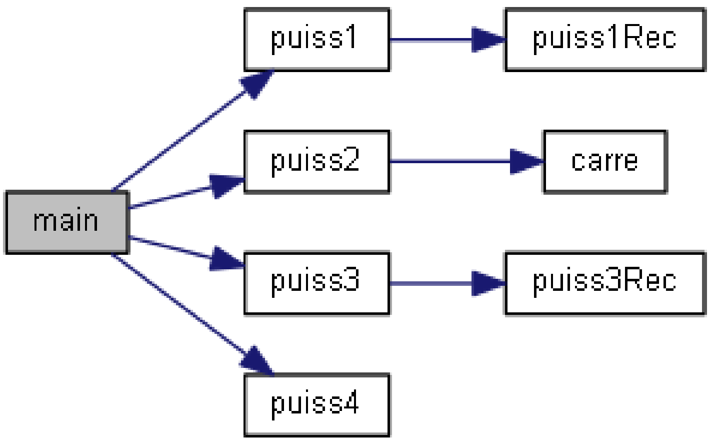
Solution C++ @[pgpuissance.cpp]

```
/**
  Puissance itérative basée sur la parité de n (suppose n >= 0)
  @param[in] x - un réel
  @param[in] n - un entier
  @return Puissance x^n
```

```
*/

double puiss4(double x, int n)
{
  double y = 1.0;
  while (n != 0)
  {
    if (n % 2 == 0)
    {
      x *= x;
      n /= 2;
    }
    else
    {
      y *= x;
      n -= 1;
    }
  }
  return y;
}
```

1.5 Programme de test

Écrivez un programme qui saisit un réel et un entier puis calcule et affiche le résultat de chacune des fonctions.



Testez. Exemple d'exécution :

```
Puissance x? 5
Ordre n? 7
==> puiss1(x,n) vaut 78125
==> puiss2(x,n) vaut 78125
==> puiss3(x,n) vaut 78125
==> puiss4(x,n) vaut 78125
```

Validez votre programme avec la solution.

Solution C++ @[pgpuissance.cpp]

```
int main()
{
  double x;
  cout<<"Puissance x? ";
  cin>>x;
  int n;
  cout<<"Ordre n? ";
  cin>>n;
  cout<<"==> puiss1(x,n) vaut "<<puiss1(x,n)<<endl;
  cout<<"==> puiss2(x,n) vaut "<<puiss2(x,n)<<endl;
  cout<<"==> puiss3(x,n) vaut "<<puiss3(x,n)<<endl;
  cout<<"==> puiss4(x,n) vaut "<<puiss4(x,n)<<endl;
}
```

2 Références générales

Comprend [Divay-CC1 :c1 :xm5] ■