# Recursive Page Layout Method

November 24, 2021

# Quick Overview

- Fortran program with some routines to recursively Layout Page
- Basic objects are Box, Cell and Arrow
- Cell is Contained inside the Box with specified space around it
- Latex content can be put inside the Cell
- Multiple Cells can be linked with Arrows
- All the sizes are defined in relative terms
- For each relative size, there is default multiplier and further multipliers can be optionally specified
- Division of the boxes can be done in a 2D fashion using a matrix or 1D fashion Horizontally or Vertically

# Getting Started – Basic Example Program

The following code is used to produce a simple Basic Box

```
1 TYPE(Box) :: A(1,1) !Declaration of Box
2
3 CALL StartTekzDiag(A(1,1)) !Initialize the Page with Box
4 CALL Draw(A) !Draw the Box in Page
5 CALL EndTikzDiag() !End the Process
```

This will produce the Box and Cell pair in Page (Gray outer shadowed rectangle)

Figure 1: Default Box

By Default, the Box Border is Invisible and Cell contained in it has Rectangle Shape with Space of 0.9 Multiplier around it on each side and Rounded Corners of .1 Multiplier with respect to smallest side

# Refinement of Box

We can Refine the Box with the Options available.

```
3 TYPE(Box) :: A(1,1) !Declaration of Box
4
5 CALL StartTekzDiag(A(1,1))
6 CALL  Refine(A(1,1), BoxBoundaryColour = "red" ,
      CellInteriorColour = "blue")
7 CALL Draw(A)!Refine Box with specified Options
8 CALL EndTekzDiag()
```

This Code Refines the Box A from Figure 1 as

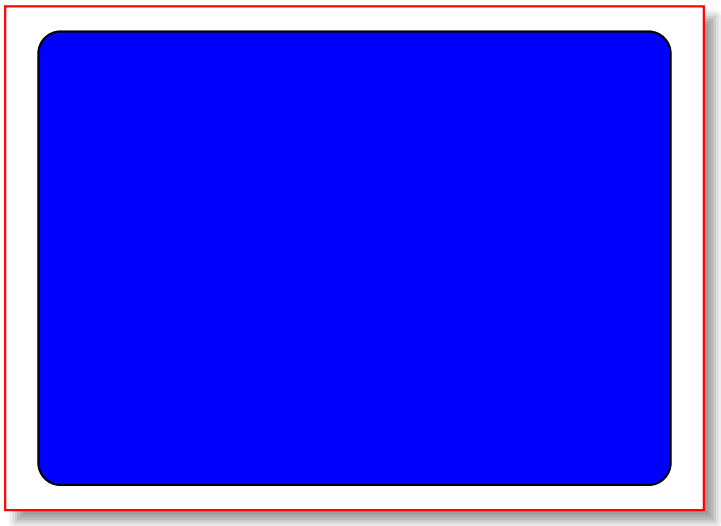


Figure 2: Refined Box

# Options for Refinement of Box

- There are many options which can be specified to Refine the Default Box.
  - "BoxBoundaryColour" – Colour of Box Boundary
  - "BoxInteriorColour" – Colour of Box Interior
  - Options for Cell contained inside Box
- By Default, the Box has no Interior or Boundary Colour and contains Cell with Default Options
- The Default Cell can also be Refined with Options
  - "CellShape" – Shape of the Cell
  - "LatexContent" – Content Inside Cell
  - "CellXShift" – Shift along X-Axis relative to Box
  - "CellYShift" – Shift along Y-Axis relative to Box
- By Default, the Cell Shape is Rectangle with .9 Multiplier Dimensions, .1 Multiplier Corners, has no Latex Content in it and has zero Shift along each Axis
- There are various Shapes available like Oval, Diamond, Cloud etc.
- Each Shape can be further Refined with respective Options

# Examples of Options – Cell Shape

1. To change the Shape of cell to Circle inside Box, use the following call

```
3 CALL StartTekzDiag(A(1,1))
4
5 CALL Refine(A(1,1), CellShape = "Circle")
6 CALL Draw(A)!Refine the Cell with Circle
7
8 CALL EndTekzDiag()
```

This will Refine the Box in Figure 1 with Cell of Shape “Circle”

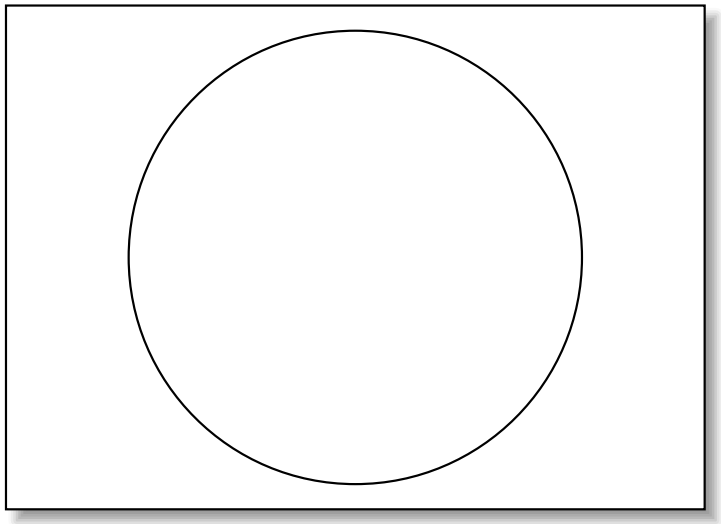


Figure 3: Shape Cell

# Examples of Options – Dimensions of Shape

3. To change the Dimensions of cell inside Box, use the following call

```
3 CALL StartTekzDiag(A(1,1))
4 CALL Refine(A(1,1), Width = .5, Height = .5)
5 CALL Draw(A) !Refine the Width and Height of Rectangle of
      Cell in Box A(1,1)
6 CALL EndTekzDiag()
```

