20.6.1 Task-Projekt 2

Dem Projekt 2 liegt die folgende Aufgabe zugrunde: *Berechnung der Adjunkte einer (quadratischen) Matrix über den Laplace'schen Entwicklungssatz.*

Diese Lösungsidee wurde entwickelt: Bei der Berechnung der Adjunkte einer Matrix – als Transponierte der Co-Faktor-Matrix – müssen sehr viele Unter-Determinanten (Minoren) berechnet werden, die i.A. nichts mit einander zu tun haben. Die Aufgabe wird in n Teilaufgaben zur Berechnung von Unter-Determinanten zerlegt.

Aus der Lösungsidee entstand dieser Vorschlag zur Umsetzung:

- In der Task-Klasse wird eine Main()-Funktion deklariert, bei der für alle Berechnungen der Unter-Determinanten der gleiche Algorithmus verwendet werden kann:

```
Public Function Main() As Variant
' Array von Unter-Determinanten berechnen...
  Return aFloatArray
End
```

- Im (Haupt-)Programm wird in einer FOR-NEXT-Kontrollstruktur jeweils ein eigener Task angeschoben, in dem n Unter-Determinanten berechnet werden. Die Bearbeitung der n Teil-Aufgaben wird jeweils an einen Task delegiert! Die einzelnen Tasks werden zwar sequentiell gestartet, laufen dann aber alle parallel. Das wird man praktisch leider kaum wahrnehmen, weil die Berechnungen – selbst bei einer Matrix der Dimension 30 – bei den modernen Computern so kurz ausfallen.

- Wenn alle n Unter-Determinanten berechnet sind, dann wird aus den n Teillösungen die Adjunkte berechnet und ausgegeben.

Es gelten folgende Spezifikationen:

- Es werden den Tasks jeweils 2 Argumente übergeben.
- Den Tasks werden bei der Erzeugung (_new() in der Task-Klasse) sowohl die *Matrix* als Argument übergeben, auf der sie zu arbeiten haben, als auch die spezifische *Zeile*, in der alle Co-Faktoren bestimmt werden sollen.
- Es wird jeweils nach dem automatischen Beenden der Tasks nur der Funktionswert der Main-Methode (Datentyp Variant) im Task_Kill()-Event mit *LAST.Value* ausgelesen und in einer Matrix zwischen-gespeichert.

Die Quelltexte für die Task-Klasse und das Haupt-Programm werden vollständig angegeben. In ihnen wird die o.a. Lösungsidee umgesetzt. Zuerst wird der Inhalt der Task-Klasse vorgestellt – gespeichert in der Datei *TaskMinors.class*:

```
' Gambas class file

''' Diese Klasse berechnet die Minoren der angegebenen Matrix, wenn die angegebene Zeile gestrichen wurde.
''' Der Funktionswert ist ein entsprechend dimensioniertes Float-Array der einzelnen Determinanten.

Inherits Task

Property Tag As Variant

Private $hMatrix As Matrix
Private $vTag As Variant
Private $iRow As Integer

Public Sub _new({Matrix} As Matrix, Row As Integer)
  $hMatrix = {Matrix}
  $iRow = Row
End ' _new(..)

Public Function Main() As Variant
  Dim hMinor As New Matrix($hMatrix.Width - 1, $hMatrix.Width - 1)
  Dim iRemCol, iSrcRow, iSrcCol, iDstRow, iDstCol As Integer
  Dim aFloatArray As New Float[]

  For iRemCol = 0 To hMinor.Width
  ' Untermatrix aufbauen
```

```
    iDstRow = 0
    For iSrcRow = 0 To hMinor.Height
      If iSrcRow = $iRow Then Continue
      iDstCol = 0
      For iSrcCol = 0 To hMinor.Width
        If iSrcCol = iRemCol Then Continue
        hMinor[iDstRow, iDstCol] = $hMatrix[iSrcRow, iSrcCol]
        Inc iDstCol
      Next
      Inc iDstRow
    Next
    aFloatArray.Add(hMinor.Det())
  Next

  Return aFloatArray ' Funktionswert vom Datentyp Float-Array

End ' Function Main() As Variant

Private Function Tag_Read() As Variant
  Return $vTag
End ' Function Tag_Read()

Private Sub Tag_Write(Value As Variant)
  $vTag = Value
End ' Tag_Write(Value As Variant)
```

