# **二期项目代码**

创建桥梁分孔线并且输出数据到excel

```c#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

using System.IO;

using MECMOD;

using INFITF;

using KnowledgewareTypeLib;

using HybridShapeTypeLib;

using System.Windows;

namespace Interop.CATCIVAxisDataLib.Code

{

public class CreatAxisFormExcel:MainWindow

{

public void creatAxisAlongCurve(int installtype, INFITF.AnyObject planerCurve, INFITF.AnyObject Curve3d, INFITF.AnyObject StartPoint)

{

ArrayList dislist = new ArrayList();

ArrayList namelist = new ArrayList();

ArrayList ispierlist = new ArrayList();

ArrayList anglelist = new ArrayList();

excel xle = new excel();

dislist=xle.Open(out namelist,out ispierlist,out anglelist);

//for (int i = 1; i < dislist.Count; i++)

//{

// MessageBox.Show(dislist[i].ToString());

//}

Part part = GetContainPart(planerCurve);

HybridShapeFactory hsf = (HybridShapeFactory)part.HybridShapeFactory;

HybridBody AXISHB = part.HybridBodies.Add(); AXISHB.set\_Name("PIER\_AXIS\_SET");

HybridBody hbpts = AXISHB.HybridBodies.Add(); hbpts.set\_Name("AXIS\_PTS");

HybridBody hbref = AXISHB.HybridBodies.Add(); hbref.set\_Name("AXIS\_REF");

HybridBody hb = AXISHB.HybridBodies.Add(); hb.set\_Name("AXIS\_RESULT");

ListParameter axislist = part.Parameters.CreateList("AXISLIST");

//MessageBox.Show("距离数量："+dislist.Count);

for (int i = 0; i < dislist.Count; i++)

{

double dislens = Convert.ToDouble(dislist[i]) \* 1000;

double RoatateAngle = Convert.ToDouble(anglelist[i]);

part.InWorkObject = hb;

HybridShapeProject proj;

if (installtype == 1)

{

HybridShape pts = hsf.AddNewPointOnCurveWithReferenceFromDistance((Reference)planerCurve, (Reference)StartPoint, dislens, false);

StartPoint = pts;

pts.set\_Name(namelist[i].ToString());

hbpts.AppendHybridShape(pts);

// HybridShape pts = hsf.AddNewPointOnCurveFromDistance((Reference)crv, aa, false); pts.Compute();

proj = hsf.AddNewProject((Reference)pts, (Reference)Curve3d); proj.Compute();

proj.SolutionType = 0;

proj.Normal = false;

proj.SmoothingType = 0;

proj.ExtrapolationMode = 0;

proj.Direction = hsf.AddNewDirectionByCoord(0, 0, 1);

proj.set\_Name( namelist[i].ToString());

hbref.AppendHybridShape(proj);

proj.Compute();

AxisSystem pieraxis = Axis(part, (HybridShape)Curve3d, (HybridShape)proj, hb, RoatateAngle);

Parameters parms = part.Parameters.SubList(pieraxis, true);

Parameter distag = parms.CreateInteger("IsPier",Convert.ToInt32(ispierlist[i]));

Parameter angletag = parms.CreateDimension("AxisAngle","ANGLE", Convert.ToInt32(anglelist[i]));

}

else

{

if (i == 0)

{

HybridShape pts = hsf.AddNewPointOnCurveWithReferenceFromDistance((Reference)planerCurve, (Reference)StartPoint, dislens, false);

StartPoint = pts;

// HybridShape pts = hsf.AddNewPointOnCurveFromDistance((Reference)crv, aa, false); pts.Compute();

proj = hsf.AddNewProject((Reference)pts, (Reference)Curve3d); proj.Compute();

proj.SolutionType = 0;

proj.Normal = false;

proj.SmoothingType = 0;

proj.ExtrapolationMode = 0;

proj.Direction = hsf.AddNewDirectionByCoord(0, 0, 1);

proj.set\_Name( namelist[i].ToString());

hbref.AppendHybridShape(proj);

proj.Compute();

AxisSystem pieraxis = Axis(part, (HybridShape)Curve3d, (HybridShape)proj, hb, RoatateAngle);

Parameters parms = part.Parameters.SubList(pieraxis, true);

Parameter distag = parms.CreateInteger("IsPier", Convert.ToInt32(ispierlist[i]));

Parameter angletag = parms.CreateInteger("AxisAngle", Convert.ToInt32(anglelist[i]));

}

else

{

HybridShapeLineTangency tar1 = hsf.AddNewLineTangency((Reference)planerCurve, (Reference)StartPoint, 10, 0, false);

tar1.Compute();

HybridShapeSphere sph1 = hsf.AddNewSphere((Reference)StartPoint, null, dislens, -45, 45, 0, 180);

sph1.Limitation = 1;

sph1.Compute();

HybridShapeIntersection intt1 = hsf.AddNewIntersection((Reference)sph1, (Reference)planerCurve);

intt1.PointType = 0;

intt1.Compute();

HybridShapeNear near1 = hsf.AddNewNear((Reference)intt1, (Reference)tar1);

StartPoint = near1;

near1.Compute();

//hbref.AppendHybridShape(near1);

proj = hsf.AddNewProject((Reference)near1, (Reference)Curve3d); proj.Compute();

proj.SolutionType = 0;

proj.Normal = false;

proj.SmoothingType = 0;

proj.ExtrapolationMode = 0;

proj.Direction = hsf.AddNewDirectionByCoord(0, 0, 1);

proj.set\_Name(namelist[i].ToString());

hbref.AppendHybridShape(proj);

proj.Compute();

proj.set\_Name(namelist[i].ToString());

AxisSystem pieraxis = Axis(part, (HybridShape)Curve3d, (HybridShape)proj, hb, RoatateAngle);

Parameters parms = part.Parameters.SubList(pieraxis, true);

Parameter distag = parms.CreateInteger("IsPier", Convert.ToInt32(ispierlist[i]));

Parameter angletag = parms.CreateInteger("AxisAngle", Convert.ToInt32(anglelist[i]));

}

}

part.InWorkObject = hb;

//xle = null;

}

//释放资源

xle = null;

}

static AxisSystem Axis(Part part, HybridShape crv, HybridShape pt, HybridBody hb, double rotateAngle)

{

HybridShapeFactory hsf = (HybridShapeFactory)part.HybridShapeFactory;

HybridShapeDirection zdir = hsf.AddNewDirection((Reference)part.OriginElements.PlaneXY);

HybridShape zln = hsf.AddNewLinePtDir((INFITF.Reference)pt, zdir, -100, 500, false); zln.Compute();

HybridShape pln = hsf.AddNewPlaneOffsetPt((Reference)part.OriginElements.PlaneXY, (Reference)pt); pln.Compute();

HybridShapeLineTangency tarln = hsf.AddNewLineTangency((Reference)crv, (Reference)pt, -100, 300, false); tarln.Compute();

HybridShapeProject xdirln = hsf.AddNewProject((Reference)tarln, (Reference)pln); xdirln.Compute();

xdirln.SolutionType = 0;

xdirln.Normal = false;

xdirln.SmoothingType = 0;

xdirln.Direction = zdir;

xdirln.ExtrapolationMode = 0;

Reference reference1 = part.CreateReferenceFromObject(xdirln);

HybridShapeRotate hybridShapeRotate1 = hsf.AddNewEmptyRotate();

hybridShapeRotate1.ElemToRotate = reference1;

hybridShapeRotate1.VolumeResult = false;

hybridShapeRotate1.RotationType = 0;

hybridShapeRotate1.Compute();

hybridShapeRotate1.Axis = (Reference)zln;

hybridShapeRotate1.AngleValue = rotateAngle;

hybridShapeRotate1.set\_Name("Roatate\_ln\_"+ Convert.ToString( rotateAngle));

hybridShapeRotate1.Compute();

AxisSystem axis = part.AxisSystems.Add();

axis.OriginType = CATAxisSystemOriginType.catAxisSystemOriginByPoint;

axis.OriginPoint = (Reference)pt;

axis.XAxisType = CATAxisSystemAxisType.catAxisSystemAxisSameDirection;

axis.XAxisDirection = (Reference)hybridShapeRotate1;

axis.YAxisType = CATAxisSystemAxisType.catAxisSystemAxisOppositeDirection;

axis.ZAxisType = CATAxisSystemAxisType.catAxisSystemAxisSameDirection;

axis.ZAxisDirection = (Reference)zln;

part.UpdateObject(axis);

axis.IsCurrent = true;

axis.set\_Name(pt.get\_Name());

part.UpdateObject(axis);

return axis;

}

public static MECMOD.Part GetContainPart(INFITF.AnyObject obj)

{

INFITF.AnyObject a = (INFITF.AnyObject)obj;

INFITF.AnyObject current = a;

bool found = false;

while (found == false)

{

current = (INFITF.AnyObject)current.Parent;

//Console.WriteLine("aaa\_\_\_!" + Microsoft.VisualBasic. Information.TypeName(current));

if (Microsoft.VisualBasic.Information.TypeName(current) == "Part")

{

found = true;

// break;

}

}

return (MECMOD.Part)current;

}

class excel

{

ArrayList dislist = new ArrayList();

// public string Filename;

public ArrayList Open(out ArrayList namelist, out ArrayList IsPierParm,out ArrayList angleList)//打开一个Microsoft.Office.Interop.Excel文件

{

dislist.Clear();

namelist = new ArrayList();

namelist.Clear();

IsPierParm = new ArrayList();

IsPierParm.Clear();

angleList = new ArrayList();

angleList.Clear();

int icount = 1000;

int colnum = 3;

int names = 5;

int ispier = 24;

int angleIndex = 21;

for (int i = 1; i < icount; i++)

{

if (xlsheet.Cells[i + 6, colnum].Value !=null)

{

if (xlsheet.Cells[i + 6, names].Value != null)

{

if (xlsheet.Cells[i + 6, ispier].Value != null)

{

dislist.Add(xlsheet.Cells[i + 6, colnum].Value);

namelist.Add(xlsheet.Cells[i + 6, names].Value);

IsPierParm.Add(xlsheet.Cells[i + 6, ispier].Value);

angleList.Add(xlsheet.Cells[i + 6, angleIndex].Value);

}

}

}

}

return dislist;

}

}

class attribure

{

public void SetAttributeBooleanToObject(MECMOD.Part part, HybridShape obj,string name,Boolean value)

{

Parameters parms = part.Parameters.SubList(obj,true);

Parameter distag = parms.CreateBoolean(name,value);

}

public void GetAttributeBooleanToObject(MECMOD.Part part, HybridShape obj, string name, Boolean value)

{

Parameters parms = part.Parameters.SubList(obj, true);

Parameter distag = parms.CreateBoolean(name, value);

}

}

}

}

