**<14j>小组<2016火车调度模拟系统>详细设计（图形化）**

**版本号：14j-Details-01**

**编制时间：2016年6月20日**

**编制人员：罗暄澍 谢天 刘润程**

1. **模块1<Main>**

**1.1函数1**

**函数名称：**int main()

**函数参数说明：**无

**函数功能：** 定义变量

轨道参数

火车参数

完善参数

确认信息

执行策略

**函数局部变量设计:** TrainInfo train[5];

int i;

int run\_method;

FILE \*setting;

int temp;

char tempinput;

**函数算法逻辑：**

当默认数据为空时

输出不能打开文件

exit(0);

否则for循环0到4

fread(&train[temp],sizeof(TrainInfo),1,setting);

fclose(setting);

调用函数SetDefaultSet(train);

制图

for循环1到3

调用函数OutputSet(train,temp);

选择是否载入默认设置

如果载入默认设置

LoadInfo(&train[1],&train[2],&train[3],1);

完善参数调用函数CompleteInfo(&train[1],&train[2],&train[3]);

确认信息

如果当前数据为空

输出不能打开文件;

exit(0);

否则for循环1到3

调用函数FileWrite(train,setting,temp);

fclose(setting);

如果输出数据为空

输出不能打开文件;

exit(0);

fclose(setting);

选择执行策略

执行策略为1时

调用函数Creatmap(train);

输出当前使用的是快车先行策略

调用函数AllControl(train);

返回 0;

**2.模块2<** CreateOrbit **>**

**2.1函数1**

**函数名称：**void LoadInfo(TrainInfo\* train1,TrainInfo\* train2,TrainInfo\* train3)

**函数参数说明：**3辆火车的初始信息

**函数功能：**输入轨道参数、作图、输出数据、赋值

**函数局部变量设计： i**nt Y2,X3;

int L1,W1,L2,W2,L3,W3;

int a[100][100]={0};

int i,j,k;

int state;

**函数算法逻辑:**

输入轨道1参数 scanf输入长和宽

While长宽不符合要求时

重新输入长宽

输入轨道2参数 scanf输入;

While输入不合适时 重新输入

Scanf输入长和宽;

While长宽不合适 scanf输入state

While state不等于0和1时 重新输入 当state==0) 输入长和宽;

If状态1时 输入长

whileY2不合适时 输入长

重新输入长和宽

输入轨道3参数（同轨道2）

For循环j=0到L a[j][0]=1; a[j][W1]=1;

For循环j=L1到j=L1+L2 a[j][Y2]=1; a[j][Y2+W2]=1; for循环j=X3到j=X3+L3 a[j][W1+W3]=1;

j=0;

for循环k=0到k=w1 a[0][k]=1; a[L1][k]=1;

for循环k=Y2到k=Y2+W2 a[L1+L2][k]=1;

for循环k=W1到k=W1+W3 a[X3][k]=1; a[X3+L3][k]=1;

for循环j=0到j=L1+L2

for循环k=0到k=W1+W3

if当a[j][k]=0时 输出空格

否则 if当a[j][k]=1时 输出图形□

/\*当CreatMap=0时

L1=火车1轨道长

W1=火车1轨道宽

L2=火车2轨道长

W2=火车2轨道宽

L3=火车3轨道长

W3=火车3轨道宽

Y2=火车2的y轴基点坐标;

X3=火车3二维x坐标;

/\* for循环j=0到j=L1

赋值a[j][0]=1;

赋值a[j][W1]=1;

for循环j=L1到j=L1+L2

赋值a[j][Y2]=1;

赋值a[j][Y2+W2]=1;

for循环j=X3到j=X3+L3

赋值a[j][W1+W3]=1;

赋值j=0;

For循环k=0到k=W1

赋值a[0][k]=1;

赋值a[L1][k]=1;

for循环k=Y2到k=Y2+W2

赋值a[L1+L2][k]=1;

for循环k=W1到k=W1+W3

赋值a[X3][k]=1;

赋值a[X3+L3][k]=1;

for循环j=0到j=L1+L2

for循环k=0到k<=W1+W3

当a[j][k]==0时

输出空格

当a[j][k]==1时

输出□

数据

赋值

**3.模块3<** Length to Coordinate **>**

**3.1函数1**

**函数名称：**Position Length\_To\_Coordinate(TrainInfo a)

函数参数说明：火车运行长度

函数功能：将长度换为火车坐标。

函数局部变量设计： int mod;

Position train\_position;

函数算法逻辑：

求距离除以周长的余数

If余数大于0且小于等于长

X坐标=x+余数

Y坐标=y

if余数大于长且小于等于长+宽

余数-=长

X坐标=x+长

y坐标=y+余数

if余数大于长+宽且<=2倍的长+一倍的宽

余数=长+宽

X坐标=x+长-余数

y坐标=y+宽

否则

余数=2倍的长+1倍的宽

X坐标=x

y坐标=y+宽-余数

返回坐标

**4.模块4<** TrainFile **>**

**4.1函数1**

**函数名称：**void SetDefaultSet(TrainInfo set[])

**函数参数说明：**默认数据集合

**函数功能：**默认数据输入

**函数局部变量设计：**无

**函数算法逻辑：**默认数据输入

**4.2函数2**

**函数名称：**void OutputSet(TrainInfo set[],int order)

**函数参数说明：**默认数据集合、A,B or C车

**函数功能：**输出运行速度，起始时间，运行方向，起始位置，停靠时间和类型

**函数局部变量设计：无**

**函数算法逻辑：**

switch(order)

case1 A车输出运行速度，起始时间，运行方向，起始位置，停靠时间和类型；break；

case2 B车输出运行速度，起始时间，运行方向，起始位置，停靠时间和类型

break；

case3 C车输出运行速度，起始时间，运行方向，起始位置，停靠时间和类型

break；

**4.3函数3**

**函数名称：**void InputSet(TrainInfo set[],int order)

**函数参数说明：**默认数据集合、A,B or C车

**函数功能：**默认数据的输入，快车慢车的储存以及运行速度的判断

**函数局部变量设计：** char carname,tempinput;

int fasttypeset[3],slowtypeset[3];

int fasttype=0,slowtype=0,a;

int tempinputflag=0;

**函数算法逻辑：**

当order=1时

为A车

当order=2时

为B车

否则

为C车

For循环(; tempinputflag==0;)

输入车的类型(1代表快车，2代表慢车)

当类型为慢车时

慢车储存

slowtype++;

当类型为慢车时

快车储存

fasttype++;