This will Refine the Box in Figure 1 with Cell Shape of Rectangle with Width and Height .5 of the Box

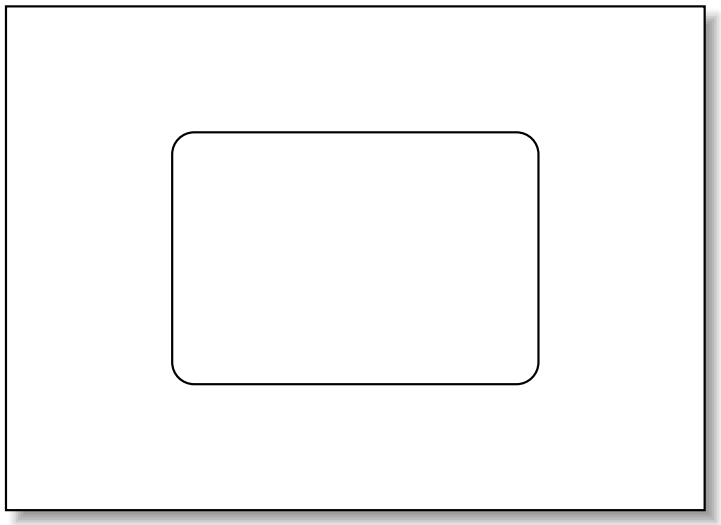


Figure 4: Dimensions Cell Shape

# Examples of Options – Shift of Shape

2. To Shift the Shape of Cell along Axes inside Box, use the following call

```
3 CALL StartTekzDiag(A(1,1))
4 CALL Refine(A(1,1), Width = .5, Height = .5)
5 CALL Refine(A(1,1), CellXShift = .5)
6 CALL Draw(A) !Refine the Cell to Shift along X-axis
7 CALL EndTekzDiag()
```

This will Refine the Box in Figure 4 with Cell Shift of .5 along X Axis

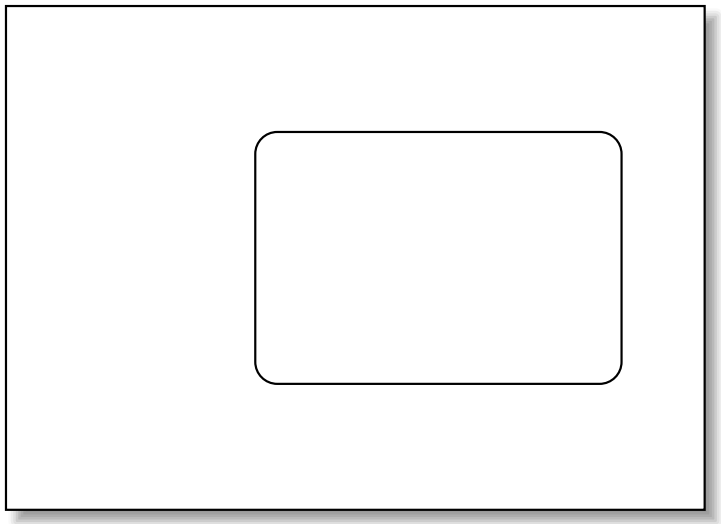


Figure 5: Shift Cell

# Division of Cell

Each Cell can be further divided into Sub-boxes

For Example, the following code divides the Cell of Box A from Figure 1 into Matrix of Sub-boxes named B

```
3 INTEGER, PARAMETER :: m= 2, n= 3 !Rows & Columns of Sub-boxes
4 TYPE(Box) :: B(m,n) !Sub-box Matrix Declaration
5
6 CALL Divide(B, A(1,1))!Divide A into Matrix of Size B
7
8 CALL Draw(B)
```

This will produce the following :

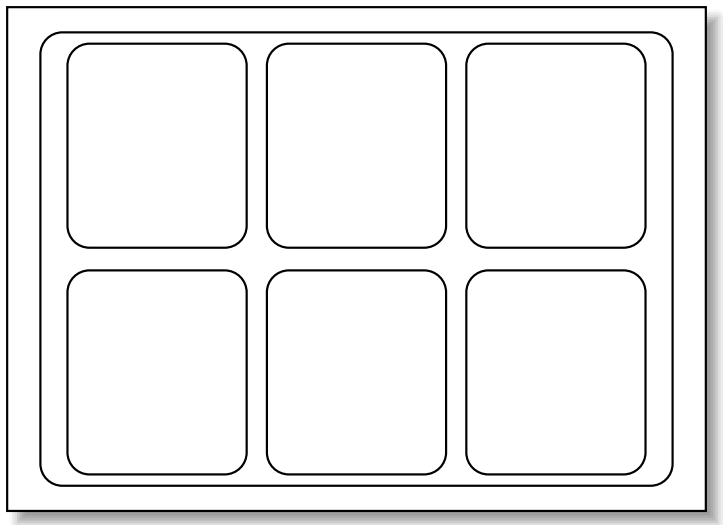


Figure 6: Cell Division

# Special Case – Vector Division of Cell

Two particular cases are commonly encountered : Horizontal and Vertical Division

For Example, the following code divides the Cell of Box A from Figure 1 into Horizontal Vector of Sub-boxes named C

```
3 INTEGER, PARAMETER :: n = 4 !Columns of Sub-boxes
4 TYPE(Box) :: C(1,n) !Sub-box Matrix Declaration
5
6 CALL Divide(C, A(1,1))!Divide A into Vector of Size C
7 CALL Draw(C)
```