```

## 连续梁整体创建

```java

//BeamProduct:ProductOccurrence,PierProduct:ProductOccurrence,AXIS\_LIST:list,ROAD\_CENTER:Curve,ROAD\_PLANER:Curve

/\* Action created by 86156 2019/6/5 \*/

let i(integer)

let F1,F2(DocumentTemplate)

let destination1,destination2(ProductOccurrence)

destination1=BeamProduct

destination2=PierProduct

let axisList(list)

axisList=AXIS\_LIST

let xlsheet(DTSheetType)

let xls(string)

xls=EXCEL

set xlsheet=CreateSheet(xls)

let nb (Integer)

nb = xlsheet.RowsNb

Notify("个数:",nb)

let resName,resExcel(string)

//声明全局变量

let L1length(length)

let R1length(length)

let L2length(length)

let R2length(length)

let PierHeight(length)

let BeamCount(integer)

let GapStr(string)

let DirStr(string)

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ProgressBar creation

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CreateProgressBar("ProgressBar Name")

i=START

for i while i <=END //axisList.Size ()

{

ProgressBarSetValue(100\*i/axisList.Size())

ProgressBarSetText(" ：已完成 共" + floor(END ))

//数据处理

//获得梁数

set BeamCount= xlsheet->CellAsReal(5+i,32)

Notify("梁数量：",BeamCount)

if (i>1)

{

//获得小里程左侧宽度

L1length=xlsheet->CellAsReal((5+(i-1)),22)\*1m

Notify("小里程左侧宽度:",L1length)

//获得小里程右侧宽度

R1length=xlsheet->CellAsReal((5+(i-1)),23)\*1m

Notify("小里程右侧宽度",R1length)

//获得大里程左侧宽度

L2length=xlsheet->CellAsReal(5+i,24)\*1m

Notify("大里程左侧宽度:",L2length)

//获得大里程右侧宽度

R2length=xlsheet->CellAsReal(5+i,25)\*1m

Notify("大里程右侧宽度:",R2length)

//获得边跨

GapStr=xlsheet->CellAsString((5+(i-1)),33)

Notify("获得边跨:",GapStr)

//获得位置设定

DirStr=xlsheet->CellAsString((5+(i-1)),34)

Notify("位置设定:",DirStr)

//实例化上部桥梁

F1=InstantiateTemplate("Re1",destination1.Instance)

F1->SetAttributeObject("P1",axisList.GetItem (i-1))

F1->SetAttributeObject("P2",axisList.GetItem (i))

F1->SetAttributeObject("ROAD\_CENTER",ROAD\_CENTER )

F1->SetAttributeObject("ROAD\_PLANER",ROAD\_PLANER)

//修正数据

F1->SetAttributeDimension("P1\_LW",L1length,"LENGTH")

F1->SetAttributeDimension("P1\_RW",R1length,"LENGTH")

F1->SetAttributeDimension("P2\_LW",L2length,"LENGTH")

F1->SetAttributeDimension("P2\_RW",R2length,"LENGTH")

F1->SetAttributeInteger("BEAM\_COUNT",BeamCount)

//F1->SetAttributeString("GapCheck",GapStr)

F1->SetAttributeString("DirCheck",DirStr)

//F1->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F1)

}

if (i>0)

{

//获得桥墩高度

PierHeight=xlsheet->CellAsReal(5+i,27)\*1m

Notify("PierHeight:",PierHeight)

//获得大里程左侧宽度

L2length=xlsheet->CellAsReal(5+i,22)\*1m

Notify("大里程左侧宽度:",L2length)

//获得大里程右侧宽度

R2length=xlsheet->CellAsReal(5+i,23)\*1m

Notify("大里程右侧宽度:",R2length)

resExcel=xlsheet->CellAsString(5+ i,31)

if resExcel=="连续梁桥-桥台"

{

resName="Res3"

}else if resExcel=="连续梁桥-门架"

{

resName="Res2"

}else if resExcel=="连续梁桥-悬臂盖梁墩"

{

resName="Res1"

}

Notify("下部结构资源名称为：",resName)

//实例化下部桥墩

F2=InstantiateTemplate(resName,destination2.Instance)

F2->SetAttributeObject("AXIS",axisList.GetItem (i))

//修正数据

F2->SetAttributeDimension("PIER\_HEIGHT",PierHeight,"LENGTH")

F2->SetAttributeDimension("WIDTH\_L",L2length-824mm,"LENGTH")

F2->SetAttributeDimension("WIDTH\_R",R2length-824mm,"LENGTH")

//F2->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F2)

}

}

```

简支梁整体创建

```java

//BeamProduct:ProductOccurrence,PierProduct:ProductOccurrence,AXIS\_LIST:list,ROAD\_CENTER:Curve,ROAD\_PLANER:Curve

/\* Action created by 86156 2019/6/5 \*/

let i(integer)

let F1,F2(DocumentTemplate)

let destination1,destination2(ProductOccurrence)

destination1=BeamProduct

destination2=PierProduct

let axisList(list)

axisList=AXIS\_LIST

let xlsheet(DTSheetType)

let xls(string)

xls=EXCEL

set xlsheet=CreateSheet(xls)

let nb (Integer)

nb = xlsheet.RowsNb

Notify("个数:",nb)

let resExcel,resName(String)

//声明全局变量

let L1length(length)

let R1length(length)

let L2length(length)

let R2length(length)

let PierHeight(length)

let BeamCount(integer)

let GapStr(string)

let DirStr(string)

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ProgressBar creation

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CreateProgressBar("ProgressBar Name")

i=START

for i while i <= END //axisList.Size ()

{

ProgressBarSetValue(100\*i/axisList.Size())

ProgressBarSetText(" ：已完成 共" + floor(axisList.Size()))

//数据处理

//获得梁数

set BeamCount= xlsheet->CellAsReal(5+i,32)

Notify("梁数量：",BeamCount)

if (i>1)

{

//获得小里程左侧宽度

L1length=xlsheet->CellAsReal((5+(i-1)),24)\*1m

Notify("小里程左侧宽度:",L1length)

//获得小里程右侧宽度

R1length=xlsheet->CellAsReal((5+(i-1)),25)\*1m

Notify("小里程右侧宽度",R1length)

//获得大里程左侧宽度

L2length=xlsheet->CellAsReal(5+i,24)\*1m

Notify("大里程左侧宽度:",L2length)

//获得大里程右侧宽度

R2length=xlsheet->CellAsReal(5+i,25)\*1m

Notify("大里程右侧宽度:",R2length)

//获得边跨

GapStr=xlsheet->CellAsString((5+(i-1)),33)

Notify("获得边跨:",GapStr)

//获得位置设定

DirStr=xlsheet->CellAsString((5+(i-1)),34)

Notify("位置设定:",DirStr)

//实例化上部桥梁

F1=InstantiateTemplate("Re2",destination1.Instance)

F1->SetAttributeObject("P1",axisList.GetItem (i-1))

F1->SetAttributeObject("P2",axisList.GetItem (i))

F1->SetAttributeObject("ROAD\_CENTER",ROAD\_CENTER )

F1->SetAttributeObject("ROAD\_PLANER",ROAD\_PLANER)

//修正数据

F1->SetAttributeDimension("P1\_LW",L1length,"LENGTH")

F1->SetAttributeDimension("P1\_RW",R1length,"LENGTH")

F1->SetAttributeDimension("P2\_LW",L2length,"LENGTH")

F1->SetAttributeDimension("P2\_RW",R2length,"LENGTH")

F1->SetAttributeInteger("BEAM\_COUNT",BeamCount)

//F1->SetAttributeString("GapCheck",GapStr)

//F1->SetAttributeString("DirCheck",DirStr)

//F1->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F1)

}

//获得桥墩高度

PierHeight=xlsheet->CellAsReal(5+i,27)\*1m

Notify("PierHeight:",PierHeight)

//获得大里程左侧宽度

L2length=xlsheet->CellAsReal(5+i,24)\*1m

Notify("大里程左侧宽度:",L2length)

//获得大里程右侧宽度

R2length=xlsheet->CellAsReal(5+i,25)\*1m

Notify("大里程右侧宽度:",R2length)

resExcel=xlsheet->CellAsString(5+i,31)

if resExcel=="简支梁桥-桥台"

{

resName="Res3"

}else if resExcel=="简支梁桥-门架"

{

resName="ResB"

}else if resExcel=="简支梁桥-悬臂盖梁墩"

{

resName="ResA"

}

Notify("下部结构资源名称为：",resName)

F2=InstantiateTemplate(resName,destination2.Instance)

F2->SetAttributeObject("AXIS",axisList.GetItem(i))

F2->SetAttributeDimension("PIER\_HEIGHT",PierHeight,"LENGTH")

F2->SetAttributeDimension("WIDTH\_L",L2length,"LENGTH")

F2->SetAttributeDimension("WIDTH\_R",R2length,"LENGTH")

//F2->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F2)

}