当类型既不是快车又不是慢车时

输出错误输入

continue;

输入车运行速度(快车速度需大于2,慢车速度需小于快车,速度不可大于等于轨道长度

当速度小于0或大于轨道长度时

输出错误输入

continue;

否则

当类型为快车时

如果速度小于2

输出错误输入

continue;

for循环a=0到 a=slowtype-1

当快车速度小于慢车速度时

输出错误输入

continue;

否则

当类型为慢车时

For循环a=0到a=fasttype-1

当慢车速度大于快车速度时

输出错误输入

continue;

输入车起始时间

当起始时间小于0时

输出错误输入

continue;

输入车运行方向(顺时针为1,逆时针为2

当运行方向既不是1又不是2时

输出错误输入

continue;

输入车起始位置(相对于参考点的位移,顺时针为正,且小于轨道总长度

当起始位置小于0或大于轨道总长度时

输出错误输入

continue;

输入车停靠时间

当停靠时间小于0时

输出错误输入

continue;

调用函数OutputSet(set,order);

getchar();

输入是否输入正确

当输入正确时

赋值tempinputflag=1;

**5.模块5<** EnsureInfo **>**

**5.1函数1**

**函数名称：**void EnsureInfo(TrainInfo\* train)

**函数参数说明：**火车信息

**函数功能：**输出信息（车号/轨道号。快慢，方向轨道基准点，启动位置，探测点，坐标，启动时间，速度，停靠时间，所在轨道）

**函数局部变量设计：**无

**函数算法逻辑：**

输出车号/轨道号

if当车的类型为1时 输出快车

if当车的类型为2时 输出慢车

if当车的方向为1时 输出顺时针

if当车的方向为2时 输出逆时针

输出轨道基准点坐标

输出启动位置

当(\*train)车数字==1)输出探测点1、2、3、4及一维坐标

否则输出探测点1、2及一维坐标

输出启动时间

输出火车速度

输出停靠时间

输出所在轨道

**6.模块6<** CompleteInfo **>**

**6.1函数1**

**函数名称：**void CompleteInfo(TrainInfo\* train1,TrainInfo\* train2,TrainInfo\* train3)

**函数参数说明：**3辆火车的初始信息

**函数功能：**指针指向重合轨道火车，确定正方向，初始位置确定，初始化状态

**函数局部变量设计：**int mod; int inPub

**函数算法逻辑：**

求余数

if余数大于等于0且小于等于长

二维坐标x=x+余数;

二维坐标y=y;

if余数大于长且小于等于长+宽

余数-=长;

二维坐标x=x+长;

二维坐标y=y+余数;

if余数大于长+宽且<=2倍的长+1倍的宽

余数-=长+宽;

二维坐标x=x+长-余数

二维坐标y=y+宽;

否则

余数-=2倍的长+1倍的宽;

二维坐标x=x;

二维坐标y=y+宽-余数;

train2,train3同train1

判断火车方向信息

指针指向重合轨道火车

正方向为顺时针

初始位置确定

初始化状态

next预处理

inPub=isBothInPub((\*train1),(\*train2));

while循环(inPub==1)

重新输入AB的初始位置

While 循环 火车1距离小于0或火车2距离小于0

重新输入AB的位置

inPub=isBothInPub((\*train1),(\*train2));

inPub=isBothInPub((\*train1),(\*train3));

while循环(inPub==1)

重新输入AC的初始位置

While 循环 火车1距离小于0或火车3距离小于0

重新输入AC的初始位置")

inPub=isBothInPub((\*train1),(\*train3));

火车初始时间

当IsInPubluic((\*train1))=0且IsNextInPubluic(\*train1)=0 时

火车1状态为0

当IsInPubluic((\*train2))=0且IsNextInPubluic(\*train2)==0时

火车2状态为0

当IsInPubluic((\*train3))=0且IsNextInPubluic(\*train3)=0时

火车3状态为0

当IsInPubluic((\*train1))=1且火车1距离=火车1站台1或火车1距离=火车1站台2时

火车1状态为2

当IsInPubluic((\*train2))=1且火车2距离=火车2站台1时

火车2状态为2

当IsInPubluic((\*train3))= 1且火车3距离=火车3站台1时

火车3状态为2

当IsInPubluic((\*train1))=1且((\*train1)车距离<(\*train1).station1且IsNextInPubluic(\*train1)=1或IsNextInPubluic(\*train1)=1且火车1距离>火车1站台1且火车1距离<火车1站台2时

火车1状态为3

当IsInPubluic((\*train2))=1且火车2距离<火车2站台1且sNextInPubluic(\*train2)=1时

火车2状态为3

当IsInPubluic((\*train3))=1且火车3距离<火车3站台1且IsNextInPubluic(\*train3)=1时

火车3状态为3

当IsInPubluic((\*train1))=1且火车1距离>火车1站台1或火车1距离>火车1站台2或(IsInPubluic((\*train1))=1且IsNextInPubluic(\*train1)==0))时

火车1状态为4

当IsInPubluic((\*train2))=1且火车2距离>火车2站台1 或((IsInPubluic((\*train2))=1且IsNextInPubluic(\*train2)==0)))时

火车2状态为4

当IsInPubluic((\*train3))=1且火车3距离>火车3站台1或 ((IsInPubluic((\*train3))=1且IsNextInPubluic(\*train3)=0))时

火车3状态为4

火车1初始时间不为0时

火车1状态为5

火车2初始时间不为0时

火车2状态为5

火车3初始时间不为0时

火车3状态为5

当IsInPubluic((\*train1))=0求IsNextInPubluic(\*train1)=1且火车1状态不为5且且火车2状态不为5且火车3状态不为5时

