

Fonction produit [rc02] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel  algoprogram  Version 21 mai 2018

Table des matières

1	Fonction produit / pgproduit	2
1.1	Produit naïf	2
1.2	Stratégie basée sur la parité	2
1.3	Autre stratégie basée sur la parité	3
1.4	Programme de test	4
2	Références générales	5

alg - Fonction produit (Solution)



Mots-Clés Récursivité des actions ■
Requis Algorithmes paramétrés ■
Difficulté ●○○ (20 min) ■



Objectif

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

1 Fonction produit / pgproduit

1.1 Produit naïf

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

L'idée la plus simple pour calculer le produit de a par n consiste à se dire :

- si $n = 0$ alors :

$$P(a, n) = 0$$

- si $n \geq 1$ et si je sais calculer $P(a, n - 1)$ alors :

$$\begin{aligned} P(a, n) &= a \cdot n \\ &= a \cdot (n - 1) + a \\ &= P(a, n - 1) + a \end{aligned}$$



Écrivez une fonction récursive `produit1(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution alg @[pgproduit.alg]

```

Fonction produit1 ( a : Réel ; n : Entier ) : Réel
Début
  | Si ( n = 0 ) Alors
  |   | Retourner ( 0 )
  | Sinon
  |   | Retourner ( produit1 ( a , n - 1 ) + a )
  | FinSi
Fin

```

1.2 Stratégie basée sur la parité

L'idée basée sur la méthode « diviser pour régner » consiste à se dire : supposons que l'on sache calculer le produit de a par $n \text{ div } 2$. Comment calculer alors le produit de a par n ?

Cela donne naissance à l'analyse suivante :

- si $n = 0$ alors :

$$P(a, n) = 0$$

- si n est pair et $n \geq 2$, alors si je sais calculer $P(a, n \text{ div } 2)$, on a :

$$P(a, n) = P(a, n \text{ div } 2) + P(a, n \text{ div } 2)$$

- si n est impair et $n \geq 1$, alors $n - 1$ est impair et donc :

$$P(a, n) = P(a, (n - 1) \text{ div } 2) + P(a, (n - 1) \text{ div } 2) + a$$



Écrivez une fonction récursive `produit2(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution alg @[pgproduit.alg]

```

Fonction produit2 ( a : Réel ; n : Entier ) : Réel
Variable p : Réel
Début
  | Si ( n = 0 ) Alors
  |   | Retourner ( 0 )
  | Sinon
  |   | p <- produit2 ( a , DivEnt ( n , 2 ) )
  |   | Si ( Modulo ( n , 2 ) = 0 ) Alors
  |   |   | Retourner ( p + p )
  |   |   Sinon
  |   |   | Retourner ( p + p + a )
  |   | FinSi
  | FinSi
Fin

```

1.3 Autre stratégie basée sur la parité

Une idée similaire à celle de la deuxième méthode mais basée sur l'hypothèse que l'on sait calculer $P(2a, n \text{ div } 2)$ est la suivante :

- si n est nul alors :

$$P(a, n) = 0$$

- si n est pair et $n \geq 2$, alors :

$$P(a, n) = P(2a, n \text{ div } 2)$$

- si n est impair et $n \geq 1$, alors $n - 1$ est impair, d'où :

$$P(a, n) = P(2a, (n - 1) \text{ div } 2) + a$$



Écrivez une fonction récursive `produit3(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution alg @[pgproduit.alg]

```

Fonction produit3 ( a : Réel ; n : Entier ) : Réel
Variable p : Réel
Début
  | Si ( n = 0 ) Alors
  |   | Retourner ( 0 )
  | Sinon

```

```

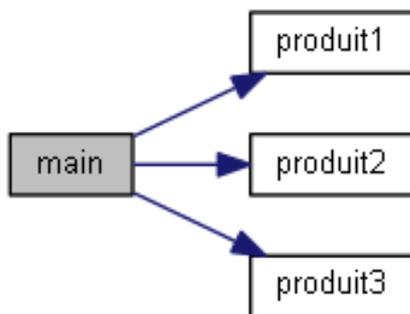
|   | p <- produit3 ( a + a , DivEnt ( n , 2 ) )
|   | Si ( Modulo ( n , 2 ) = 0 ) Alors
|   |   | Retourner ( p )
|   | Sinon
|   |   | Retourner ( p + a )
|   | FinSi
| FinSi
Fin

```

1.4 Programme de test



Écrivez un algorithme qui saisit un réel dans x et un entier dans n puis calcule et affiche le résultat des fonctions `produit1(x,n)`, `produit2(x,n)` et `produit3(x,n)`.



Testez. Exemple d'exécution :

```

Un réel x? 1.5
Un entier n? 97
==> produit1(x,n) vaut 145.5
==> produit2(x,n) vaut 145.5
==> produit3(x,n) vaut 145.5

```



Validez votre algorithme avec la solution.

Solution alg @[pgproduit.alg]

```

Algorithme pgproduit
Variable x : Réel
Variable n : Entier
Début
| Afficher ( "Un réel x? " )
| Saisir ( x )
| Afficher ( "Un entier n? " )
| Saisir ( n )
| Afficher ( "=> produit1(x,n) vaut " , produit1 ( x , n ) )
| Afficher ( "=> produit2(x,n) vaut " , produit2 ( x , n ) )
| Afficher ( "=> produit3(x,n) vaut " , produit3 ( x , n ) )
Fin

```

2 Références générales

Comprend ■