```

桥梁的创建

```java

//BeamProduct:ProductOccurrence,PierProduct:ProductOccurrence,AXIS\_LIST:list,ROAD\_CENTER:Curve,ROAD\_PLANER:Curve

/\* Action created by 86156 2019/6/5 \*/

let i(integer)

let F1,F2(DocumentTemplate)

let destination1,destination2(ProductOccurrence)

//destination1=BeamProduct

destination2=PierProduct

let axisList(list)

axisList=AXIS\_LIST

let xlsheet(DTSheetType)

let xls(string)

xls="Y:\SHARE\版本2\0827测试数据\连续梁\_总体设计表.xlsx"

set xlsheet=CreateSheet(xls)

let nb (Integer)

nb = xlsheet.RowsNb

Notify("个数:",nb)

//声明全局变量

let L1length(length)

let R1length(length)

let L2length(length)

let R2length(length)

let PierHeight(length)

let PileHeight(length)

let BeamCount(integer)

let GapStr(string)

let DirStr(string)

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ProgressBar creation

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CreateProgressBar("ProgressBar Name")

i=1

for i while i <=axisList.Size ()

{

ProgressBarSetValue(100\*i/axisList.Size())

ProgressBarSetText(" ：已完成 共" + floor(axisList.Size()))

//数据处理

//获得梁数

set BeamCount= xlsheet->CellAsReal(5+i,32)

Notify("梁数量：",BeamCount)

//获得桥墩高度

PierHeight=xlsheet->CellAsReal(5+i,27)\*1m

Notify("PierHeight:",PierHeight)

//获得大里程左侧宽度

L2length=xlsheet->CellAsReal(5+i,24)\*1m

Notify("大里程左侧宽度:",L2length)

//获得大里程右侧宽度

R2length=xlsheet->CellAsReal(5+i,25)\*1m

Notify("大里程右侧宽度:",R2length)

//桩长

PileHeight=xlsheet->CellAsReal(5+i,40)\*1m

Notify("桩长:",PileHeight)

if (i==1)

{

//实例化下部桥墩

F1=InstantiateTemplate("Res3",destination2.Instance)

F1->SetAttributeObject("AXIS",axisList.GetItem (i))

//修正数据

//F1->SetAttributeDimension("PIER\_HEIGHT",PierHeight,"LENGTH")

F1->SetAttributeDimension("P1\_RW",L2length,"LENGTH")

F1->SetAttributeDimension("P1\_LW",R2length,"LENGTH")

F1->SetAttributeInteger("DIRECTION",0)

F1->SetAttributeDimension("PILE\_H",PileHeight,"LENGTH") //F2->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F1)

}

else if (i==axisList.Size ())

{

//实例化下部桥墩

F2=InstantiateTemplate("Res3",destination2.Instance)

F2->SetAttributeObject("AXIS",axisList.GetItem (i))

//修正数据

F2->SetAttributeDimension("P1\_RW",L2length,"LENGTH")

F2->SetAttributeDimension("P1\_LW",R2length,"LENGTH")

F2->SetAttributeInteger("DIRECTION",1)

F2->SetAttributeDimension("PILE\_H",PileHeight,"LENGTH")

//F2->SetAttributeString("跨号" , "跨号-"+ToString(i) )

EndModifyTemplate(F2)

}

}

```

导出整体数据到总体设计表

```java

/\* 工作指令 创建者 liudi 2020/7/12 \*/

let i,j,k,a(Integer )

let myshape,mypart(VPMInstance)

let shapepoint,shapepoint1(PartFeature )

let shapelist,partlist,boxlist,boxlist1,pointlist,planelist(List)

let DT(DTSheetType )

let myname,sheetname,boxname,ptname,outstring,myout(String )

let box,onebox(OpenBodyFeature )

let pt,pt0,pt1,pt2,pt3(Point)

let L,D,x1,x2,x3,y1,y2,y3,z1,z2,z3(LENGTH )

let F(Feature)

let pla(Plane)

shapepoint=GetPLMOwner(myplane)

shapepoint1 = shapepoint.Find("PartFeature", "", true)

SetWorkingContext(shapepoint1 )

onebox=new("OpenBodyFeature" ,"000",shapepoint1)

pt1=new("Point","1",onebox)

pt1=pointonplane(myplane,point(0m,0m,0m),1m,10m)

pt2=new("Point","2",onebox)

pt2=pointonplane(myplane,point(0m,0m,0m),42m,35m)

pt3=new("Point","3",onebox)

pt3=pointonplane(myplane,point(0m,0m,0m),25m,18m)

shapepoint1.Update ()

x1=pt1.coord (1)

y1=pt1.coord (2)

z1=pt1.coord (3)

x2=pt2.coord (1)

y2=pt2.coord (2)

z2=pt2.coord (3)

x3=pt3.coord (1)

y3=pt3.coord (2)

z3=pt3.coord (3)

onebox.Delete ()

set DT=CreateSheet(strFileName)

sheetname="距离(m)"

partlist=Location.Query("VPMInstance",NULL)

shapelist=Location.Query("PartFeature",NULL)

i=1

for i while i<=partlist.Size ()

{

myshape=shapelist[i]

boxlist=myshape.Children

boxlist1=boxlist.Filter("OpenBodyFeature","x.Name == \"EXPORT\_DIM\"")

if boxlist1.Size ()==0

{

break

}

else

{

SetWorkingContext(myshape )

box=boxlist1[1]

pla=new("Plane","000",box)

pla=plane(point(x1,y1,z1),point(x2,y2,z2),point(x3,y3,z3))

pointlist=box.Children

k=pointlist.Size ()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[1]

D=distance(pt,pla)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,60,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[2]

D=distance(pt,pla)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,61,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[2]

pt0=pointlist[3]

D=distance(pt,pt0)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,62,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[3]

D=distance(pt,pla)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,63,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[4]

pt0=pointlist[3]

D=distance(pt,pt0)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,64,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[4]

D=distance(pt,pla)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,65,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pt=pointlist[5]

D=distance(pt,pla)

L=D/1000

outstring=L

a=outstring.Length ()

myout=outstring.Extract (0,a-2)

DT->SetCell(6+i,66,myout)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pla.Delete ()

}

i=i+1

}

```

导出垫石数据

```java

//Location : VPMReference,strFileName : String,ReferAxis : AxisSystem

/\* 工作指令 创建者 liudi 2020/7/21 \*/

let i,j,k,a,a1,b,c,d,e(Integer )

let myshape,mypart(VPMInstance)

let partlist,chlist1,chlist2,chlist3,mmlist1,mmlist2,mylist,mylist1,mylist2,onelist(List)

let DT(DTSheetType )

let box,onebox(OpenBodyFeature )

let partname,outpartname,username,outname,mylength,outmylength,myv,outmyv,x1,y1,z1,x2,y2,z2(String )

let yx1,yx2,yx3,yx4(MMOrderedGeometricalSet)

let mybody,mybody2,body1(BodyFeature )

let myuserfeature(UserFeature )

let sur(Surface )

let pt(Point )

let v,vo,outv(Real )

let xplane(Plane )

let xdir(Direction )

let x,y,z(LENGTH)

let myaxis(AxisSystem )

set DT=CreateSheet(strFileName)

partlist=Location.Query("PartFeature",NULL)

i=1

a1=1

for i while i<=partlist.Size ()

{

mypart=partlist[i]

SetWorkingContext(mypart)

partname=mypart.Name

b=partname.Length ()

outpartname=partname.Extract (0,b-4)

DT->SetCell(a1+1,1,outpartname) //第一栏数据

chlist1=mypart.Children

mmlist1=chlist1.Filter ("MMOrderedGeometricalSet",NULL)

yx1=mmlist1[1]

chlist2=yx1.Children

mmlist2=chlist2.Filter ("MMOrderedGeometricalSet",NULL)

yx2=mmlist2[3]

chlist3=yx2.Children

j=chlist3.Size ()

yx3=chlist3[j]

//body1=new("BodyFeature","0000",yx3)

mylist=yx3.Children

mybody=mylist[1]

mybody2=mylist[2]

mylist1=mybody.Children

mylist2=mybody2.Children

onelist=mylist1+mylist2

k=onelist.Size ()

a=1

for a while a<=k

{

myuserfeature=onelist[a]

username=myuserfeature.Name

c=username.Length ()

outname=username.Extract(0,c-3)

DT->SetCell(a1+1,2,username) //第二栏数据

v=myuserfeature.GetAttributeReal('下垫石体积')

myv=ToString(v)

DT->SetCell(a1+1,6,myv)

v=myuserfeature.GetAttributeReal('上垫石体积')

myv=ToString(v)

DT->SetCell(a1+1,7,myv)

pt=myuserfeature.GetAttributeObject('COORD\_PT')

x=pt.coord (1,ReferAxis)

y=pt.coord (2,ReferAxis)

z=pt.coord (3,ReferAxis)

x1=ToString(x)

c=x1.Length ()

x2=x1.Extract(0,c-2)

y1=ToString(y)

c=y1.Length ()

y2=y1.Extract(0,c-2)

z1=ToString(z)

c=z1.Length ()

z2=z1.Extract(0,c-2)

DT->SetCell(a1+1,3,x2)

DT->SetCell(a1+1,4,y2)

DT->SetCell(a1+1,5,z2)

a1=a1+1

}

}

```

导出基桩数据

```java

//Location : VPMReference,strFileName : String,ReferAxis : AxisSystem

/\* 工作指令 创建者 liudi 2020/7/23 \*/

let i,j,k,a,b(Integer )

let myshape(VPMInstance)

let mypart(PartFeature)

let DT(DTSheetType )

let partlist,shapelist,boxlist,list1,list2,ptlist,yxlist1,yxlist2,yxlist3,bodylist,list3,culist(List )

let yx1,yx2,yx3,yx4(MMOrderedGeometricalSet)

let mybox,box(OpenBodyFeature )

let partname(String )

let pt,mypoint(Point )

let xplane,plane1,myplane(Plane )

let mybody(BodyFeature )

let allcu,mycu(Curve )

let tutai(Surface)

let x,y(LENGTH )

set DT=CreateSheet(strFileName)

partlist=Location.Query("VPMInstance",NULL)

shapelist=Location.Query("PartFeature",NULL)

j=partlist.Size ()

i=1

k=1

for i while i<=j

{

mypart=partlist[i]

myshape=shapelist[i]

partname=mypart.Name ()

DT->SetCell(k+1,1,partname)

SetWorkingContext(myshape)

list1=myshape.Children

list2=list1.Filter ("OpenBodyFeature",NULL)

box=list2[2]

ptlist=box.Children

pt=ptlist[4]

mybox=new("OpenBodyFeature","参照",myshape)

xplane=new("Plane" ,"水平面",mybox)

xplane=plane(point(0mm,0mm,0mm),point(1mm,0mm,0mm),point(0mm,1mm,0mm))

plane1=new("Plane" ,"过渡平面",mybox)

set plane1=planeoffset(xplane,pt)

myplane=new("Plane" ,"相交平面",mybox)

myplane=planeoffset(plane1,-200mm,true)

yxlist1=list1.Filter ("MMOrderedGeometricalSet",NULL)

yx1=yxlist1[1]

yxlist2=yx1.Children

yxlist3=yxlist2.Filter ("MMOrderedGeometricalSet",NULL)

yx2=yxlist3[3]

bodylist=yx2.Children

mybody=bodylist[5]

list3=mybody.Children

allcu=new("Curve" ,"相交线",mybox)

allcu=intersect(myplane,list3[1])

culist=disassemble(allcu,false)

a=culist.Size ()

b=1

for b while b<=a

{

mycu=culist[b]

mypoint=new("Point" ,"点\_"+b,mybox)

mypoint=centerofgravity(mycu)

DT->SetCell(k+1,2,b)

x=mypoint.coord (1,ReferAxis)

y=mypoint.coord (2,ReferAxis)

DT->SetCell(k+1,3,x)

DT->SetCell(k+1,4,y)

k=k+1

}

mybox.Update ()

mybox.Delete ()

}

