

Projet de Mathématiques

Algorithme de calcul des logarithmes

DURAND Pierre (P2)

FABRE Morgan (P2)

LECOINTRE Arnaud (P2)

Enseignant tuteur : Monsieur Le Mignot

IUT Paris XIII – Semestre 4 - Année 2008

Sommaire

I – Rappel du sujet.....	3
II – Analyse du sujet	4
1) Démonstrations	4
2) Algorithme	5
III – Présentation de notre programme	7
IV – Code source de la fonction logarithme ..	8

I – Rappel du sujet

Sujet 3 :

x désigne un réel strictement positif.

1) Montrer que $x=M.10^\alpha$ avec $M \in [1;10[$ et $\alpha \in \mathbb{Z}$ puis que $M^{10}=M_1.10^{\alpha_1}$ avec $M_1 \in [1;10[$ et $\alpha_1 \in \{0,1,\dots,9\}$.

Déduire des résultats précédents un algorithme de calcul de $\log_{10}(x)$.

Ecrire un programme qui calcule $\log_{10}(x)$ lorsqu'on lui donne x .

2) Comment peut-on obtenir $\log_2(x)$?

Ecrire le programme correspondant.

II – Analyse du sujet

1) Démonstrations

On veut montrer que $x = M \cdot 10^\alpha$

On sait que :

- $x \in \mathbb{R}^{+*}$
- $\alpha \in \mathbb{Z}$
- $M \in [1; 10[$

On peut donc dire que $10^\alpha \in \mathbb{R}^{+*}$

Ceci implique que $M \cdot 10^\alpha \in \mathbb{R}^{+*}$ (tout comme x)

$$\text{Or, } x = \frac{x}{10^\alpha} * 10^\alpha$$

$$\text{On en déduit donc que } M = \frac{x}{10^\alpha}$$

Si $x \in]0; 1[$ α est négatif

Exemple : $0.00123 = 1.23 * 10^{-3}$ ($\alpha = -3$)

Si $x \in [1; +\infty[$ α est positif

Exemple : $563.67 = 5.6367 * 10^2$ ($\alpha = 2$)

$x = M \cdot 10^\alpha$ est en fait tout simplement l'écriture scientifique de x .

On veut montrer que $M^{10} = M_1 \cdot 10^{\alpha_1}$

On sait que :

- $M \in [1; 10[$
- $M_1 \in [1; 10[$
- $\alpha_1 \in \{0, 1, \dots, 9\}$

On peut donc déduire que $M^{10} \in [1; +\infty[$ et $10^{\alpha_1} \in [1; +\infty[$

On remarque que $M^{10} = M_1 \cdot 10^{\alpha_1}$ est de la même forme que $x = M \cdot 10^\alpha$.

On peut donc conclure de la même manière que $M_1 = \frac{M}{10^{\alpha_1}}$.

Pour calculer le logarithme en base 2, il suffit de procéder de la même manière mais en remplaçant 10 par 2.

2) Algorithme

A partir des résultats précédents, déterminons un algorithme de calcul de $\log_n(x)$.

Pour certains x , $\log_n(x)$ peut avoir un très grand nombre de chiffres après la virgule. C'est pourquoi nous allons introduire une variable de précision qui détermine le nombre de calculs que doit faire l'algorithme.

Nous avons choisi de procéder par récursivité plutôt que par itération.

Algorithme de la fonction logarithme de X

Fonction logarithme(X , base , precision):

retour=0

Tant que X<1

 On multiplie « X » par « base »

 On décrémente « retour » de 1

Fin Tant que

Tant que X≥base

 On divise « X » par « base »

 On incrémente « retour » de 1

Fin Tant que

On décrémente « precision » de 1 car on vient de faire un calcul

Si precision>0 ET X!=1

 retour=retour+log_{base}(X^{base})*base⁻¹

Fin Si

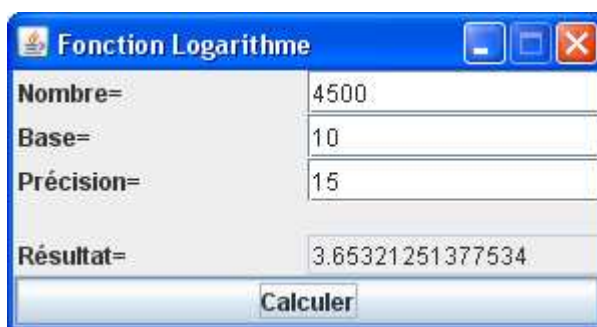
On retourne « retour »

Fin de la fonction logarithme

III – Présentation de notre programme

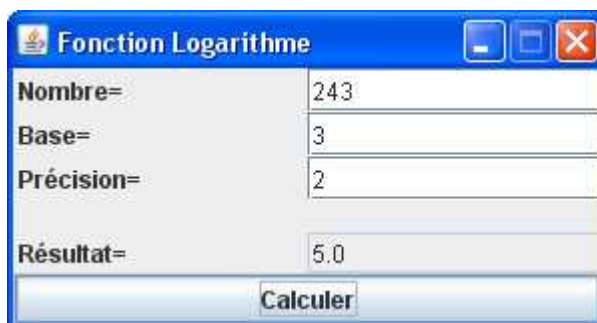
Nous avons choisi le langage Java pour implémenter notre programme de calcul d'un algorithme. Le sujet demandait de créer un programme devant calculer les logarithmes en base 2 et 10 d'un réel strictement positif. Notre programme satisfait à la demande et même plus, dans le sens où il fonctionne pour n'importe quelle base. Il est clair, simple d'utilisation et performant.

Voici quelques aperçus de calculs effectués avec :



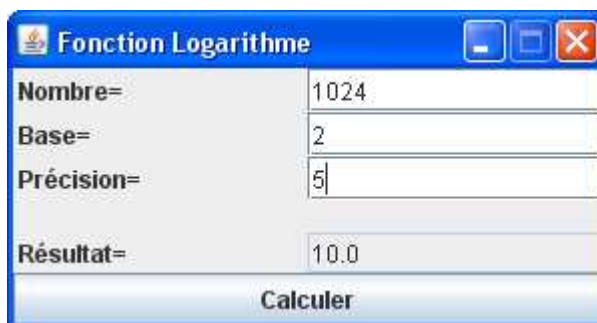
Nombre=	4500
Base=	10
Précision=	15
Résultat=	3.65321251377534

Calculer



Nombre=	243
Base=	3
Précision=	2
Résultat=	5.0

Calculer



Nombre=	1024
Base=	2
Précision=	5
Résultat=	10.0

Calculer

IV – Code source de la fonction logarithme

```
import java.io.*;

public class MathsIUT
{
    /*
    n est le nombre
    p est la puissance
    */
    public static double puissance(double n,int p)
    {
        double r;

        if(p==0)
            r=1;
        else if(p>0)
            r=n*puissance(n,p-1);
        else
            r=puissance(n,p+1)/n;

        return r;
    }
}
```

```

/*
n est le nombre
b est la base du logarithme
p est la précision du logarithme
*/
public static double logarithme(double n,int b,int p)
{
    double r=0;
    while(n<1)
    {
        n=n*b;
        r--;
    }
    while(n>=b)
    {
        n=n/b;
        r++;
    }

    p--;
    if(p>0 && n!=1)
        r=r+logarithme(puissance(n,b),b,p)*puissance(b,-1);

    return r;
}
}

```

En plus de la fonction logarithme, nous avons également dû réécrire la fonction puissance.