当火车1方向为1与2时

当IsInPubluic((\*train2))=0且IsNextInPubluic(\*train2)=1时

当火车1速度>火车2速度时

火车1状态为3

火车2状态为1

否则

火车1状态为1

火车2状态为3

否则

当火车2状态不为2且火车2状态不为3且火车2状态不为4时

火车1状态为2

否则

当火车1运行方向为1且与3 时

当IsInPubluic((\*train3))=0且IsNextInPubluic(\*train3)=1时

当火车1速度>火车3速度时

火车1状态为3

火车3状态为1

否则

火车1状态为1

火车3状态为3

否则

当火车3状态不为2且火车3状态不为3且火车3状态不为4

火车1状态不为2

否则 火车1运行方向为逆时针逆时针与2

当IsInPubluic((\*train2))=0且IsNextInPubluic(\*train2)=1时

当火车1速度>火车2速度时

火车1状态为3

火车2状态为1

否则

火车1状态为1

火车2状态为3

否则

当火车2状态不为2且火车2状态不为3且火车2状态不为4

火车1状态为2

否则当火车1运行方向逆时针与3

当IsInPubluic((\*train3))=0且IsNextInPubluic(\*train3)=1时

当火车1速度>火车3速度时

火车1状态为3

火车3状态为1

否则

火车1状态为1

火车3状态为3

否则

当火车3状态不为2且火车3状态不为3且火车3状态不为4

火车1状态不为2

当IsInPubluic((\*train1))=1且火车1状态为5时

当火车1距离<火车1轨道长+宽时

当IsNextInPubluic(\*train2)==1时

火车2状态为1

否则

当IsNextInPubluic(\*train3)==1时

火车3状态为1

否则当(IsInPubluic((\*train2))==1且火车2状态为5时

当IsNextInPubluic(\*train1)==1且火车1距离<火车1轨道长+宽时

火车1状态为1

否则当(IsInPubluic((\*train3))==1或火车3状态为5时

当IsNextInPubluic(\*train1)==1且火车1距离>火车1轨道长+宽时

火车1状态为1

**6.2函数2**

**函数名称：**int isBothInPub(TrainInfo a,TrainInfo b)

**函数参数说明：**a,b

**函数功能：**共轨处理

**函数局部变量设计：**无

**函数算法逻辑：**

当a车方向为1时

当b车数字为2时

当12顺时针 时

当a车距离>a车站点1且a车距离<a车站点2且(b车距离>b车站点2或b车距离<b车站点2时

返回1

否则

返回0;

否则

当a车距离>a车站点1且a车距离<a车站点2且(b车距离>站点2或b车距离<b站点1)时

返回1;

否则

返回0

否则

当b车方向为1 时

当a车距离>a车站台3且a车距离<a车站台4且(b车距离>b车站台1或b车距离<b车站台2)时

返回1;

否则

返回0

否则

当a车距离>a车站台3且a车距离<a车站台4且(b车距离>b车站台2或b车距离<b车站台1))

返回 1;

否则

返回 0;

否则

当b车数字为1时

当b车方向为1时

当a车距离>a车站台4且a车距离<a车站台3且(b车距离>b车站台1或b车距离<b车站台2)时

返回 1;

否则

返回 0;

否则

当a车距离>a车站台4且a车距离<a车站台3且(b车距离>b车站台2或b车距离<b车站台1)时

返回 1;

否则

返回 0;

否则

当b车方向为1时

当a车距离>a车站台2且a车距离<a车站台1且(b车距离>b车站台1或b车距离<b车站台2)时

返回 1;

否则

否则

当a车距离>a车站台2且a车距离<a车站台1且(b车距离>b车站台2或b车距离<b车站台1)时

返回 1;

否则

返回 0;

**6.3函数3**

**函数名称：**int IsNextInPubluic(TrainInfo a)

**函数参数说明：**a

**函数功能：**共轨处理

**函数局部变量设计：**无

**函数算法逻辑：**

当a车数字==1时

当a车方向==1时

当(a.train\_next<a车站台2且a.train\_next>a车站台1)或(a.train\_next<a车站台4且a.train\_next>a车站台3))

返回 1;

否则

返回 0;

否则

当(a.train\_next<a车站台1且a.train\_next>a车站台2)或(a.train\_next<a车站台3且a.train\_next>a车站台4))

返回 1;

否则

返回 0;

否则

当a车方向==1时

当a.train\_next<a车站台2或a.train\_next>a车站台1时

返回 1;

否则

返回 0;

否则

当a.train\_next<a车站台1或a.train\_next>a车站台2时

返回 1;

否则

返回 0;

**7.模块7<** Finite State Machine **>**

**7.1函数1**

**函数名称：**void ControlTrain(TrainInfo\* train)

**函数参数说明：**火车信息

**函数功能：**修改状态，运行中，等待进站，在站台等待，进站，出站，执行操作，更新2维坐标

**函数局部变量设计：**无

**函数算法逻辑：**

当火车距离小于0时

火车距离+=2\*火车轨道长+2\*火车轨道宽

(\*train).train\_next=火车距离+火车速度\*0.1

当火车状态==0时

当ReadytoJudge(\*train)==1时

当火车数字==1时

当火车运行距离>=火车轨道长+火车轨道宽时

当ReadytoJudge((\*((\*train).tptr2)))==1时

当运行策略为1时

当火车速度>(\*((\*train).tptr2)).train\_speed时

火车状态=3;//1车进站

(\*((\*train).tptr2)).train\_state=1

否则

火车状态=1;//1车等待进站

(\*((\*train).tptr2)).train\_state=3

否则

当(\*train).before==0)时

火车状态=3

(\*((\*train).tptr2)).train\_state=1

(\*train).before=1;

(\*((\*train).tptr2)).before=0

否则

火车状态=1

(\*((\*train).tptr2)).train\_state=3;

(\*train).before=0;

(\*((\*train).tptr2)).before=1

否则

火车状态=3

否则 当火车运行距离<火车轨道长+火车轨道宽时

当ReadytoJudge((\*((\*train).tptr1)))==1时

当run\_method==1时

当火车运行速度>(\*((\*train).tptr1)).train\_speed时

火车状态=3

(\*((\*train).tptr1)).train\_state=1

否则

火车状态=1

(\*((\*train).tptr1)).train\_state=3

否则

当(\*train).before==0)时 {

火车状态=3

(\*((\*train).tptr1)).train\_state=1

(\*train).before=1;

(\*((\*train).tptr1)).before=0;

否则

火车状态=1;

(\*((\*train).tptr1)).train\_state=3;

(\*train).before=0;

(\*((\*train).tptr1)).before=1

否则

火车状态=3

否则

当ReadytoJudge((\*((\*train).tptr1)))==1)时

当run\_method==1时

当(\*((\*train).tptr1))火车距离<=(\*((\*train).tptr1))火车轨道长+(\*((\*train).tptr1))火车轨道时

当火车数字==2时

当火车运行速度>(\*((\*train).tptr1)).train\_speed时

火车状态=3

(\*((\*train).tptr1)).train\_state=1

否则

火车状态=1;

(\*((\*train).tptr1)).train\_state=3;