```

连续梁与简支梁工程模板代码

反转输入条件

```java

set ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Curve.6=invert(ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Sketch.242 )

set ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Curve.7=invert(ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Sketch.243 )

```

中跨位置设定

```java

/\*规则 创建者 DESKTOP 2020/4/26\*/

if `ALL\_BEAMS\PARAMETERS\位置设定` =="中跨"

{

`ALL\_BEAMS\PARAMETERS\边跨判断`=1

}

else

{

`ALL\_BEAMS\PARAMETERS\边跨判断` =0

}

```

位置参数定义

```JAVA

/\*规则 创建者 DESKTOP 2020/4/27\*/

if `ALL\_BEAMS\PARAMETERS\位置设定`=="中跨"

`端封板判断`="ALL"

else if `ALL\_BEAMS\PARAMETERS\位置设定`=="小里程边跨"

`端封板判断`="P2"

else if `ALL\_BEAMS\PARAMETERS\位置设定`=="大里程边跨"

`端封板判断`="P1"

```

```java

/\*规则 创建者 DESKTOP 2020/4/27\*/

let F(UserFeature)

//let l(list)

// l=obj.Query("UserFeature","")

if UDFList.Size()>0

{

for F inside UDFList

{

if `ALL\_BEAMS\PARAMETERS\位置设定`=="小里程边跨"

{

F->SetAttributeInteger("DIRECTION",0)

}

else if `ALL\_BEAMS\PARAMETERS\位置设定`=="大里程边跨"

{

F->SetAttributeInteger("DIRECTION",1)

}

else

{

Notify("梁为中跨箱梁，无需方向判断")

}

}

}

```

梁距离的计算

```java

/\*Rule created by PZG4 4/28/2020\*/

let i,j(Integer)

let ptlist1,ptlist2(List)

let pt1,pt2(point)

let dis1,dis2 (length)

let p1,p2,p3,p4(point)

set ptlist1 =Point1

set ptlist2 =Point2

if ptlist1.Size()>2

{

p1=ptlist1[1]

p2=ptlist1[2]

Notify("DIS:",dis1)

dis1=distance(p1,p2)

`ALL\_BEAMS\PARAMETERS\P1梁距`=dis1

}

if ptlist2.Size()>2

{

p3=ptlist2[1]

p4=ptlist2[2]

Notify("DIS:",dis2)

dis2=distance(p3,p4)

`ALL\_BEAMS\PARAMETERS\P2梁距`=dis2

}

```

### 删除和创建梁集合

```java

If hb1.Bodies.Count <> 0 Then

CATIA.ActiveEditor.Selection.Clear

For k = 1 To hb1.Bodies.Count

Dim hb0obj As Body

Set hb0obj = hb1.Bodies.Item(k)

CATIA.ActiveEditor.Selection.Add hb0obj

Next

CATIA.ActiveEditor.Selection.Delete

End If

Dim i as integer

For i = 1 To icount

Dim hb2 As Body

Set hb2 = hb1.Bodies.Add

hb2.Name = "BEAM\_" + CStr(i)

Next

End Sub

```

更新梁模板

```java

Dim part as Part

Set part =ContainingPart(prt)

part.InWorkObject=prt

On Error Resume Next

Part.UpdateObject(prt)

End Sub

Function ContainingPart(ByVal CatObject)

Dim found As Boolean

Dim Current

Set Current = CatObject

found = False

Do While found = False

Set Current = Current.Parent

If TypeName(Current) = "Part" Then

found = True

End If

Loop

Set ContainingPart = Current

End Function

```

创建小箱梁

```java

//定义变量

let i,j(integer)

let beamcount (integer)

let pt1,pt2,ptA1,pta1,ptA2,pta2(point)

let planerLn(line)

let pl1,pl2,pl3,pl4(plane)

let cu,cu1,cu2,cu3,cu4(curve)

let a1,a2(angle)

let D1,D2(length)

let F(feature)

let p1,p2,drwrailpt1,drwrailpt2(point)

let body1(body)

set p1=ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\p1

set p2=ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\p2

set cu=project(ALL\_BEAMS\INPUT\PLANER\_CRV ,`xy plane` ,direction(`xy plane` ))

//相交出两条曲线

set cu1=`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-1`

set cu2=`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-2`

i=1

beamcount=ALL\_BEAMS\PARAMETERS\BEAM\_COUNT

for i while i<=beamcount

{

pta1=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\POINTS\_1 ,Point11 ,i)

set pta1=pointoncurveRatio(ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Line.8507 ,NULL,(i-1)/(beamcount-1),true)

ptA1=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\POINTS\_1 ,Point1 ,i)

set ptA1=project(pta1 ,`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` ,direction(`xy plane` ))

pta2=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\POINTS\_2 ,Point22 ,i)

set pta2=pointoncurveRatio(ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Line.8508 ,NULL,(i-1)/(beamcount-1),true)

ptA2=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\POINTS\_2 ,Point2 ,i)

set ptA2=project(pta2 ,`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` ,direction(`xy plane` ))

ptA1.Name ="PTA\_1\_"+i

ptA2.Name ="PTA\_2\_"+i

pta1.Name ="PTB\_1\_"+i

pta1.Name ="PTB\_2\_"+i

planerLn=CreateOrModifyDatum("Line",ALL\_BEAMS\DATUM\LINES ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAMS\PlanerLnList ,i)

planerLn=line(pta1,pta2)

planerLn.Color="224,2,32"

planerLn.Name="LN\_"+i

}

j=1

ALL\_BEAMS\RELATIONS\REFERENCE\_RELATIONS\VBScriptNewBody\CREATE\_BODY.Run(ALL\_BEAMS\MODEL\BEAMS\_SETS ,beamcount)

let blist(List)

blist = ALL\_BEAMS\MODEL\BEAMS\_SETS.Query("BodyFeature","")

let beambody(BodyFeature)

for j while j <=beamcount

{

beambody=blist[j]

if (`ALL\_BEAMS\PARAMETERS\边跨判断`=="1")

{

F=CreateOrModifyTemplate("R1" ,beambody ,UDFList ,j)

F->SetAttributeObject("p1",Point1 .GetItem (j))

F->SetAttributeObject("p2", Point2 .GetItem (j))

F->SetAttributeObject("curve1",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` )

F->SetAttributeObject("curve2",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-02`)

F->SetAttributeObject("curve3",ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\MiddleCrvDRV )

//F->SetAttributeInteger("ORIENTATION",1)

}else if (`ALL\_BEAMS\PARAMETERS\边跨判断`=="0")

{

F=CreateOrModifyTemplate("R2" ,beambody ,UDFList ,j)

F->SetAttributeObject("p1",Point1 .GetItem (j))

F->SetAttributeObject("p2", Point2 .GetItem (j))

F->SetAttributeObject("curve1",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` )

F->SetAttributeObject("curve2",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-02`)

F->SetAttributeObject("curve3",ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\MiddleCrvDRV )

}

EndModifyTemplate(F)

F.Name ="梁段\_"+j

}

drwrailpt1=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\OTHER\_PTS ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAMS\DrwRail\_List\_1 ,1)

drwrailpt1=pointoncurveRatio(`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` ,NULL,0,true)

drwrailpt2=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\OTHER\_PTS ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAMS\DrwRail\_List\_1 ,1)

drwrailpt2=pointoncurveRatio(`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` ,NULL,1,true)

Notify("梁段数=",beamcount)

```

更新小箱梁

```java

let p1list,p2list,udflist(List)

let udf,blockudf1,blockudf2 (Feature)

let i(integer)

let dis1,dis2,disa,disb(Length)

set p1list=Point11

set p2list=Point22

set udflist=UDFList

let BeamType(string)

BeamType=`ALL\_BEAMS\PARAMETERS\位置设定`

i=1

set dis1=distance(p1list[1],p1list[2])

set dis2=distance(p2list[1],p2list[2])

for i while i<=p1list.Size()

{

Notify("Distance1:",dis1)

Notify("Distance2:",dis2)

if (BeamType=="中跨")

{

set udf=udflist[i]

if i==1

{

udf.SetAttributeDimension("DIM\_1\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",637mm,"LENGTH")

Notify("修改成功",udf.Name)

}

if i>1 and i< p1list.Size()

{

udf.SetAttributeDimension("DIM\_1\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

if i==p1list.Size()

{

udf.SetAttributeDimension("DIM\_1\_L",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

}

else if (BeamType=="小里程边跨")

{

set udf=udflist[i]

if i==1

{

udf.SetAttributeDimension("DIM\_1\_R",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

if i>1 and i< p1list.Size()

{

udf.SetAttributeDimension("DIM\_1\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

if i==p1list.Size()

{

udf.SetAttributeDimension("DIM\_1\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_L",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",637mm,"LENGTH")

}

}

else if (BeamType=="大里程边跨")

{

set udf=udflist[i]

if i==1

{

udf.SetAttributeDimension("DIM\_1\_L",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_L",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

if i>1 and i< p1list.Size()

{

udf.SetAttributeDimension("DIM\_2\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

}

if i==p1list.Size()

{

udf.SetAttributeDimension("DIM\_2\_L",(dis1-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度`-`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_L",(dis2-`ALL\_BEAMS\PARAMETERS\中梁预制固定宽度` -`ALL\_BEAMS\PARAMETERS\现浇间距` )/2,"LENGTH")

udf.SetAttributeDimension("DIM\_1\_R",637mm,"LENGTH")

udf.SetAttributeDimension("DIM\_2\_R",637mm,"LENGTH")

}

}

}