This will produce the following :

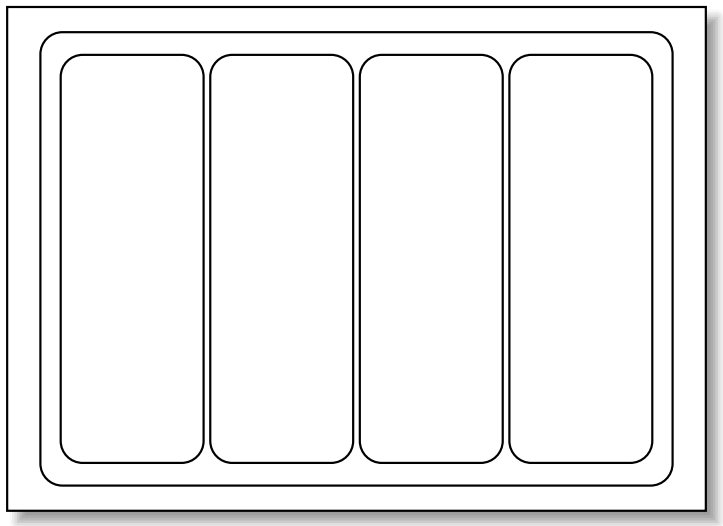


Figure 7: Vector Cell Division

# Linking of Cells

Example to produce a Simple Link

```
3 CALL DrawLink(B(1,1), B(1,2)) !Link Cell of B(1,1) and B(1,2)
```

This will produce the following Link

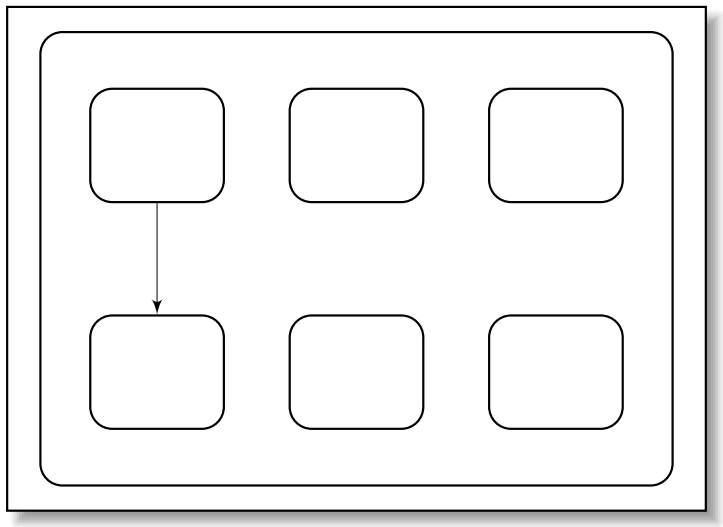


Figure 8: Simple Link

# Special Case – Array Linkage of Cells

The following code produces array of links among Cells

```
3 INTEGER :: i, j
4
5 DO i = 1, size(B,1)
6   DO j = 1,  size(B,2) -1
7     CALL DrawLink(B(i,j), B(i,j+1))
8   END DO ! Links Cell of Boxes   B(i,j) and B(i+1,j)
9 END DO
```

This will produce the following figure

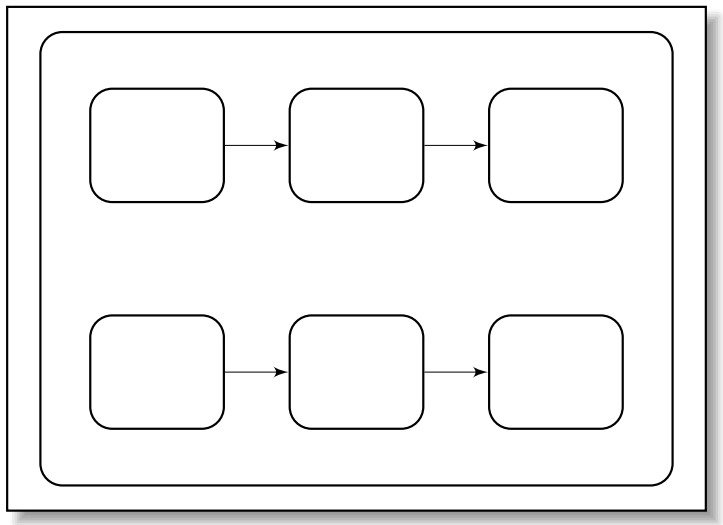


Figure 9: Array of Links

# Refinement of Link

Default Link has 0pt Width of Line, "-latex"' Latex Type Arrow, Black Colour, $0^{\circ}$ Angle along X-Axis and joins Centers of Cells

The following code refines the Default Link joining the Cells

```
3 CALL DrawLink(B(1,1), B(2,1), Angle = 60., ArrowColour = "
      blue") !Refine Link Cell of B(1,1) and B(1,2)
```