否则

火车状态=3

否则

当火车数字==3时

当火车运行速度>(\*((\*train).tptr1)).train\_speed时

火车状态=3;//3车进站

(\*((\*train).tptr1)).train\_state=1

否则

火车状态=1;//3车等待进站

(\*((\*train).tptr1)).train\_state=3

否则

火车状态=3;//2车正常进站

否则

当(\*((\*train).tptr1))火车距离<=火车轨道长+火车轨道宽时

当火车数字==2时

当(\*train).before==0时

火车状态=3

(\*((\*train).tptr1)).train\_state=1

(\*train).before=1;

(\*((\*train).tptr1)).before=0

否则

火车状态=1;//2车等待进站

(\*((\*train).tptr1)).train\_state=3;

(\*train).before=0;

(\*((\*train).tptr1)).before=1

否则

火车状态=3

(\*((\*train).tptr1)).train\_state=3

否则

当火车数字==3时

当(\*train).before==0时

火车状态=3;//3车进站

(\*((\*train).tptr1)).train\_state=1;//1车等待进站

(\*train).before=1;

(\*((\*train).tptr1)).before=0

否则

火车状态=1;//3车等待进站

(\*((\*train).tptr1)).train\_state=3;

(\*train).before=0;

(\*((\*train).tptr1)).before=1

否则

火车状态=3;//2车正常进站

(\*((\*train).tptr1)).train\_state=3

否则

火车状态=3

否则 当火车状态==1时

当火车数字==1时

当火车运行距离>=火车轨道长+火车轨道宽时

当(\*((\*train).tptr2)).train\_state==0时

火车状态=3

否则 当火车运行距离<火车轨道长+火车轨道宽时

当(\*((\*train).tptr1)).train\_state==0)//2车已经进入非公共轨道

火车状态=3

否则

当(\*((\*train).tptr1)).train\_state==0)//1车进入公共轨道

火车状态=0

否则 当火车状态==2)

当(\*train).timeofkey<=0)

火车状态=4

否则 当火车状态==3时

当(\*train).train\_next+(\*train).train\_speed\*0.1>=2\*火车轨道长+2\*火车轨道宽时

(\*train).train\_next-=2\*火车轨道长+2\*火车轨道宽;

(\*train).train\_next+=火车运行速度\*0.1

当(\*train).train\_direction==1时

当火车数字==1时

当火车运行距离<=火车轨道长+火车轨道宽时

当(\*train).train\_next>=(\*train).station1)时

火车状态=2

否则

当(\*train).train\_next>=(\*train).station2)时

火车状态=2

否则

当(\*train).train\_next>=(\*train).station1)

火车状态=2

否则

当火车数字==1时

当火车运行距离<=火车轨道长+火车轨道宽时

当(\*train).train\_next<=(\*train).station1)

火车状态=2;

否则

当(\*train).train\_next<=(\*train).station2)

火车状态=2

否则

当(\*train).train\_next<=(\*train).station1时

火车状态=2

否则 当火车状态==4时

当火车数字==1时

当(\*train).train\_direction==1时

当火车运行距离>=火车轨道长+火车轨道宽)时

当火车运行距离+(\*train).train\_speed\*0.1>(\*train).train\_detect\_1D\_4

火车状态=0

否则

火车状态=4;

否则 当火车运行距离<火车轨道长+火车轨道宽时

当火车运行距离+(\*train).train\_speed\*0.1>(\*train).train\_detect\_1D\_2

火车状态=0

否则

火车状态=4;

否则 当(\*train).train\_direction==2)

当火车运行距离<火车轨道长+火车轨道宽时

当火车运行距离+(\*train).train\_speed\*0.1<(\*train).train\_detect\_1D\_4)

火车状态=0

否则 当火车运行距离>=火车轨道长+火车轨道宽)

当火车运行距离+(\*train).train\_speed\*0.1<(\*train).train\_detect\_1D\_2)

火车状态=0

否则 当火车数字==2时

当(\*train).train\_direction==1时

当(火车运行距离>(\*train).train\_detect\_1D\_2)&&(火车运行距离<火车轨道长))

火车状态=0

否则 当(\*train).train\_direction==2时

当火车运行距离<(\*train).train\_detect\_1D\_2&&火车运行距离>火车轨道长+火车轨道宽)

火车状态=0

否则

当(\*train).train\_direction==1)

当火车运行距离>(\*train).train\_detect\_1D\_2&&火车运行距离<火车轨道长+火车轨道宽)

火车状态=0

否则 当(\*train).train\_direction==2)

当火车运行距离<(\*train).train\_detect\_1D\_2&&火车运行距离>2\*火车轨道长+火车轨道宽)

火车状态=0

当火车状态==0时

否则 当火车状态==1时

(\*train).train\_speed=0;

当火车数字==1时

当(\*train).train\_direction==1时

当(\*train).train\_next<=火车轨道长+火车轨道宽时

火车运行距离=(\*train).train\_detect\_1D\_1;

否则

火车运行距离=(\*train).train\_detect\_1D\_3

否则

当(\*train).train\_next<=火车轨道长+火车轨道宽时

火车运行距离=(\*train).train\_detect\_1D\_3

否则

火车运行距离=(\*train).train\_detect\_1D\_1;

否则

火车运行距离=(\*train).train\_detect\_1D\_1

否则 当火车状态==2时

当火车数字==1时

当火车运行距离>=火车轨道长+火车轨道宽时

火车运行距离=(\*train).station2

否则

火车运行距离=(\*train).station1

否则

火车运行距离=(\*train).station1;

(\*train).train\_speed=0;

当patten==0时

(\*train).timeofkey=(\*train).train\_pause\_time\*10;

patten=1;

(\*train).timeofkey-=1;

否则 当火车状态==3时

当火车数字==1时

(\*train).train\_speed=speed\_temp1;

否则 当火车数字==2时

(\*train).train\_speed=speed\_temp2;

否则 当火车数字==3时

(\*train).train\_speed=speed\_temp3

否则 当火车状态==4

patten=0;

当火车数字==1时

(\*train).train\_speed=speed\_temp1;

否则 当火车数字==2时

(\*train).train\_speed=speed\_temp2;

否则 当火车数字==3时

(\*train).train\_speed=speed\_temp3

当火车数字==1时

(\*train).timeofkey=time\_temp1;

否则 当火车数字==2时

(\*train).timeofkey=time\_temp2;

否则 当火车数字==3时

(\*train).timeofkey=time\_temp3

火车运行距离+=(\*train).train\_speed\*0.1;

(\*train).train\_next=火车运行距离+(\*train).train\_speed\*0.1;//不断更新next

当火车运行距离<0)

火车运行距离+=2\*火车轨道长+2\*火车轨道宽;