```

端封板创建

```java

let B1,B2(feature)

let i,j(Integer)

let ptlist1,ptlist2(List)

let pt1,pt2(point)

let dis1,dis2 (length)

let udf1,udf2(Feature)

let surf(surface)

set surf =`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62`

set ptlist1 =Point1

set ptlist2 =Point2

i=1

j=1

if(`端封板判断`=="P1" )//P1

{

for i while i<=ptlist1.Size()

{

B1=CreateOrModifyTemplate("BB" ,ALL\_BEAMS\MODEL\MODEL\_POURING\_BEAM\_P1 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list1 ,i)

B1->SetAttributeObject("PT",ptlist1 .GetItem (i))

B1->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` )

B1->SetAttributeObject("SURFACE",surf)

B1.SetAttributeInteger("ORENTATION",0)

B1.Name="B1-"+i

if i==1

{

B1.SetAttributeInteger("CHECK\_A",1)

B1.SetAttributeInteger("CHECK\_B",0)

}else if (i==ptlist1.Size())

{

B1.SetAttributeInteger("CHECK\_A",0)

B1.SetAttributeInteger("CHECK\_B",1)

}a

EndModifyTemplate(B1)

}

}

else if (`端封板判断`=="P2") //P2

{

for j while j<=ptlist2.Size()

{

B2=CreateOrModifyTemplate("BB" ,ALL\_BEAMS\MODEL\MODEL\_POURING\_BEAM\_P2 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list2 ,j)

B2->SetAttributeObject("PT",ptlist2 .GetItem (j))

B2->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-02` )

B2->SetAttributeObject("SURFACE",surf)

B2.SetAttributeInteger("ORENTATION",1)

B2.Name="B2-"+j

if j==1

{

B2.SetAttributeInteger("CHECK\_A",0)

B2.SetAttributeInteger("CHECK\_B",1)

}else if (j==ptlist2.Size())

{

B2.SetAttributeInteger("CHECK\_A",1)

B2.SetAttributeInteger("CHECK\_B",0)

}

EndModifyTemplate(B2)

}

}

else //all

{

for i while i<=ptlist1.Size()

{

B1=CreateOrModifyTemplate("BB" ,ALL\_BEAMS\MODEL\MODEL\_POURING\_BEAM\_P1 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list1 ,i)

B1->SetAttributeObject("PT",ptlist1 .GetItem (i))

B1->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` )

B1->SetAttributeObject("SURFACE",surf)

B1.SetAttributeInteger("ORENTATION",0)

B1.Name="B1-"+i

if i==1

{

B1.SetAttributeInteger("CHECK\_A",1)

B1.SetAttributeInteger("CHECK\_B",0)

}else if (i==ptlist1.Size())

{

B1.SetAttributeInteger("CHECK\_A",0)

B1.SetAttributeInteger("CHECK\_B",1)

}

EndModifyTemplate(B1)

}

for j while j<=ptlist2.Size()

{

B2=CreateOrModifyTemplate("BB" ,ALL\_BEAMS\MODEL\MODEL\_POURING\_BEAM\_P2 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list2 ,j)

B2->SetAttributeObject("PT",ptlist2 .GetItem (j))

B2->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-02` )

B2->SetAttributeObject("SURFACE",surf)

B2.SetAttributeInteger("ORENTATION",1)

B2.Name="B2-"+j

if j==1

{

B2.SetAttributeInteger("CHECK\_A",0)

B2.SetAttributeInteger("CHECK\_B",1)

}else if (j==ptlist2.Size())

{

B2.SetAttributeInteger("CHECK\_A",1)

B2.SetAttributeInteger("CHECK\_B",0)

}

EndModifyTemplate(B2)

}

}

```

端部现浇部位自动适应

```java

let p1list,p2list,udflist1,udflist2(List)

let udf1,udf2 (UserFeature)

let i,j(integer)

let dis1,dis2,disa,disb(Length)

set p1list=Point11

set p2list=Point22

set udflist1=ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list1

set udflist2=ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_H\list2

let BeamType(string)

let gap(length)

gap=`ALL\_BEAMS\PARAMETERS\现浇间距`/2

BeamType=`ALL\_BEAMS\PARAMETERS\位置设定`

set dis1=distance(p1list[1],p1list[2])

set dis2=distance(p2list[1],p2list[2])

Notify("Name:",dis1)

Notify("Name:",dis2)

i=1

j=1

if udflist1.Size()<>0

{

for i while i<=p1list.Size()

{

udf1=udflist1[i]

if i==1

{

udf1->SetAttributeDimension("Length1",dis1/2-(2676mm/2),"Length")

udf1->SetAttributeDimension("Length2",637mm,"Length")

}

if i>1 and i< p1list.Size()

{

udf1->SetAttributeDimension("Length1",dis1/2-(2676mm/2),"Length")

udf1->SetAttributeDimension("Length2",dis1/2-(2676mm/2),"Length")

}

if i==p1list.Size()

{

udf1->SetAttributeDimension("Length1",637mm,"Length")

udf1->SetAttributeDimension("Length2",dis1/2-(2676mm/2),"Length")

}

}

}

if udflist2.Size()<>0

{

for j while j<=p2list.Size()

{

udf2=udflist2[j]

if j==1

{

udf2->SetAttributeDimension("Length1",637mm,"Length")

udf2->SetAttributeDimension("Length2",dis2/2-(2676mm/2),"Length")

}

if j>1 and j< p2list.Size()

{

udf2->SetAttributeDimension("Length1",dis2/2-(2676mm/2),"Length")

udf2->SetAttributeDimension("Length2",dis2/2-(2676mm/2),"Length")

}

if j==p2list.Size()

{

udf2->SetAttributeDimension("Length1",dis2/2-(2676mm/2),"Length")

udf2->SetAttributeDimension("Length2",637mm,"Length")

}

}

}

```

#### 更新边跨UDF数据

```java

/\*规则 创建者 DESKTOP 2020/7/7\*/

let udf1,udf2(Feature)

if `ALL\_BEAMS\PARAMETERS\位置设定`=="大里程边跨"

{

udf1=UDFList[1]

udf1->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",0)

udf1->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",1)

udf2=UDFList[UDFList.Size()]

udf2->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",1)

udf2->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",0)

}

else if `ALL\_BEAMS\PARAMETERS\位置设定`=="小里程边跨"

{

udf1=UDFList[1]

udf1->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",1)

udf1->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",0)

udf2=UDFList[UDFList.Size()]

udf2->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",0)

udf2->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",1)

}

else

{

udf1=UDFList[1]

udf1->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",1)

udf1->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",0)

udf2=UDFList[UDFList.Size()]

udf2->SetAttributeInteger("SIDEBEAM\_LEFT\_CHECK",0)

udf2->SetAttributeInteger("SIDEBEAM\_RIGHT\_CHECK",1)

}

```

垫石的创建

```java

let B1,B2(feature)

let i,j(Integer)

let ptlist1,ptlist2(List)

let pt1,pt2(point)

let dis1,dis2 (length)

let udf1,udf2(Feature)

let cushionHeight(length)

set ptlist1 =Point1

set ptlist2 =Point2

set cushionHeight=CUSHION\_HEIGHT

i=1

j=1

if(`ALL\_BEAMS\PARAMETERS\位置设定` =="中跨" )//P1

{

for i while i<=ptlist1.Size()

{

B1=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P1 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.1 ,i)

B1->SetAttributeObject("PT",ptlist1.GetItem (i))

B1->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P1` )

B1.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B1.SetAttributeDimension("OFFSET\_X",0mm,"LENGTH")

B1.Name="B1-"+i

B1.SetAttributeDimension("上垫石顺桥向长",750mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B1.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支座顺桥向长",500mm,"LENGTH")

B1.SetAttributeDimension("支座横向宽",500mm,"LENGTH")

B1.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B1.SetAttributeDimension("下垫石顺桥向长",750mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B1.SetAttributeDimension("支座高",110mm,"LENGTH")

B1.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

EndModifyTemplate(B1)

}

for j while j<=ptlist2.Size()

{

B2=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P2 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.2 ,j)

B2->SetAttributeObject("PT",ptlist2 .GetItem (j))

B2->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P2` )

B2.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B2.SetAttributeDimension("OFFSET\_X",0mm,"LENGTH")

B2.SetAttributeDimension("上垫石顺桥向长",750mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B2.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支座顺桥向长",500mm,"LENGTH")

B2.SetAttributeDimension("支座横向宽",500mm,"LENGTH")

B2.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B2.SetAttributeDimension("下垫石顺桥向长",750mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B2.SetAttributeDimension("支座高",110mm,"LENGTH")

B2.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

B2.Name="B2-"+j

EndModifyTemplate(B2)

}

}

else if (`ALL\_BEAMS\PARAMETERS\位置设定` =="大里程边跨") //P2

{

for i while i<=ptlist1.Size()

{

B1=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P1 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.1 ,i)

B1->SetAttributeObject("PT",ptlist1 .GetItem (i))

B1->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P1` )

B1.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B1.SetAttributeDimension("OFFSET\_X",-0mm,"LENGTH")

B1.Name="B1-"+i

B1.SetAttributeDimension("上垫石顺桥向长",750mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B1.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支座顺桥向长",500mm,"LENGTH")

B1.SetAttributeDimension("支座横向宽",500mm,"LENGTH")

B1.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B1.SetAttributeDimension("下垫石顺桥向长",750mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B1.SetAttributeDimension("支座高",110mm,"LENGTH")

B1.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

EndModifyTemplate(B1)

}

for j while j<=ptlist2.Size()

{

B2=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P2 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.2 ,j)

B2->SetAttributeObject("PT",ptlist2 .GetItem (j))

B2->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P2` )

B2.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B2.SetAttributeDimension("OFFSET\_X",-250mm,"LENGTH")

B2.Name="B2-"+j

B2.SetAttributeDimension("上垫石顺桥向长",500mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B2.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支座顺桥向长",400mm,"LENGTH")

B2.SetAttributeDimension("支座横向宽",350mm,"LENGTH")

B2.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B2.SetAttributeDimension("下垫石顺桥向长",500mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B2.SetAttributeDimension("支座高",110mm,"LENGTH")

B2.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

EndModifyTemplate(B2)

}

}

else // 小里程边跨

{

for i while i<=ptlist1.Size()

{

B1=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P1 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.1 ,i)

B1->SetAttributeObject("PT",ptlist1 .GetItem (i))

B1->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P1` )

B1.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B1.SetAttributeDimension("OFFSET\_X",250mm,"LENGTH")

B1.Name="B1-"+i

B1.SetAttributeDimension("上垫石顺桥向长",500mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B1.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支座顺桥向长",400mm,"LENGTH")

B1.SetAttributeDimension("支座横向宽",350mm,"LENGTH")

B1.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B1.SetAttributeDimension("下垫石顺桥向长",500mm,"LENGTH")

B1.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B1.SetAttributeDimension("支座高",110mm,"LENGTH")

B1.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B1.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

EndModifyTemplate(B1)

}

for j while j<=ptlist2.Size()

{

B2=CreateOrModifyTemplate("DIANKUAI" ,ALL\_BEAMS\MODEL\MODEL\_CUSHION\P2 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_CUSHION\List.2 ,j)

B2->SetAttributeObject("PT",ptlist2 .GetItem (j))

B2->SetAttributeObject("AXIS",`ALL\_BEAMS\INPUT\AX-P2` )

B2.SetAttributeDimension("OFFSET\_Z",1610mm,"LENGTH")

B2.SetAttributeDimension("OFFSET\_X",0mm,"LENGTH")

B2.Name="B2-"+j

B2.SetAttributeDimension("上垫石顺桥向长",750mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

//B2.SetAttributeDimension("上垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支座顺桥向长",500mm,"LENGTH")

B2.SetAttributeDimension("支座横向宽",500mm,"LENGTH")

B2.SetAttributeDimension("支座中心间距",750mm,"LENGTH")

B2.SetAttributeDimension("下垫石顺桥向长",750mm,"LENGTH")

B2.SetAttributeDimension("上垫石横向宽",1500mm,"LENGTH")

B2.SetAttributeDimension("支座高",110mm,"LENGTH")

B2.SetAttributeDimension("下垫石高",150mm,"LENGTH")

B2.SetAttributeDimension("支承组合高度",cushionHeight,"LENGTH")

EndModifyTemplate(B2)

}

}