This will produce the following Link

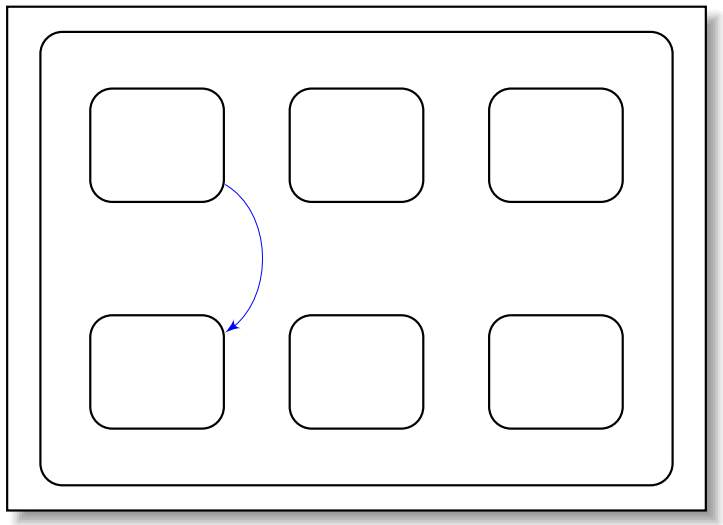
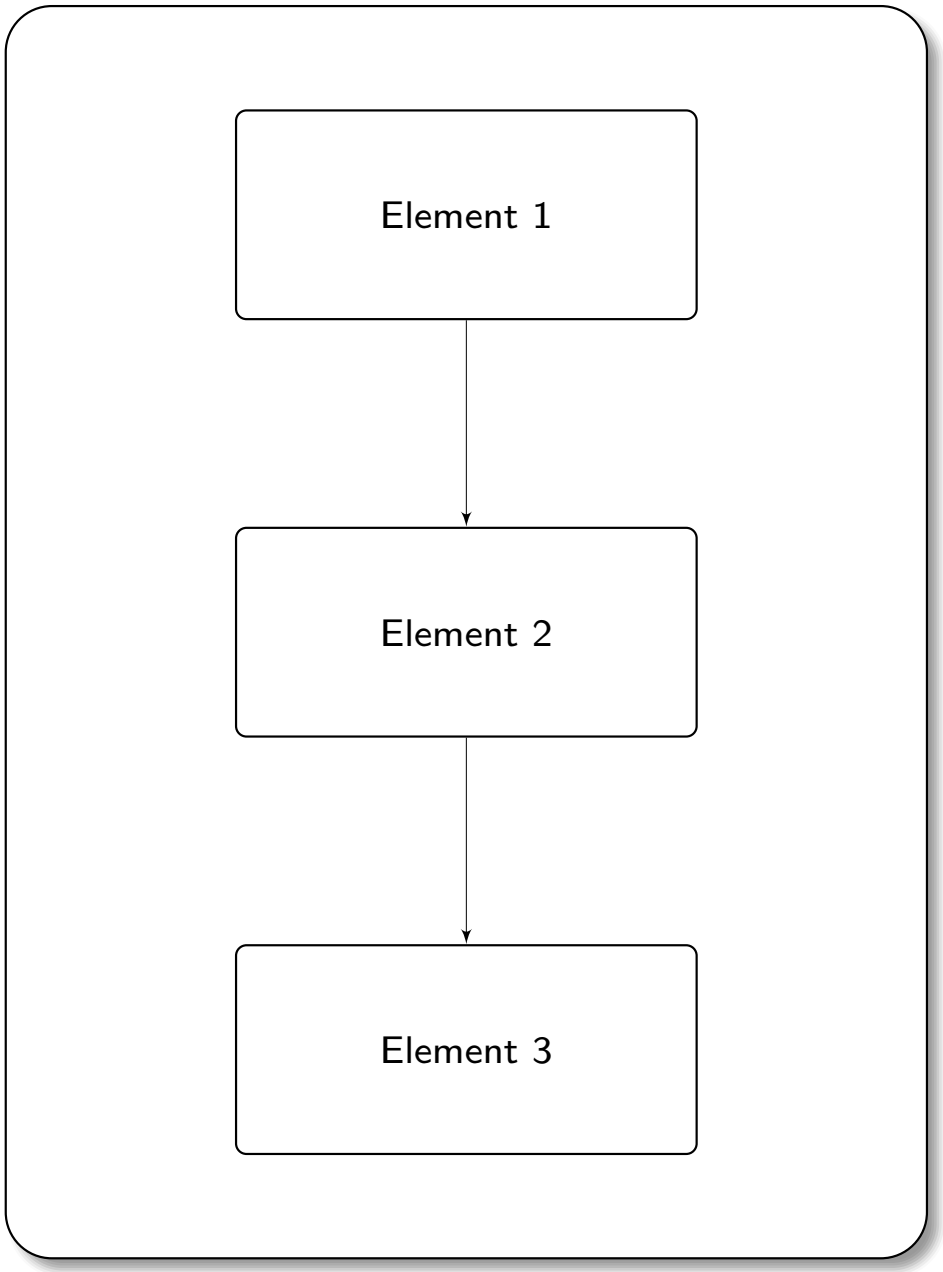