当(\*train).train\_speed==0时

当火车数字==1时

(\*train).train\_next=火车运行距离+speed\_temp1\*0.1;

当火车数字==2时

(\*train).train\_next=火车运行距离+speed\_temp2\*0.1;

当火车数字==3时

(\*train).train\_next=火车运行距离+speed\_temp3\*0.1;

(\*train).train\_distance\_2D=Length\_To\_Coordinate((\*train));

(\*train).time++

int ReadytoJudge(TrainInfo a)

当a.train\_num==1时

当a.train\_direction==1时

当(a.train\_next>=a.train\_detect\_1D\_1&&a.train\_next<=a.train\_detect\_1D\_2)||(a.train\_next>=a.train\_detect\_1D\_3&&a.train\_next<=a.train\_detect\_1D\_4))

返回 1;

否则

返回 0;

否则 当a.train\_direction==2时

当(a.train\_next>=a.train\_detect\_1D\_4&&a.train\_next<=a.train\_detect\_1D\_3)||(a.train\_next>=a.train\_detect\_1D\_2&&a.train\_next<=a.train\_detect\_1D\_1))

返回 1;

否则

返回 0

否则 当a.train\_num==2||a.train\_num==3时

当a.train\_direction==1时

当a.train\_next>=a.train\_detect\_1D\_1||a.train\_next<=a.train\_detect\_1D\_2)

返回 1;

否则

返回 0

否则 当a.train\_direction==2时

当a.train\_next>=a.train\_detect\_1D\_2||a.train\_next<=a.train\_detect\_1D\_1)

返回 1;

否则

返回 0;

**7.2函数2**

**函数名称：**int ReadytoJudge(TrainInfo a)

**函数参数说明：**火车运行长度

**函数功能：**判断下一秒是否进入探测区。

**函数局部变量设计：**无

**函数算法逻辑：**

If 1车情况

if当车的方向为1时

进入探测区，返回1

否则，返回0

if当车的方向为2时

进入探测区，返回1

否则，返回0

If 2、3车情况

if当车的方向为1时

当a.train\_next>=a车站台1或a.train\_next<=a车站台2) 返回1

否则

返回0

if当车的方向为2时

当a.train\_next>=a车站台2或a.train\_next<=a车站台1) 返回1

否则

返回0

**8.模块8<**File Write **>**

**8.1函数1**

**函数名称：**void FileWrite(TrainInfo set[],FILE \*fp,int temp)

**函数参数说明：**默认数据集合、文件输入、set(temp)

**函数功能：**文件输入

**函数局部变量设计：**无

**函数算法逻辑：**

f输出火车车号、火车类型、轨道基准点坐标、启动位置

当set[temp]车数字==1)

f输出探测点1、2、3、4及一维坐标

否则

f输出探测点1、2及一维坐标

f输出启动时间、火车速度、停靠时间、所在轨道:长 宽

**8.2函数2**

**函数名称：**void FileWriteOut(TrainInfo\* train,FILE \*fp,int OutputTime)

**函数参数说明：**火车信息、文件输入、输出时间

**函数功能：**文件输入

**函数局部变量设计：**无

**函数算法逻辑**

当run\_method==1时

f输出当前策略：快车先行

否则

f输出当前策略：交替策略

f输出当前时间

f输出 [A车]

f输出运行方向

当train[0].train\_direction==1时

f输出顺时针

否则

f输出逆时针

f输出运行状态;

当train[0].train\_state==0时

f输出运行-非公共轨道

否则 当train[0].train\_state==1时

f输出暂停-等待公共轨道

否则 当train[0].train\_state==2时

f输出停靠-占用公共轨道(还需等待%ds);

否则 当train[0].train\_state==3时

f输出运行-公共轨道

否则 当train[0].train\_state==4时

f输出运行-公共轨道

否则 当train[0].train\_state==5时

f输出接受命令暂停等状态

f输出当前位置

f输出[B车]

f输出运行方向

当train[1].train\_direction==1时

f输出顺时针

否则

f输出逆时针

f输出运行状态

当train[1].train\_state==0时

f输出运行-非公共轨道

否则 当train[1].train\_state==1时

f输出暂停-等待公共轨道

否则 当train[1].train\_state==2时

f输出停靠-占用公共轨道(还需等待s)

否则 当train[1].train\_state==3时

f输出运行-公共轨道

否则 当train[1].train\_state==4时

f输出运行-公共轨道

否则 当train[1].train\_state==5时

f输出接受命令暂停等状态

f输出当前位置

f输出[C车]

f输出运行方向

当train[2].train\_direction==1时

f输出顺时针

否则

f输出逆时针

f输出运行状态

当train[2].train\_state==0时

f输出运行-非公共轨道

否则 当train[2].train\_state==1时

f输出暂停-等待公共轨道

否则 当train[2].train\_state==2时

f输出停靠-占用公共轨道(还需等待%ds)

否则 当train[2].train\_state==3时

f输出运行-公共轨道

否则 当train[2].train\_state==4时

f输出运行-公共轨道

否则 当train[2].train\_state==5)

f输出接受命令暂停等状态

f输出当前位置

f输出[公共轨道状态]

当IsInPubluic(train[0])==0且IsInPubluic(train[1])==0且IsInPubluic(train[2])==0)

f输出空闲

否则

当IsInPubluic(train[0])==1时

f输出被A车占用

当IsInPubluic(train[1])==1时

f输出被B车占用

当IsInPubluic(train[2])==1时

f输出被C车占用

f输出

**9.模块9<** outputCli>

**9.1函数1**

**函数名称：** PrintTrainInfo(TrainInfo\* train,int OutputTime)

**函数参数说明：**火车信息，输出时间

**函数功能：**命令行输出：当前时间，当前策略，3辆火车运行状态，3辆火车当前位置，公共轨道的状态

**函数局部变量设计：**无

**函数算法逻辑：**

输出当前时间

输出 [A车]

输出运行方向

if当车的方向为1时 输出顺时针

否则输出逆时针

输出运行状态

if当车的运行状态为0时 输出运行-非公共轨道

if当车的运行状态为1时 输出暂停-等待公共轨道

if当车的运行状态为2时 输出停靠-占用公共轨道(还需等待%ds)

if当车的运行状态为3时 输出f运行-公共轨道

if当车的运行状态为4时 输出运行-公共轨道

if当车的运行状态为5时 输出接受命令暂停等状态

输出当前位置

同理输出B,C车信息

输出占用轨道状态

FILE \*outputfile;