```

创建梁缝隙现浇块

```java

let F(UserFeature)

let i(Integer)

let p1,p2,p3,p4,pa,pb(point)

let ln(curve)

i=1

for i while i<Point1.Size()

{

/\*

set pa=pointbetween(Point1[i],Point1[i+1],0.5,false)

set pa=CreateOrModifyDatum("Point",`ALL\_BEAMS\RESULT\MODEL\_GAP\Ordered Geometrical Set.63` ,`ALL\_BEAMS\RELATIONS\KP\_MODEL\Knowledge Pattern.18\PA\_List` ,i)

pa.Name="Pa\_"+ToString(i)

pa.Update()

set pb=pointbetween(Point2 [i],Point2 [i+1],0.5,false)

set pb=CreateOrModifyDatum("Point",`ALL\_BEAMS\RESULT\MODEL\_GAP\Ordered Geometrical Set.63` ,`ALL\_BEAMS\RELATIONS\KP\_MODEL\Knowledge Pattern.18\PB\_List` ,i)

pb.Name="Pb\_"+ToString(i)

pb.Update()

\*/

ln=CreateOrModifyDatum("Line",ALL\_BEAMS\DATUM\LINES ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_GAP\LineList ,i)

set ln=project(line(pointbetween(Point1[i],Point1[i+1],0.5,false),pointbetween(Point2 [i],Point2 [i+1],0.5,false)),`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` ,direction(0,0,1))

ln.Name="LN-"+i

ln.Update()

F=CreateOrModifyTemplate("BEAM\_GAP",ALL\_BEAMS\MODEL\MODEL\_GAP\Body.8 ,ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAM\_GAP\BEAM\_GAP\_List ,i)

if `ALL\_BEAMS\PARAMETERS\位置设定` =="中跨"

{

F->SetAttributeObject("LINE",ln)

F->SetAttributeObject("LINE1",`ALL\_BEAMS\CONSTRUCTION\DRW\_REF\Ordered Geometrical Set.33\DRW\_INT\_LN` )

F->SetAttributeObject("SURF",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` )

F->SetAttributeString("位置设定","中跨")

}

else if `ALL\_BEAMS\PARAMETERS\位置设定` =="小里程边跨"

{

F->SetAttributeObject("LINE",ln)

F->SetAttributeObject("LINE1",`ALL\_BEAMS\CONSTRUCTION\DRW\_REF\Ordered Geometrical Set.33\DRW\_INT\_LN` )

F->SetAttributeObject("SURF",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` )

F->SetAttributeString("位置设定","小里程边跨")

}

else

{

F->SetAttributeObject("LINE",ln)

F->SetAttributeObject("LINE1",`ALL\_BEAMS\CONSTRUCTION\DRW\_REF\Ordered Geometrical Set.33\DRW\_INT\_LN` )

F->SetAttributeObject("SURF",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\Multi-sections Surface.62` )

F->SetAttributeString("位置设定","大里程边跨")

}

EndModifyTemplate(F)

F.Name="GAP\_"+i

}

```

出图模型

梁平面图

```java

let pt,pt1,pt2(point)

let i(integer)

let ln(line)

let l1,l2(list)

let F(Feature)

l1=Point11

l2=Point22

i=1

for i while i< l1.Size()

{

ln=CreateOrModifyDatum("line",ALL\_BEAMS\DATUM\PLANER\_LN ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_GAP\_BEAM\LineList ,i)

pt1=pointbetween(l1[i],l1[i+1],0.5,false)

pt2=pointbetween(l2[i],l2[i+1],0.5,false)

ln=line(pt1,pt2)

ln.Name="LN-"+i

ln.Update()

Set F=CreateOrModifyTemplate("DRW\_BEAM\_P1",ALL\_BEAMS\MODEL\DRW\_PLANER\_BEAM\_GAP ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_GAP\_BEAM\BeamGapList ,i)

F->SetAttributeObject("LN",ln)

F->SetAttributeObject("CRV-1",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-1` )

F->SetAttributeObject("CRV-2",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-2` )

F->SetAttributeDimension("BEAM\_WIDTH",300mm,"LENGTH")

F->SetAttributeString("TYPE","B")

F.Name="BEAM-PLANER-"+i

//F.Color="0,255,255"

EndModifyTemplate(F)

}

```

平面图文字标注

```java

let pt,pt1(point)

let i,j,k(integer)

let ln(line)

let l1(list)

//

let F(ArcAnnotationOnePointFeature)

let F1(ArcAnnotationTwoPointFeature)

//let DEMO1,

//let DEMO2(feature)

l1=ALL\_BEAMS\RELATIONS\KP\_MODEL\KP\_BEAMS\PlanerLnList

ln=`ALL\_BEAMS\CONSTRUCTION\DRW\_REF\Ordered Geometrical Set.33\DRW\_INT\_LN`

i=1

for i while i<= l1.Size()

{

pt=CreateOrModifyDatum("point",ALL\_BEAMS\DATUM\DRW\_POINTS ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\IntPointList ,i)

pt=intersect(ln,l1[i])

pt.Name="INT-POINT-"+i

}

j=1

for j while j<= l1.Size()

{

/\*Set F=InstantiateTemplate("DRW\_TAG\_1",ALL\_BEAMS\RESULT\DRW\_TAGS )

F->SetAttributeObject("PT",ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\IntPointList [j])

//F->SetAttributeDimension("BEAM\_WIDTH",300mm,"LENGTH")

F->SetAttributeString("ComponentName","demo")

F->SetAttributeString("Name","TEST-"+j)

F->SetAttributeString("Value","K-"+`ALL\_BEAMS\PARAMETERS\跨号` +"\_"+j+"-号梁")

EndModifyTemplate(F)

ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\LabelList->SetItem(F->GetAttributeObject("LABEL"),j)

DEMO1=F->GetAttributeObject("LABEL")

DEMO1.Name="Label-"+j

\*/

F=CreateArcAnnotationOnePointFeature(`Annotation\梁号标注` ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\IntPointList [j],"RTTT","demo","K-"+`ALL\_BEAMS\PARAMETERS\跨号` +"\_"+j+"-号梁")

ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\LabelList->SetItem(F,j)

}

k=1

for k while k< l1.Size()

{

/\*Set F1=InstantiateTemplate("DRW\_DIM\_1",ALL\_BEAMS\RESULT\DRW\_DIM )

F1->SetAttributeObject("P1",ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\IntPointList [k])

F1->SetAttributeObject("P2",ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\IntPointList [k+1])

F1->SetAttributeString("Name","Distance-"+k)

F1->SetAttributeInteger("DimType",3)

EndModifyTemplate(F1)

ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\DimListM ->SetItem(F1->GetAttributeObject("DIM"),k)

DEMO2=F1->GetAttributeObject("DIM")

DEMO2->SetAttributeString("Value","K-"+k+"-号梁")

DEMO2.Name="Dim-M-"+k

\*/

F1=CreateArcAnnotationTwoPointFeature(`Annotation\现浇缝隙标注`,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\IntPointList [k],ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\IntPointList [k+1],"Distance-"+k,"K-"+k+"-号梁","1")

ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\DimListM ->SetItem(F1,k)

}

//得到两边标注

k=1

for k while k< l1.Size()

{

/\*

Set F1=InstantiateTemplate("DRW\_DIM\_1",ALL\_BEAMS\RESULT\DRW\_DIM )

F1->SetAttributeObject("P1",Point11 [k] )

F1->SetAttributeObject("P2",Point11 [k+1])

F1->SetAttributeString("Name","Distance-"+k)

F1->SetAttributeInteger("DimType",3)

EndModifyTemplate(F1)

ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\DimListL ->SetItem(F1->GetAttributeObject("DIM"),k)

DEMO2=F1->GetAttributeObject("DIM")

DEMO2->SetAttributeString("Value","K-"+k+"-号梁")

DEMO2.Name="Dim-L-"+k

\*/

F1=CreateArcAnnotationTwoPointFeature(`Annotation\现浇缝小里程标注`,Point11[k],Point11[k+1],"Distance-"+k,"K-"+k+"-号梁","3")

ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\DimListL ->SetItem(F1,k)

}

//得到两边标注

k=1

for k while k< l1.Size()

{

/\*Set F1=InstantiateTemplate("DRW\_DIM\_1",ALL\_BEAMS\RESULT\DRW\_DIM )

F1->SetAttributeObject("P1",Point22 [k] )

F1->SetAttributeObject("P2",Point22 [k+1])

F1->SetAttributeString("Name","Distance-"+k)

F1->SetAttributeInteger("DimType",3)

EndModifyTemplate(F1)

ALL\_BEAMS\RELATIONS\KP\KP\_DRW\_LABEL\_POSITION\DimList ->SetItem(F1->GetAttributeObject("DIM"),k)

DEMO2=F1->GetAttributeObject("DIM")

DEMO2->SetAttributeString("Value","K-"+k+"-号梁")

DEMO2.Name="Dim-R-"+k \*/

F1=CreateArcAnnotationTwoPointFeature(`Annotation\现浇缝大里程标注`,Point22 [k],Point22[k+1],"Distance-"+k,"K-"+k+"-号梁","3")

ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_LABEL\_POSITION\DimList ->SetItem(F1,k)

}

