Modèle Quadrinomial :

```
FUNCTION PrixQuadrinomial
(Spot1,Spot2,     // Prix des Sous-Jacents à l'instant initial
Taux,             // Valeur du taux d'intérêt supposé constant
Volatilite1, Volatilite2,  // Valeur des volatilités
MaturiteAnnuelle,          // Maturité de l'option
CoefCorrelation,           // Valeur de la corrélation des rendements
PrixExercice : Double;     // Prix d'exercice
NombreDePeriodes,          // Nombre de Périodes de l'arbre
CallOuPut : Integer) : Double;
VAR
i,j,k : Integer;
DeltaT : Double;           // Durée d'une période
u11,u12,u2,d11,d12,d2 : Double;          // Coefficients de hausse
et de baisse
Discount : Double;         // Coefficient d'actualisation
ValeurSousJacent1, ValeurSousJacent2 , // Tableau de valeurs des
                                       sous-jacents
ValeurOption : array of array of double;   // Tableau de valeurs de
                                       l'option
BEGIN
//          1. Calculs Prélimaires
// Calcul de la longueur d'une période 'DeltaT'
DeltaT := MaturiteAnnuelle / NombreDePeriodes;
```

```
// Calculs des coefficients de hausse 'u' et de baisse 'd' au cours de
chaque
// période
u11 := Exp(0.5*(Taux-sqr(CoefCorrelation*Volatilite1)) *
DeltaT
 +CoefCorrelation*Volatilite1*sqrt(DeltaT));
u12 := Exp(0.5*(Taux-(1-sqr(CoefCorrelation))*sqr
(Volatilite1)) * DeltaT
 +sqrt((1-sqr(CoefCorrelation)))*Volatilite1*sqrt(DeltaT));
d11 := Exp(0.5*(Taux-sqr(CoefCorrelation*Volatilite1)) *
DeltaT
 -CoefCorrelation*Volatilite1*sqrt(DeltaT));
d12 := Exp(0.5*(Taux-(1-sqr(CoefCorrelation))*sqr
(Volatilite1)) * DeltaT
 -sqrt((1-sqr(CoefCorrelation)))*Volatilite1*sqrt(DeltaT));
u2 := Exp((Taux-0.5*sqr(Volatilite2)) * DeltaT
 + Volatilite2*sqrt(DeltaT));
d2 := Exp((Taux-0.5*sqr(Volatilite2)) * DeltaT
 - Volatilite2*sqrt(DeltaT));

// Calculs du coefficient d'actualisation
Discount := Exp(- Taux*DeltaT);

//          2. Calcul des Valeurs des Sous-Jacent 'ValeurSousJacent1'
//          et 'ValeurSousJacent2' à l'Echéance

// Allocation de la Mémoire de la Matrice : ValeurSousJacent1
SetLength(ValeurSousJacent1, NombreDePeriodes+1,
NombreDePeriodes+1);

// Allocation de la Mémoire de la Matrice : ValeurSousJacent2
SetLength(ValeurSousJacent2, NombreDePeriodes+1,
NombreDePeriodes+1);

// Calculs des valeurs des actifs
```

```
For i:=1 To NombreDePeriodes+1 Do
For j:=1 To NombreDePeriodes+1 Do
Begin
ValeurSousJacent1[i][j] := Spot1          * IntPower(u11,
NombreDePeriodes+1-i)
                                  * IntPower(d11, i-1)
                                  * IntPower(u12,
NombreDePeriodes+1-j)
                                  * IntPower(d12, j-1);
ValeurSousJacent2[i][j] := Spot2          * IntPower(u2,
NombreDePeriodes+1-i)
                                  * IntPower(d2, i-1);
End;

//          3. Calcul des Valeurs Finales de l'Option

// Allocation de la Mémoire du Vecteur : ValeurOption
SetLength(ValeurOption, NombreDePeriodes+1,
NombreDePeriodes+1);

// Calculs des Valeurs Finales de l'Option
For i:= 1 To NombreDePeriodes+1 Do
For j:= 1 To NombreDePeriodes+1 Do
ValeurOption[i][j] := Max( CallOuPut * Max
(ValeurSousJacent1[i][j],
                    ValeurSousJacent2[i][j]) - PrixExercice
                    , 0);
```

```
//          4. Induction Arrière, Calcul de la Valeur de l'Option à
l'Instant Initial
For i:=NombreDePeriodes DownTo 1 Do
For j:=1 To i Do
For k:=1 To i Do
ValeurOption[j][k] := Discount * (ValeurOption[j][k]
                                + ValeurOption[j][k+1]
                                + ValeurOption[j+1][k]
                                + ValeurOption[j+1][k+1]) / 4;

Result := ValeurOption[1][1];

// Libération Mémoire des Matrices
ValeurSousJacent1 := nil ;
ValeurSousJacent2:= nil ;
ValeurOption := nil ;

END;
```