If当outputfile=fopen("输出数据.ini","a+")为空时

exit(0);

否则 否则调用函数FileWriteOut(train,outputfile,OutputTime);

fflush(outputfile);

**9.2函数2**

**函数名称：**IsInPubluic(TrainInfo a)

**函数参数说明：**车辆信息

**函数功能：**判断是否占用公共轨道

**函数局部变量设计：**无

**函数算法逻辑：**

顺时针方向

占用返回1

否则返回0

逆时针方向

占用返回1

否则返回0

**10.模块10<** Thread>

**10.1函数1**

**函数名称：** TRAIN1RUN(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**火车1线程

**函数局部变量设计：** TrainInfo \*train =(TrainInfo \*)lpParam;

int a=1;

**函数算法逻辑：**

当火车起始时间不为0时

state\_temp1=火车状态

火车状态为5

Sleep(1000\*(\*train).train\_start\_time);

火车状态=state\_temp1;

while(1)进行循环

调用函数WaitForSingleObject(TrianRunMutex1,INFINITE);

TAB=火车运行距离;

if当KeyofTrain1=1时

当火车状态不为5时

state\_temp1=火车状态

火车状态为5

赋值a=0

调用函数ReleaseMutex(TrianRunMutex1)

Continue

当KeyofTrain1==2且a==0时

火车状态= state\_temp1

a=1

当KeyofTrain1==2且a==1时

state\_temp1=火车状态

定义clock\_t start,finish;

定义long long duration;

赋值start=clock();

调用函数ControlTrain(train);

赋值finish=clock();

赋值duration=finish-start;

调用函数ReleaseMutex(TrianRunMutex1)

Sleep(1000-duration)；

返回0;

**10.2函数2**

**函数名称：** TRAIN2RUN(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**火车2线程

**函数局部变量设计：**int a=1

**函数算法逻辑：**

当火车起始时间不为0时

state\_temp2=火车状态

火车状态为5

Sleep(1000\*(\*train).train\_start\_time);

火车状态=state\_temp2;

while(1)进行循环

调用函数WaitForSingleObject(TrianRunMutex1,INFINITE);

TAB=火车运行距离;

if当KeyofTrain2=1时

当火车状态不为5时

state\_temp2=火车状态

火车状态为5

赋值a=0

调用函数ReleaseMutex(TrianRunMutex1)

Continue

当KeyofTrain2==2且a==0时

火车状态= state\_temp2

a=1

当KeyofTrain2==2且a==1时

state\_temp2=火车状态

定义clock\_t start,finish;

定义long long duration;

赋值start=clock();

调用函数ControlTrain(train);

赋值finish=clock();

赋值duration=finish-start;

调用函数ReleaseMutex(TrianRunMutex1)

Sleep(1000-duration)；

返回0;

**10.3函数3**

**函数名称：**TRAIN3RUN(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**火车3线程

**函数局部变量设计：**int a=1;

**函数算法逻辑：**

当火车起始时间不为0时

state\_temp3=火车状态

火车状态为5

Sleep(1000\*(\*train).train\_start\_time);

火车状态=state\_temp3;

while(1)进行循环

调用函数WaitForSingleObject(TrianRunMutex1,INFINITE);

TAB=火车运行距离;

if当KeyofTrain3=1时

当火车状态不为5时

state\_temp3=火车状态

火车状态为5

赋值a=0

调用函数ReleaseMutex(TrianRunMutex1)

Continue

当KeyofTrain3==2且a==0时

火车状态= state\_temp3

a=1

当KeyofTrain2==3且a==1时

state\_temp3=火车状态

定义clock\_t start,finish;

定义long long duration;

赋值start=clock();

调用函数ControlTrain(train);

赋值finish=clock();

赋值duration=finish-start;

调用函数ReleaseMutex(TrianRunMutex1)

Sleep(1000-duration)；

返回0;

**10.4函数4**

**函数名称：** OutPutFILENCMD(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**每秒输出，包括信息见要求的doc， 文件和命令行

**函数局部变量设计：** int OutputTime=1;

TrainInfo \*train =(TrainInfo \*)lpParam;

clock\_t start,finish;

long long duration;

**函数算法逻辑：**

循环while(1)

调用函数WaitForSingleObject(TrianRunMutex1,INFINITE);

赋值start=clock();

调用函数PrintTrainInfo(train,OutputTime,caidan);

赋值finish=clock();

duration=finish-start;

OutputTime++;

调用函数ReleaseMutex(TrianRunMutex1);

调用函数Sleep(1000-duration);

**10.5函数5**

**函数名称：** ModDit(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**

**函数局部变量设计：** TrainInfo \*train =(TrainInfo \*)lpParam;

clock\_t start,finish;

long long duration;

**函数算法逻辑：**

循环while(1)

调用函数WaitForSingleObject(TrianRunMutex1,INFINITE);

赋值start=clock();

火车距离=(火车距离)%(2\*(火车轨道长+宽);

火车+1距离=(火车+1距离)%(2\*火车+1轨道长+宽);

火车+2距离=(火车+2距离)%(2\*火车+2轨道长+宽)

赋值finish=clock();

赋值duration=finish-start;

调用函数ReleaseMutex(TrianRunMutex1);

**10.6函数6**

**函数名称：**CreatCartoon(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**

**函数局部变量设计：** TrainInfo \*train =(TrainInfo \*)lpParam;

clock\_t start,finish;

long long duration;

char timep[10];

**函数算法逻辑：**

循环while(1)

赋值start=clock();

赋值btkey=1;

调用函数itoa((\*train).time/10,timep,10);

调用函数SetActiveEgg(rail);

调用函数SetPenColor(WHITE);

当run\_method==1时

调用函数调用函数MovePen(570,180)

调用函数调用函数DrawString("快车优先");

否则

调用函数调用函数MovePen(570,180);

调用函数调用函数DrawString("交替策略");

调用函数调用函数MovePen(570,120);

调用函数调用函数DrawString(timep);

调用函数SetActiveEgg(TA);

当火车运行方向为1时

调用函数MoveEgg(火车二维x坐标\*1.5,900-火车二维y坐标\*1.5);

调用函数SetActiveEgg(backdrop)

否则

调用函数SetActiveEgg(TB);

当火车+1运行方向为1时

调用函数MoveEgg(火车+1二维x坐标\*1.5,900-火车+1二维y坐标\*1.5);

调用函数SetActiveEgg(backdrop);

否则

调用函数SetActiveEgg(TC);