Figure 10: Refine Link

# Options for Refinement of Link

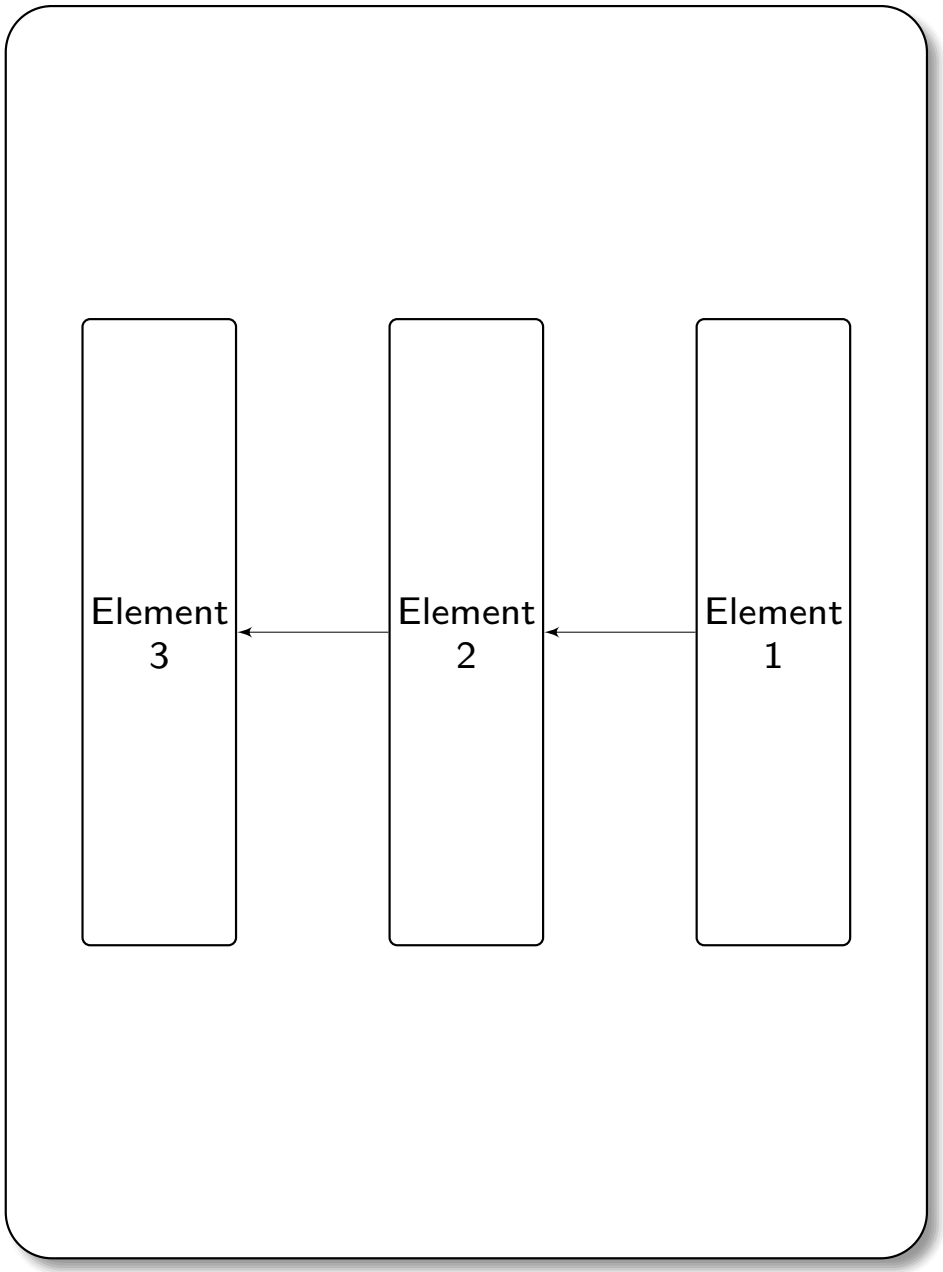
- There are many Options available to Refine The Link
  - "ArrowType" – Type of Arrow
  - "Path" – Path of Link
  - "Width" – Width of Link
  - "ArrowColour" – Colour of Link
  - "Angle" – Angle of Link
  - "StartPoint" – Start Point of Link
  - "EndPoint" – End Point of Link
- The Array Type can also be refined with Latex Options to Double Headed, Line etc..
- The Path of Link determines the Path Link should follow between Cells. It can be refined to any defined Latex Path
- The Width of Link should be specified in pt.
- The Angle of Link is the Angle between the Tangents joining Start and End Points
- The Start and End Points of the Links are Angle in degrees along X-Axis by which the Link should Start and End at the Cells respectively