```

横断面图纸模板的生成

```java

let i,j(integer)

let p1,p2,p3,p4(point)

let dis1,dis2(LENGTH)

let L1,L2(list)

L1=Point11

L2=Point22

let F1,F2(Feature)

let gap(length)

gap=`ALL\_BEAMS\PARAMETERS\现浇间距`/2

i=1

j=1

p1=Point1[1]

p2=Point1[2]

Notify("DIS:",dis1)

dis1=distance(p1,p2)

p3=Point2[1]

p4=Point2[2]

Notify("DIS:",dis2)

dis2=distance(p3,p4)

for i while i<=Point11.Size()

{

F1=CreateOrModifyTemplate("S1" ,ALL\_BEAMS\MODEL\DRW\_SECTION\_UDFS\_P1 ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_BEAM\_SECTION\BeamSectionList\_1 ,i)

F1->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-01` )

F1->SetAttributeObject("PT", Point1.GetItem (i))

if i==1

{

F1->SetAttributeDimension("Length\_L",1975mm,"Length")

F1->SetAttributeDimension("Length\_R",dis1/2-gap,"Length")

}

if i>1 and i< Point11.Size()

{

F1->SetAttributeDimension("Length\_L",dis1/2-gap,"Length")

F1->SetAttributeDimension("Length\_R",dis1/2-gap,"Length")

}

if i==Point11.Size()

{

F1->SetAttributeDimension("Length\_L",dis1/2-gap,"Length")

F1->SetAttributeDimension("Length\_R",1975mm,"Length")

}

EndModifyTemplate(F1)

F2=CreateOrModifyTemplate("S1" ,ALL\_BEAMS\MODEL\DRW\_SECTION\_UDFS\_P2 ,ALL\_BEAMS\RELATIONS\KP\_DRW\KP\_DRW\_BEAM\_SECTION\BeamSectionList\_2 ,i)

F2->SetAttributeObject("CRV",`ALL\_BEAMS\CONSTRUCTION\CONSTRUCTION\CRV-02` )

F2->SetAttributeObject("PT", Point2 .GetItem (i))

if i==1

{

F2->SetAttributeDimension("Length\_L",1975mm,"Length")

F2->SetAttributeDimension("Length\_R",dis2/2-gap,"Length")

}

if i>1 and i< Point11.Size()

{

F2->SetAttributeDimension("Length\_L",dis2/2-gap,"Length")

F2->SetAttributeDimension("Length\_R",dis2/2-gap,"Length")

}

if i==Point11.Size()

{

F2->SetAttributeDimension("Length\_L",dis2/2-gap,"Length")

F2->SetAttributeDimension("Length\_R",1975mm,"Length")

}

EndModifyTemplate(F2)

}