当火车+2运行方向为1时

调用函数MoveEgg(火车+2二维x坐标\*1.5-75,1050-火车+2二维y坐标 \*1.5+火车+2轨道宽 \*1.5);

调用函数SetActiveEgg(backdrop);

否则

当IsInPubluic(\*train)==0且IsInPubluic(\*(train+1))==0且IsInPubluic(\*(train+2))==0时

调用函数调用函数MovePen(180,180);

调用函数调用函数DrawString("无人占用");

调用函数调用函数MovePen(180,100);

调用函数调用函数DrawString("无人占用");

否则

当IsInPubluic(\*train)==1时

当火车运行距离<=火车轨道长+宽时

调用函数MovePen(180,180);

调用函数DrawString("A车占用");

当IsInPubluic(\*(train+1))==0时

调用函数MovePen(180,180);

调用函数DrawString("无人占用");

否则

调用函数MovePen(180,100);

调用函数DrawString("A车占用");

当IsInPubluic(\*(train+2))==0时{

调用函数MovePen(180,100);

调用函数DrawString("无人占用")

当IsInPubluic(\*(train+1))==1)时

调用函数MovePen(180,180);

调用函数DrawString("B车占用");

当IsInPubluic(\*(train+2))==0时

调用函数MovePen(180,100);

调用函数DrawString("无人占用")

当IsInPubluic(\*(train+2))==1)

当IsInPubluic(\*(train+1))==0时

调用函数MovePen(180,180);

调用函数DrawString("无人占用");

调用函数MovePen(180,100);

调用函数DrawString("C车占用");

调用函数MovePen(850,180);

定义char pauseA[10];

itoa((\*train).timeofkey,pauseA,10);

调用函数DrawString(pauseA);

调用函数MovePen(850,140);

定义char pauseB[10];

itoa((\*(train+1)).timeofkey,pauseB,10);

调用函数DrawString(pauseB);

调用函数MovePen(850,100);

定义char pauseC[10];

itoa((\*(train+2)).timeofkey,pauseC,10);

调用函数DrawString(pauseC);

赋值finish=clock();

赋值duration=finish-start;

Update();

Sleep(100-duration);

**10.7函数7**

**函数名称：**TRAINCONTROL(LPVOID lpParam)

**函数参数说明：**地址参数

**函数功能：**用户挂起--挂起对应线程即可，无操作

**函数局部变量设计：** int t1;

int t2;

int t3;

double Xk,Yk;

EVENT\_TYPE event;

TrainInfo \*train =(TrainInfo \*)lpParam;

**函数算法逻辑：**

while 循环(event=WaitForEvent()!=EXIT)

当IsKeyDown(VK\_LBUTTON)时

赋值Xk=GetMouseX(),

赋值Yk=GetMouseY();

当(Xk>=1200)且(Xk<=1300)且(Yk>=670)且(Yk<=750)时

KeyofTrain1=1;

输出("A车被用户暂停\n");

否则 当(Xk>=1320)且(Xk<=1420)且(Yk>=670)且(Yk<=750)时

赋值KeyofTrain1=2;

输出("A车重新启动\n");

否则 当(Xk>=1200)且(Xk<=1300)且(Yk>=580)且(Yk<=660)时

赋值KeyofTrain2=1;

输出("B车被用户暂停\n");

否则 当(Xk>=1320)且(Xk<=1420)且(Yk>=580)且(Yk<=660))

赋值KeyofTrain2=2;

输出("B车重新启动\n");

否则 当(Xk>=1200)且(Xk<=1300)且(Yk>=490)且(Yk<=570))

赋值KeyofTrain3=1;

输出("C车被用户暂停\n");

否则 当(Xk>=1320)且(Xk<=1420)且(Yk>=490)且(Yk<=570))

赋值KeyofTrain3=2;

输出("C车重新启动\n");

否则 当(Xk>=1200)且(Xk<=1300)且(Yk>=400)且(Yk<=480))

赋值run\_method=1;

输出("控制策略切换为快车优先策略。\n");

否则 当(Xk>=1320)且(Xk<=1420)且(Yk>=400)且(Yk<=480)

赋值run\_method=2;

输出("控制策略切换为交替策略。\n")

输出("用户已退出!");

\_exit(0);

**10.8函数8**

**函数名称：**AllControl(TrainInfo train[4])

**函数参数说明：**火车信息

**函数功能：**全部控制

**函数局部变量设计：**无

**函数算法逻辑：**

头文件HANDLE Thread1,Thread2,Thread3,Thread4,Thread5,Thread6,Thread7;

TrianRunMutex1=CreateMutex(NULL,FALSE,"KeyofTrain1");

调用函数SetAutoUpdate(0);

赋值Thread5=CreateThread(NULL,0,CreatCartoon,&train[1],0,0);

Sleep(3);

赋值Thread6=CreateThread(NULL,0,ModDit,&train[1],0,0);

Sleep(3);

赋值Thread1=CreateThread(NULL,0,TRAIN1RUN,&train[1],0,0);

Sleep(3);

赋值Thread2=CreateThread(NULL,0,TRAIN2RUN,&train[2],0,0);

Sleep(3);

赋值Thread3=CreateThread(NULL,0,TRAIN3RUN,&train[3],0,0);

Sleep(3);

赋值Thread4=CreateThread(NULL,0,TRAINCONTROL,&train[1],0,0);

Sleep(5);

赋值Thread7=CreateThread(NULL,0,OutPutFILENCMD,&train[1],0,0);

调用函数CloseHandle(Thread1);

CloseHandle(Thread2);

CloseHandle(Thread3);

CloseHandle(Thread4);

CloseHandle(Thread5);

Sleep(10000000);

**11.模块11<** CreatMap>

**11.1函数1**

**函数名称：**void Creatmap(TrainInfo train[4])

**函数参数说明：** 火车信息

**函数功能：**绘制A,B,C车轨道及其公轨，绘制运行策略，绘制当前时间，暂停时间，模拟火车实际运行情况。

**函数局部变量设计：无**

**函数算法逻辑：**

调用函数EggStart(1440,900);

赋值backdrop=LayEgg();

调用函数DrawBitmap("backdrop.bmp");

赋值rail=LayEgg();

调用函数MovePen(50,900-50);

调用函数SetPenColor(RED);

调用函数SetPenWidth(6);

调用函数DrawLine(0,-火车1轨道宽\*1.5);

调用函数DrawLine(火车1轨道长\*1.5,0);

调用函数DrawLine(0,火车1轨道宽\*1.5);