# Examples – Vertical Array of Connected Elements



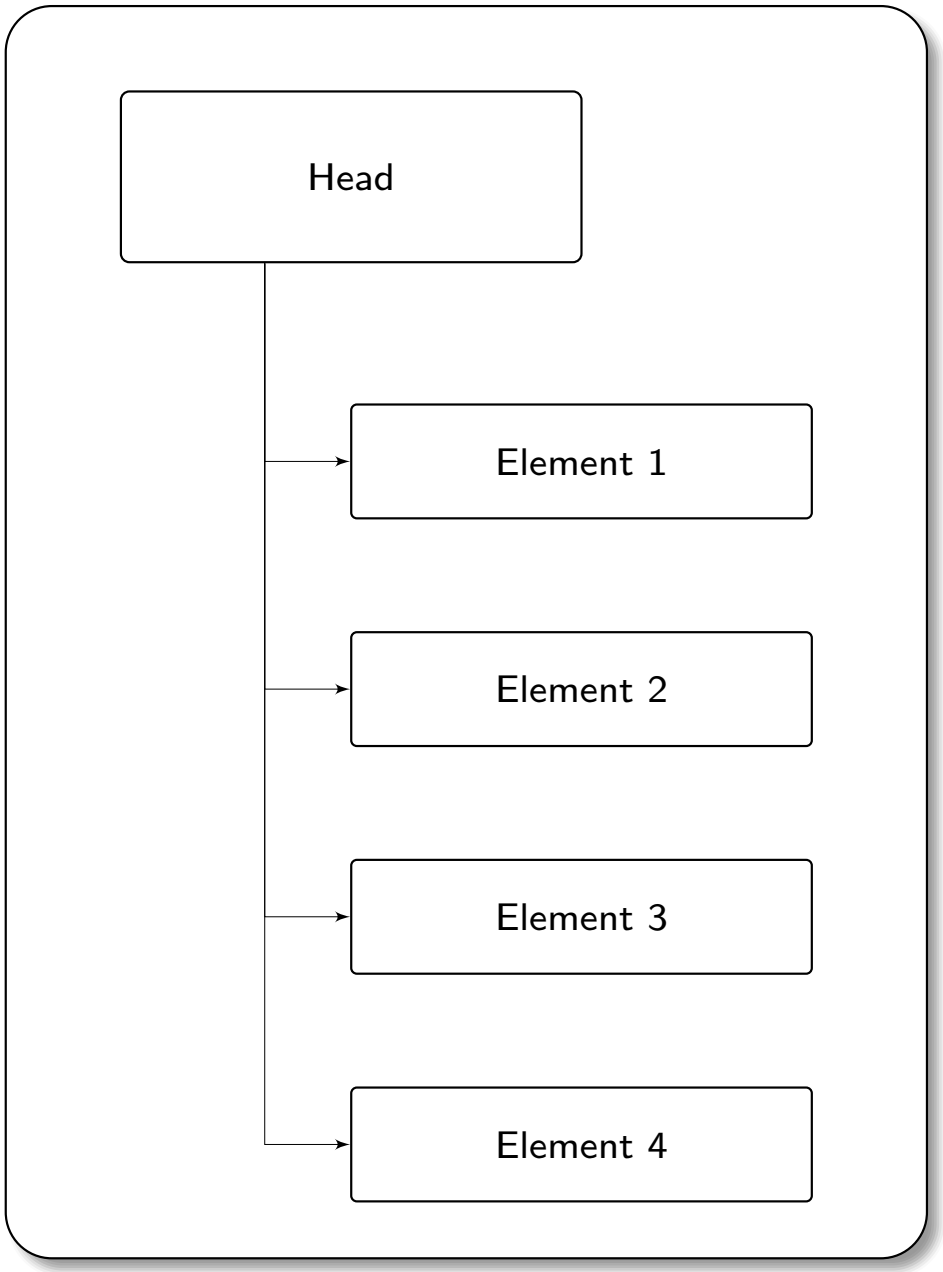

```
1  TYPE(Box) :: Main(1,1), Elements(3,1)
2
3  INTEGER :: i
4
5
6  CALL StartTekzDiag(Main(1,1), Document = A4Page)
7
8  CALL Refine(Main(1,1), CellShape = "None")
9
10 CALL Draw(Main)
11
12
13 CALL Divide(Elements, Main(1,1))
14
15 CALL Refine(Elements, Width = .5, Height = .25)
16 CALL Refine(Elements(1,1), LatexContent = "
       Element 1")
17 CALL Refine(Elements(2,1), LatexContent = "
       Element 2")
18 CALL Refine(Elements(3,1), LatexContent = "
       Element 3")
19
20 CALL Draw(Elements)
21
22 DO i = 1, size(Elements,1) -1
23   CALL DrawLink(Elements(i,1), Elements(i+1, 1))
24 END DO
25
26 CALL EndTekzDiag()
```

# Examples – Horizontal Array of Connected Elements




```
1  TYPE(Box) :: Main(1,1), Elements(1,3)
2
3  INTEGER :: i
4
5
6  CALL StartTekzDiag(Main(1,1), Document = A4Page)
7
8  CALL Refine(Main(1,1), CellShape = "None")
9
10 CALL Draw(Main)
11
12
13 CALL Divide(Elements, Main(1,1))
14
15 CALL Refine(Elements, Width = .5, Height = .75)
16 CALL Refine(Elements(1,1), LatexContent = "
       Element 3")
17 CALL Refine(Elements(1,2), LatexContent = "
       Element 2")
18 CALL Refine(Elements(1,3), LatexContent = "
       Element 1")
19
20 CALL Draw(Elements)
21
22 DO i = size(Elements,2), 2, -1
23 CALL DrawLink(Elements(1,i), Elements(1, i-1))
24 END DO
25
26 CALL EndTekzDiag()
```