```

梁数量计算器

```cpp

void BrdLogical::mainLoop(double brd\_width)

{

std::cout << "开始计算!\n";

double B = 0;

double maxgap = 4300;//最大梁距

double mingap = 3000;//最小梁距

double stdgap = 300;//现浇段距离

double stdLength = 4250;//标准梁距离

double DS0 = 1975;//边梁外侧宽

double guardrailwidth = 150;//防撞护栏外包宽

double beamTF = 0;//是否处于少梁状态

double NB = 0; //梁数（初始）

double DT = 0.0;//两侧边梁中心距

double beamdistance = 0;

double beamdis3 = 0;

double beamdis4\_3 = 0;

double beamdisTF = 0;

double NB2 = 0;

double DL2 = 0.0;

double DL3 = 0.000;//梁距（修正3）

//B = i\*500;

B = brd\_width;

cout << "" << endl;

cout << "桥宽=" << B << endl;

DT = B - 2 \* guardrailwidth - 2 \* DS0;

cout << "-----两侧边梁中心距=" << DT << endl;

//梁数最小值（估算）；

if (int(B / maxgap) < 2)

{

beamdistance = 2;

}

else

{

beamdistance = (int(B / maxgap));

}

cout << "-----梁数最小值（估算）=" << beamdistance << endl;

//梁数最大值（估算）

if (int(B / mingap) < 2)

{

beamdistance = 2;

}

else

{

beamdistance = (int(B / mingap));

}

cout << "-----梁数最大值（估算）=" << beamdistance << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定是否处于少梁状态（很窄）

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout << "-----\*\*此循环判定是否处于少梁状态（很窄）\*\*" << endl;

//是否处于少梁状态

if (DT < stdLength)

{

beamTF = 1;

}

else

{

beamTF = 0;

}

cout << "-----是否处于少梁状态=" << beamTF << endl;

//梁数（初始）

NB = int(DT / stdLength);

cout << "-----梁数（初始）=" << NB << endl;

//梁数（修正1）

if (beamTF == 1)

{

NB = NB + 2;

}

else

{

NB = NB + 1;

}

cout << "-----梁数（修正1）=" << NB << endl;

//梁距（修正1）

beamdistance = DT / (NB - 1);

cout << "-----梁距（修正1）=" << beamdistance << endl;

//梁距判定 >=3.0

if (beamdistance >= mingap)

{

beamdis3 = 1;

}

else

{

beamdis3 = 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定梁距是否合理

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout << "-----\*\*此循环判定梁距是否合理\*\*" << endl;

cout << "-----梁距判定 >=3.0=" << beamdis3 << endl;

//梁距判定 <=4.3

if (beamdistance <= maxgap)

{

beamdis4\_3 = 1;

}

else

{

beamdis4\_3 = 0;

}

cout << "-----梁距判定 <=4.3=" << beamdis4\_3 << endl;

//梁距是否合理

if (beamdis3 \* beamdis4\_3 == 1)

{

beamdisTF = 1;

}

else

{

beamdisTF = 0;

}

cout << "-----梁距是否合理=" << beamdisTF << endl;

//梁数（修正2）

if (beamdisTF == 1)

{

NB2 = NB;

}

else if (beamdis3 == 0)

{

if (beamdistance < mingap)

{

NB2 = NB;

}

else

{

NB2 = NB - 1;

}

}

else if (beamdis4\_3 == 0)

{

NB2 = NB + 1;

}

cout << "-----梁数（修正2）=" << NB2 << endl;

//梁距（修正2）

DL2 = DT / (NB2 - 1);

cout << "-----梁距（修正2）=" << DL2 << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定边梁外侧宽是否需调整

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//梁距判定 >= 3.0

int beamDisCheck2\_3 = 0;

if (DL2 > mingap)

{

beamDisCheck2\_3 = 1;

}

else {

beamDisCheck2\_3 = 0;

}

cout << "-----\*\*此循环判定边梁外侧宽是否需调整\*\*" << endl;

cout << "-----梁距判定 >= 3.0=" << beamDisCheck2\_3 << endl;

//梁距判定 <= 4.3

int beamDisCheck2\_4\_3 = 0;

if (DL2 <= maxgap)

{

beamDisCheck2\_4\_3 = 1;

}

else

{

beamDisCheck2\_4\_3 = 0;

}

cout << "-----梁距判定 <= 4.3=" << beamDisCheck2\_4\_3 << endl;

//梁距是否合理

int beamdisTF2 = 0;

if (beamDisCheck2\_3 \* beamDisCheck2\_4\_3 == 1)

{

beamdisTF2 = 1;

}

else

{

beamdisTF2 = 0;

}

cout << "-----梁距是否合理=" << beamdisTF2 << endl;

if (beamDisCheck2\_3 == 1)

{

DL3 = DL2;

}

else

{

DL3 = B / NB2;

}

cout << "-----梁距（修正3）=" << DL3 << endl;

//边梁外侧宽（修正1）

double DS01 = 0;

DS01 = (B - (2 \* guardrailwidth) - (NB2 - 1) \* DL3) / 2;

cout << "-----边梁外侧宽（修正1）=" << DS01 << endl;

cout << "\n" << endl;

cout << "计算结束!\n" << endl;

}

void BrdLogical::GetCount(double BW ,int &A,int &B )

{

//declare variables;

double minLength = 3.000;

double maxLength = 4.300;

double stdLength = 4.250;

double DT = 0.000;

double DL1 = 0.000;

MaxCount = int(BW / minLength);

if (MaxCount < 2)

{

MaxCount = 2;

}

else

{

MaxCount = int(BW / minLength);

}

MinCount = int(BW / maxLength);

if (MinCount < 2)

{

MinCount = 2;

}

else

{

MinCount = int(BW / maxLength);

}

DT = BW - (1.975 \* 2 + 0.150 \* 2);

cout << "-----两侧边梁中心距=" << DT << endl;

if (DT < stdLength)

{

cout << "-----梁距不满足一个标准梁4.25米的距离。" << endl;

//计算初始值

Bcount = (DT / stdLength) + 2;

cout << "-----梁数（修正1）=" << Bcount << endl;

}

else

{

//计算初始值

Bcount = (DT / stdLength) + 1;

cout << "-----梁数（修正1）=" << Bcount << endl;

DL1 = DT / (Bcount - 1);

cout << "-----梁距DL1=" << DL1 << endl;

}

//纠正

if (DL1 < minLength)

{

Bcount = (DT / stdLength) + 2;

}

else if (DL1 > minLength && DL1 <= maxLength)

{

Bcount = (DT / stdLength) + 1;

}

else if (DL1 >= maxLength)

{

Bcount = Bcount + 1;

}

cout << "-----梁数（修正2）=" << Bcount << endl;

A = Bcount;

//梁数最大值（估算）的 梁距

double MaxBeamLength = 0;

MaxBeamLength = DT / (MaxCount - 1);

cout << "-----梁数最大值（估算）的 梁距 = " << MaxBeamLength << endl;

//梁数最大值（取用 试算1）

double MaxBeamCount = 0;

if (MaxBeamLength>minLength && MaxBeamLength<=maxLength)

{

MaxBeamCount = MaxCount;// (DT / stdLength) + 2;

//cout << "-----梁数最大值[if (MaxBeamLength<minLength)]（取用 试算1）=" << MaxBeamCount << endl;

}

else if(MaxCount-1>= Bcount)

{

MaxBeamCount = MaxCount-1;

//cout << "-----梁数最大值[(MaxCount-1>= Bcount)]（取用 试算1）=" << MaxBeamCount << endl;

}

else

{

MaxBeamCount = Bcount;

//cout << "-----梁数最大值[else]（取用 试算1）=" << MaxBeamCount << endl;

}

cout << "-----梁数最大值（取用 试算1）=" << MaxBeamCount << endl;

B = MaxBeamCount;

}

void BrdLogical::Compare(double P1Length, double P2Length, int& oCount)

{

BrdLogical aa;

// P1

int CountA = 0;

int CountB = 0;

aa.GetCount(P1Length, CountA, CountB);

cout << "-----最终P1：=" << "桥宽：" << P1Length << "\_" << "A:" << CountA << "\_" << "B:" << CountB << endl;

// P2

int Counta = 0;

int Countb = 0;

aa.GetCount(P2Length, Counta, Countb);

cout << "-----最终P2：=" << "桥宽：" << P2Length << "\_" << "A:" << Counta << "\_" << "B:" << Countb << endl;

//----------------------------------------------------------------------

//交集的类型 1（小里程最小梁数<大里程最小梁数）

int type1 = 0;

if (CountA < Counta && CountB >= Counta)

{

type1 = 1;

}

else

{

type1 = 0;

}

cout << "-----交集的类型 1（小里程最小梁数<大里程最小梁数）：" << type1 << endl;

//交集的类型 2（小里程最小梁数>=大里程最小梁数）

int type2 = 0;

if (CountA >= Counta && CountA >= Countb)

{

type2 = 1;

}

else

{

type2 = 0;

}

cout << "-----交集的类型 2（小里程最小梁数>=大里程最小梁数）：" << type2 << endl;

//----------------------------------------------------------------------

//梁数 是否有交集

int beamintbool = 0;

if (type1==0 && type2==0)

{

beamintbool = 0;

}

else

{

beamintbool = 1;

}

cout << "-----梁数 是否有交集：" << beamintbool << endl;

//----------------------------------------------------------------------

//如有交集 梁数（取公有数的最小值）

int beamGetMin = 0;

if (type1==1)

{

beamGetMin = Counta;

}

else

{

if (type1==0)

{

beamGetMin = CountA;

}

}

cout << "-----梁数 如有交集 梁数（取公有数的最小值）：" << beamGetMin << endl;

// 如无交集 梁数（取四个数中，第二大的那个数）推荐

int noInterVal = 0;

if (beamintbool==0)

{

if (CountA<Counta)

{

noInterVal = Counta;

}

else

{

noInterVal = CountA;

}

}

else

{

noInterVal = beamGetMin;

cout << "-----梁数 查看有交集解：" << noInterVal << endl;

}

cout << "-----梁数 如无交集 梁数（取四个数中，第二大的那个数）推荐：" << noInterVal << endl;

//noInterVal 为最终解

//----------------------------------------------------------------------

//如无交集 需修改桥宽的一侧

//----------------------------------------------------------------------

int NoBeamCheck = 0;

if (beamintbool==0)

{

if (CountA< Counta)

{

cout << "-----梁数 调节小里程侧 " << endl;

}

else

{

cout << "-----梁数 调节大里程侧 " << endl;

}

}

else

{

NoBeamCheck = noInterVal;

cout << "-----梁数 查看有交集解 " << endl;

}

//----------------------------------------------------------------------

// 如无交集 需修改桥宽的一侧 桥宽推荐

//----------------------------------------------------------------------

int beamFixVal = 0;

if (beamintbool == 1)

{

cout << "-----梁数 查看有交集解 " << endl;

}

else

{

cout << "-----梁数 桥宽推荐 " << endl;

}

//以下为其他算法计算结果，可以参考，可以删除；

//----------------------------------------------------------------------

int sresult = 0;

if (type1==1)

{

sresult = Counta;

}

else if ( type2==1)

{

sresult = CountA;

}

else if (beamintbool ==1)

{

if (CountA<Counta)

{

sresult = Counta;

}

else if(CountA>Counta)

{

sresult = CountA;

}

}

else if (beamintbool == 0)

{

if (CountA<Counta)

{

sresult = Counta;

}

else if(CountA > Counta)

{

sresult = CountA;

}

}

else

{

////----------------------------------------------------------------------

vector<int>dArr;

dArr.push\_back(CountA);

dArr.push\_back(CountB);

dArr.push\_back(Counta);

dArr.push\_back(Countb);

int icoutt = GetSmallNumber(dArr);

cout << "最终梁得数量为:" << icoutt << endl;

//////----------------------------------------------------------------------

oCount = icoutt;

}

oCount = sresult;

////----------------------------------------------------------------------

//vector<int>dArr;

//dArr.push\_back(CountA);

//dArr.push\_back(CountB);

//dArr.push\_back(Counta);

//dArr.push\_back(Countb);

//int icoutt = GetSmallNumber(dArr);

//cout << "最终梁得数量为:" << icoutt << endl;

//////----------------------------------------------------------------------

//oCount = icoutt;

//----------------------------------------------------------------------

}

void BrdLogical::run()

{

std::cout << "开始计算!\n";

double B = 0;

double maxgap = 4300;//最大梁距

double mingap = 3000;//最小梁距

double stdgap = 300;//现浇段距离

double stdLength = 4250;//标准梁距离

double DS0 = 1975;//边梁外侧宽

double guardrailwidth = 150;//防撞护栏外包宽

double beamTF = 0;//是否处于少梁状态

double NB = 0; //梁数（初始）

double DT = 0.0;//两侧边梁中心距

double beamCount = 0;

double beamdis3 = 0;

double beamdis4\_3 = 0;

double beamdisTF = 0;

double NB2 = 0;

double DL2 = 0.0;

double DL3 = 0.000;//梁距（修正3）

for (int i = 0; i < 150; i++)

{

B = i \* 500;

cout << "" << endl;

cout << "桥宽=" << B << endl;

DT = B - 2 \* guardrailwidth - 2 \* DS0;

cout << "-----两侧边梁中心距=" << DT << endl;

//梁数最小值（估算）；

if (int(B / maxgap) < 2)

{

beamCount = 2;

}

else

{

beamCount = (int(B / maxgap));

}

cout << "-----梁数最小值（估算）=" << beamCount << endl;

//梁数最大值（估算）

if (int(B / mingap) < 2)

{

beamCount = 2;

}

else

{

beamCount = (int(B / mingap));

}

cout << "-----梁数最大值（估算）=" << beamCount << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定是否处于少梁状态（很窄）

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout << "-----\*\*此循环判定是否处于少梁状态（很窄）\*\*" << endl;

//是否处于少梁状态

if (DT < stdLength)

{

beamTF = 1;

}

else

{

beamTF = 0;

}

cout << "-----是否处于少梁状态 =" << beamTF << endl;

//梁数（初始）

NB = int(DT / stdLength);

cout << "-----梁数（初始）=" << NB << endl;

//梁数（修正1）

if (beamTF == 1)

{

NB = NB + 2;

}

else

{

NB = NB + 1;

}

cout << "-----梁数（修正1）=" << NB << endl;

//梁距（修正1）

beamCount = DT / (NB - 1);

cout << "-----梁距（修正1）=" << beamCount << endl;

//梁距判定 >=3.0

if (beamCount >= mingap)

{

beamdis3 = 1;

}

else

{

beamdis3 = 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定梁距是否合理

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout << "-----\*\*此循环判定梁距是否合理\*\*" << endl;

cout << "-----梁距判定 >=3.0=" << beamdis3 << endl;

//梁距判定 <=4.3

if (beamCount <= maxgap)

{

beamdis4\_3 = 1;

}

else

{

beamdis4\_3 = 0;

}

cout << "-----梁距判定 <=4.3=" << beamdis4\_3 << endl;

//梁距是否合理

if (beamdis3 \* beamdis4\_3 == 1)

{

beamdisTF = 1;

}

else

{

beamdisTF = 0;

}

cout << "-----梁距是否合理=" << beamdisTF << endl;

//梁数（修正2）

if (beamdisTF == 1)

{

NB2 = NB;

}

else if (beamdis3 == 0)

{

if (beamCount < mingap)

{

NB2 = NB;

}

else

{

NB2 = NB - 1;

}

}

else if (beamdis4\_3 == 0)

{

NB2 = NB + 1;

}

cout << "-----梁数（修正2）=" << NB2 << endl;

//梁距（修正2）

DL2 = DT / (NB2 - 1);

cout << "-----梁距（修正2）=" << DL2 << endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*此循环判定边梁外侧宽是否需调整

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//梁距判定 >= 3.0

int beamDisCheck2\_3 = 0;

if (DL2 > mingap)

{

beamDisCheck2\_3 = 1;

}

else {

beamDisCheck2\_3 = 0;

}

cout << "-----\*\*此循环判定边梁外侧宽是否需调整\*\*" << endl;

cout << "-----梁距判定 >= 3.0=" << beamDisCheck2\_3 << endl;

//梁距判定 <= 4.3

int beamDisCheck2\_4\_3 = 0;

if (DL2 <= maxgap)

{

beamDisCheck2\_4\_3 = 1;

}

else

{

beamDisCheck2\_4\_3 = 0;

}

cout << "-----梁距判定 <= 4.3=" << beamDisCheck2\_4\_3 << endl;

//梁距是否合理

int beamdisTF2 = 0;

if (beamDisCheck2\_3 \* beamDisCheck2\_4\_3 == 1)

{

beamdisTF2 = 1;

}

else

{

beamdisTF2 = 0;

}

cout << "-----梁距是否合理=" << beamdisTF2 << endl;

if (beamDisCheck2\_3 == 1)

{

DL3 = DL2;

}

else

{

DL3 = B / NB2;

}

cout << "-----梁距（修正3）=" << DL3 << endl;

//边梁外侧宽（修正1）

double DS01 = 0;

DS01 = (B - (2 \* guardrailwidth) - (NB2 - 1) \* DL3) / 2;

cout << "-----边梁外侧宽（修正1）=" << DS01 << endl;

//梁数最小值（估算）的 梁距

}

//cout << "\_groupMaxCount 数组的数量是：" << \_groupMaxCount.Size() << endl;

//cout << "\_groupMinCount 数组的数量是：" << \_groupMinCount.Size() << endl;

cout << "\n" << endl;

cout << "计算结束!\n" << endl;

}

bool BrdLogical::sortFun(const int& p1, const int& p2)

{

return p1 < p2;

}

int BrdLogical::GetSmallNumber(vector<int> dArr)

{

vector<int> dArrVec;

for (int i = 0; i < dArr.size(); i++)

{

int dOne = dArr[i];

for (int j = i + 1; j < dArr.size(); j++)

{

if (dOne == dArr[j])

{

dArrVec.push\_back(dOne);

cout << "ONE:" << dOne << endl;

}

}

}

sort(dArrVec.begin(), dArrVec.end(), sortFun);

if (dArrVec.size() <= 0)

{

return 0;

}

cout << dArrVec[0] << endl;

return dArrVec[0];

}

```