调用函数DrawLine(-火车1轨道长\*1.5,0);

调用函数OffsetPen(火车1轨道长\*7.5-25,-火车1轨道宽\*7.5);

调用函数DrawString("A车轨道");

调用函数MovePen(50+火车1轨道长\*1.5,900-50-train[2].train\_datum\_point.y\*1.5);

调用函数SetPenColor(YELLOW);

调用函数DrawLine(0,-火车2轨道宽\*1.5);

调用函数SetPenColor(GREEN);

调用函数DrawLine(火车2轨道长\*1.5,0);

调用函数DrawLine(0,火车2轨道宽\*1.5);

调用函数DrawLine(-火车2轨道长\*1.5,0);

调用函数OffsetPen(火车2轨道长\*7.5-25,-火车2轨道宽\*7.5);

调用函数DrawString("B车轨道");

调用函数MovePen(50+train[3].train\_datum\_point.x\*1.5,900-50-火车1轨道宽\*1.5);

调用函数SetPenColor(CYAN);

调用函数DrawLine(0,-火车3轨道宽\*1.5);

调用函数DrawLine(火车3轨道长\*1.5,0);

调用函数DrawLine(0,火车3轨道宽\*1.5);

调用函数SetPenColor(YELLOW);

调用函数DrawLine(-火车3轨道长\*1.5,0);

调用函数OffsetPen(火车3轨道长\*7.5-25,-火车3轨道宽\*7.5);

SetPenColor(CYAN);

调用函数DrawString("C车轨道");

调用函数MovePen(1200,750);

DrawBitmap("Button1.bmp");

OffsetPen(20,0);

DrawBitmap("Button2.bmp");

OffsetPen(-100,-90);

DrawBitmap("Button4.bmp");

OffsetPen(-220,0);

DrawBitmap("Button3.bmp");

OffsetPen(-100,-90);

DrawBitmap("Button5.bmp");

OffsetPen(20,0);

DrawBitmap("Button6.bmp");

OffsetPen(-100,-90);

DrawBitmap("Button8.bmp");

OffsetPen(-220,0);

DrawBitmap("Button7.bmp");

SetPenColor(WHITE);

SetPenWidth(10);

调用函数MovePen(100,180);

调用函数DrawString("A-B公轨");

double x;

x=GetPenXPos()-100;

OffsetPen(-x,-80);

调用函数DrawString("A-C公轨");

调用函数MovePen(500,180);

调用函数DrawString("运行策略：");

调用函数MovePen(500,120);

调用函数DrawString("当前时间：");

调用函数MovePen(700,180);

调用函数DrawString("A暂停时间(0.1s)：");

调用函数MovePen(700,140);

调用函数DrawString("B暂停时间(0.1s)：");

调用函数MovePen(700,100);

调用函数DrawString("C暂停时间(0.1s)：");

lightR1=LayEgg();

TA=LayEgg();

调用函数MovePen(25,900-25);

当train[1].train\_start\_1D<=火车1轨道长)

OffsetPen(train[1].train\_start\_1D\*1.5,0);

当train[1].train\_start\_1D<=火车1轨道长+火车1轨道宽且train[1].train\_start\_1D>火车1轨道长)

OffsetPen(火车1轨道长\*1.5,-(train[1].train\_start\_1D-火车1轨道长)\*1.5);

当train[1].train\_start\_1D>火车1轨道长+火车1轨道宽且train[1].train\_start\_1D<=2\*火车1轨道长+火车1轨道宽)

OffsetPen(火车1轨道长\*1.5-(train[1].train\_start\_1D-火车1轨道长-火车1轨道宽)\*1.5,-火车1轨道宽\*1.5);

当train[1].train\_start\_1D<=(火车1轨道长+火车1轨道宽)\*2且train[1].train\_start\_1D>2\*火车1轨道长+火车1轨道宽)

OffsetPen(0,-火车1轨道宽\*1.5+(train[1].train\_start\_1D-火车1轨道长\*2-火车1轨道宽)\*1.5);

DrawBitmap("TA.bmp");

TB=LayEgg();

调用函数MovePen(25+火车1轨道长\*1.5,900-25-train[2].train\_datum\_point.y\*1.5);

当train[2].train\_start\_1D<=火车2轨道长)

OffsetPen(train[2].train\_start\_1D\*1.5,0);

当train[2].train\_start\_1D<=火车2轨道长+火车2轨道宽且train[2].train\_start\_1D>火车2轨道长)

OffsetPen(火车2轨道长\*1.5,-(train[2].train\_start\_1D-火车2轨道长)\*1.5);

当train[2].train\_start\_1D>火车2轨道长+火车2轨道宽且train[2].train\_start\_1D<=2\*火车2轨道长+火车2轨道宽)

OffsetPen(火车2轨道长\*1.5-(train[2].train\_start\_1D-火车2轨道长-火车2轨道宽)\*1.5,-火车2轨道宽\*1.5);

当train[2].train\_start\_1D<=(火车2轨道长+火车2轨道宽)\*2且train[2].train\_start\_1D>2\*火车2轨道长+火车2轨道宽)

OffsetPen(0,-火车2轨道宽\*1.5+(train[2].train\_start\_1D-火车2轨道长\*2-火车2轨道宽)\*1.5);

DrawBitmap("TB.bmp");

TC=LayEgg();

调用函数MovePen(25+train[3].train\_datum\_point.x\*1.5,900-25-火车1轨道宽\*1.5);

当train[3].train\_start\_1D<=火车3轨道长)

OffsetPen(train[3].train\_start\_1D\*1.5,0);

当train[3].train\_start\_1D<=火车3轨道长+火车3轨道宽且train[3].train\_start\_1D>火车3轨道长)

OffsetPen(火车3轨道长\*1.5,-(train[3].train\_start\_1D-火车3轨道长)\*1.5);

当train[3].train\_start\_1D>火车3轨道长+火车3轨道宽且train[3].train\_start\_1D<=2\*火车3轨道长+火车3轨道宽)

OffsetPen(火车3轨道长\*1.5-(train[3].train\_start\_1D-火车3轨道长-火车3轨道宽)\*1.5,-火车3轨道宽\*1.5);

当train[3].train\_start\_1D<=(火车3轨道长+火车3轨道宽)\*2且train[3].train\_start\_1D>2\*火车3轨道长+火车3轨道宽)

OffsetPen(0,-火车3轨道宽\*1.5+(train[3].train\_start\_1D-火车3轨道长\*2-火车3轨道宽)\*1.5);

DrawBitmap("TC.bmp");