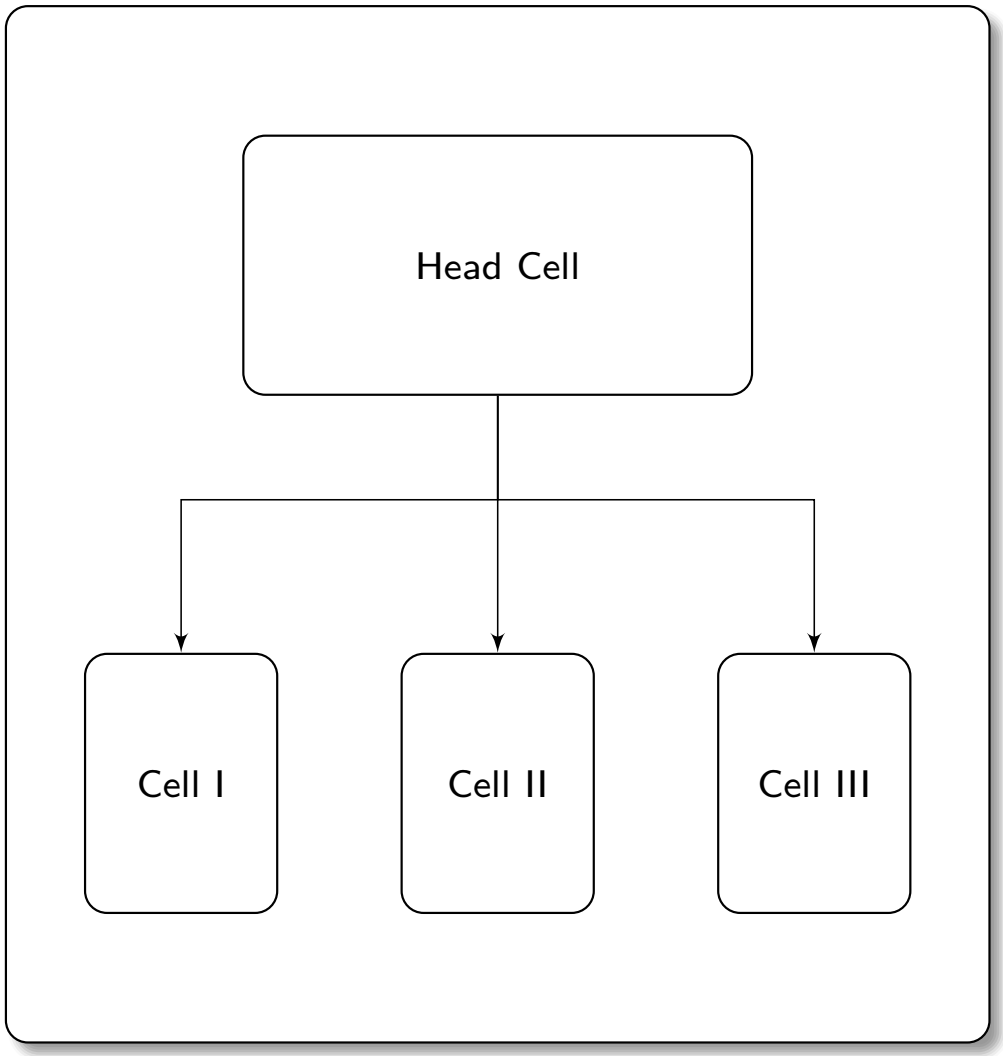
# Examples – Vertical Classification of Elements




```
1  TYPE(Box) :: Main(1,1), Elements(5,1)
2  INTEGER :: i
3
4  CALL StartTekzDiag(Main(1,1), Document = A4Page
       )
5  CALL Refine(Main(1,1), CellShape = "None")
6  CALL Draw(Main)
7
8  CALL Divide(Elements, Main(1,1))
9
10 CALL Refine(Elements, MatrixRatio = RESHAPE
       ((/1.5 , 1., 1., 1., 1./), SHAPE(Elements
       )), Width = .5, Height = .75)
11 CALL Refine(Elements(2:,1),CellXShift = .5)
12 CALL Refine(Elements(1,1),CellXShift = -.5,
       LatexContent = "Head")
13 CALL Refine(Elements(2,1), LatexContent = "
       Element 1")
14 CALL Refine(Elements(3,1), LatexContent = "
       Element 2")
15 CALL Refine(Elements(4,1), LatexContent = "
       Element 3")
16 CALL Refine(Elements(5,1), LatexContent = "
       Element 4")
17
18 CALL Draw(Elements)
19
20 DO i = 2, size(Elements,1)
21 CALL DrawLink(Elements(1,1), Elements(i, 1),
       StartPoint = 225., Path = "|-")
22 END DO
23
24 CALL EndTekzDiag()
```