Datei MMain.module:

```
' Gambas module file

' Dimension einer zufälligen quadratischen Matrix (d > 1)
Const Dimension As Integer = 15 ' Maximum 30

Private $hMatrix As Matrix
Private $hAdjugate As Matrix
Private $iTasks As Integer

Public Sub Main()
  Dim iRow, iCol As Integer
  Dim hDet As TaskMinors
  Dim hOther As Matrix

' Matrix der vorgegebenen Dimension mit zufälligen Elementen erzeugen
  $hMatrix = RandomMatrix()
  Print "Matrix:";; $hMatrix.ToString()

  $hAdjugate = New Matrix($hMatrix.Width, $hMatrix.Height)
  $iTasks = 0

  For iRow = 0 To $hMatrix.Height – 1 ' k Task werden erzeugt
    For iCol = 0 To $hMatrix.Width - 1
    ' Alle Minoren beim Streichen von Zeile iRow berechnen
      hDet = New TaskMinors($hMatrix, iRow) As "TaskMinors"
      hDet.Tag = iRow
      Inc $iTasks ' Zählt die gestarteten Tasks
    Next
  ' Die Tasks werden nach ihrer Instantiierung beim nächsten Aufruf der Event-Schleife gestartet. Damit
  ' nicht alle auf einmal starten hier jeden einzelnen Task über WAIT anlaufen lassen.
    Wait
  Next

' Auf die Berechnung der Adjunkte warten (-->> Minor_Kill())
  While $iTasks ' So lange noch gerechnet wird ...
    Wait 0.001  ' ...0.001 Sekunden warten
  Wend

  Print "Adjunkte adj(A):";; $hAdjugate.ToString() ' Anzeige Adjunkte

End ' Main()

Public Sub TaskMinors_Kill()
  Dim iRow, iCol As Integer
  Dim aDet As Float[] = Last.Value

  iRow = Last.Tag
  For iCol = 0 To aDet.Max
  ' Die Adjunkte ist die Transponierte der Co-Faktor-Matrix.
    $hAdjugate[iCol, iRow] = IIf(Even(iRow + iCol), 1, -1) * aDet[iCol]
  Next
  Dec $iTasks ' Task-Anzahl um 1 verringern

End ' TaskMinors_Kill()
```

```
Public Function RandomMatrix() As Matrix
  Dim hMatrix As New Matrix(Dimension, Dimension, False)
  Dim iRow, iCol As Integer

  For iRow = 0 To hMatrix.Height - 2
    hMatrix[iRow, 0] = Rnd(0, hMatrix.Height + 1)
    For iCol = 1 To hMatrix.Width - 1
      hMatrix[iRow, iCol] = IIf(iRow = iCol, iRow + 1, hMatrix.Height)
    Next
  Next

  For iCol = 0 To hMatrix.Width - 1
    hMatrix[hMatrix.Height - 1, iCol] = Rnd(0, hMatrix.Width + 1)
  Next

  Return hMatrix ' → Matrix mit zufälligen Elementen bereitstellen

End ' RandomMatrix() As Matrix
```

Das wurde in der Konsole der IDE für eine Berechnung mit der Dimension 3 ausgegeben:

```
Matrix:
[[0.13973 | 3 | 3] ; [0.60071 | 2 | 3] ; [2.73226 | 0.28943 | 2.89609]]

Adjunkte adj(A):
[[4.92387 | -7.81996 | 3] ; [6.45708 | -7.79211 | 1.38293] ; [-5.29066 | 8.15636 | -1.52266]]
```

Beachten Sie, dass sich bei Matrizen höherer Dimension (>10) Rundungsfehler bereits sehr stark auf das Ergebnis auswirken. Die Identität adj(A) = det(A)·A^{-1} liefert eine präzisere Berechnung der Adjunkten.