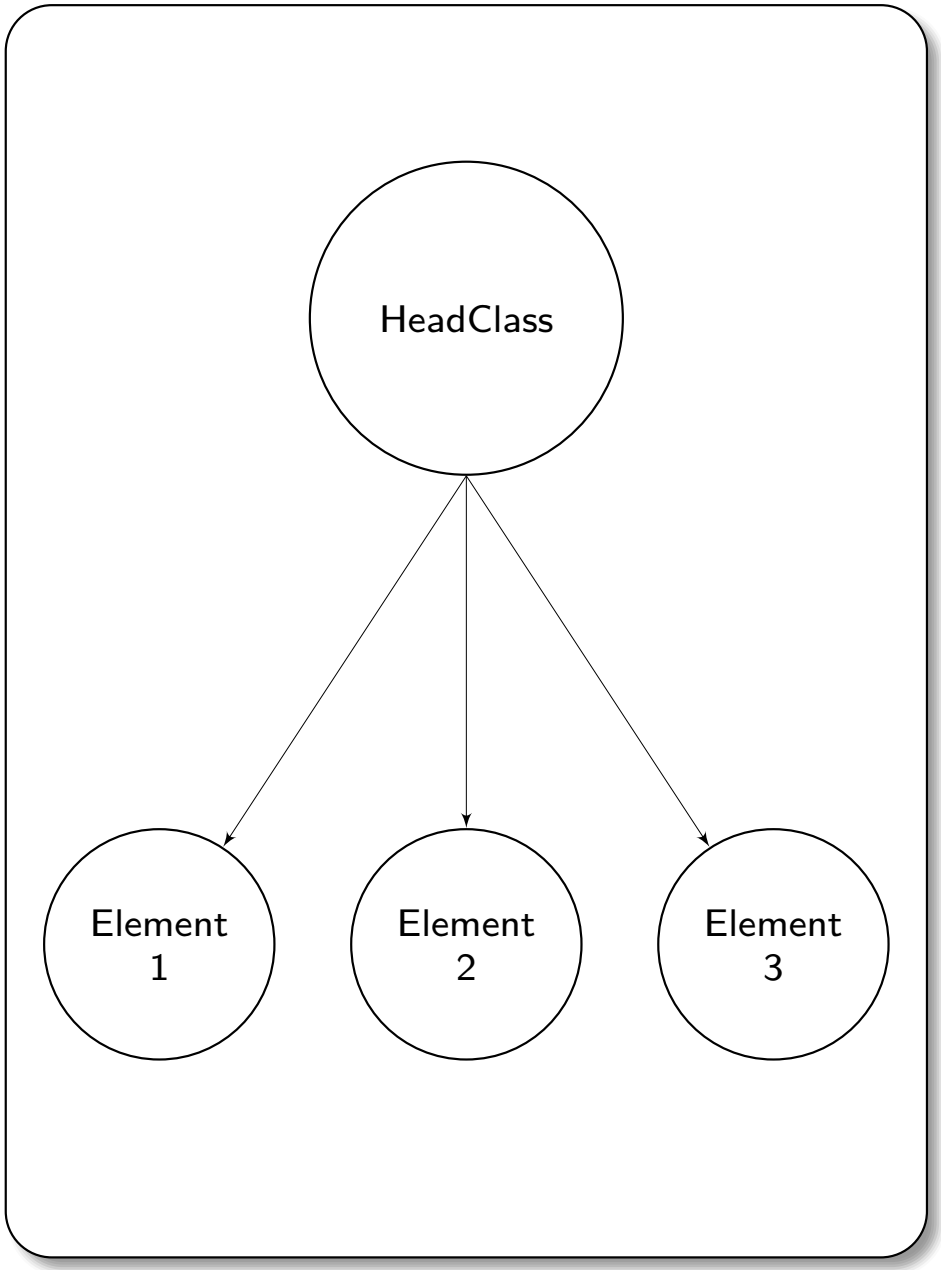
# Example 4

**Diagram**




```
1  TYPE(Box) :: Main(1,1), ElementClasses(2,1),
       ChildElements(1,3)
2
3  INTEGER :: i
4
5
6  CALL StartTekzDiag(Main(1,1), Document = A4Page)
7  CALL Refine(Main(1,1), CellShape = "None")
8  CALL Draw(Main)
9
10 CALL Divide(ElementClasses, Main(1,1))
11 CALL Refine(ElementClasses(1,1), LatexContent =
       "HeadClass", Width = .5, Height = .5)
12 CALL Refine(ElementClasses(2,1), CellShape = "
       None")
13 CALL Draw(ElementClasses)
14
15 CALL Divide(ChildElements, ElementClasses(2,1))
16 CALL Refine(ChildElements, Width = .5, Height =
       .5)
17 CALL Refine(ChildElements(1,1), LatexContent = "
       Element 1")
18 CALL Refine(ChildElements(1,2), LatexContent = "
       Element 2")
19 CALL Refine(ChildElements(1,3), LatexContent = "
       Element 3")
20 CALL Draw(ChildElements)
21
22 DO i = 1, size(ChildElements,2)
23 CALL DrawLink(ElementClasses(1,1), ChildElements
       (1, i), Path = "-- ++(0,-1) -|",
       StartPoint = 270.)
24 END DO
25
26 CALL EndTekzDiag()
```

# Examples – Horizontal Classification of Circular Elements




```
1  TYPE(Box) :: Main(1,1), ElementClasses(2,1),
       ChildElements(1,3)
2
3  INTEGER :: i
4
5
6  CALL StartTekzDiag(Main(1,1), Document = A4Page)
7  CALL Refine(Main(1,1), CellShape = "None")
8  CALL Draw(Main)
9
10
11 CALL Divide(ElementClasses, Main(1,1))
12 CALL Refine(ElementClasses(1,1), LatexContent =
       "HeadClass", CellShape = "Circle", Radius
        = .5)
13 CALL Refine(ElementClasses(2,1), CellShape = "
       None")
14 CALL Draw(ElementClasses)
15
16
17 CALL Divide(ChildElements, ElementClasses(2,1))
18 CALL Refine(ChildElements, CellShape = "Circle",
        Radius= .75)
19 CALL Refine(ChildElements(1,1), LatexContent = "
       Element 1")
20 CALL Refine(ChildElements(1,2), LatexContent = "
       Element 2")
21 CALL Refine(ChildElements(1,3), LatexContent = "
       Element 3")
22 CALL Draw(ChildElements)
23
24 DO i = 1, size(ChildElements,2)
25   CALL DrawLink(ElementClasses(1,1),
       ChildElements(1, i), StartPoint = 270.)
26 END DO
27
28 CALL EndTekzDiag()
```