在设计函数过程中要用到的结构体：

struct TRAIN{

double speed;//初始速度

int starttime;//火车开始运行的时间

int startpoint;//火车开始运行的位置（为非公共轨道位置）

int direction;//火车的起始运动方向（1表示顺时针，2表示逆时针）

char run[10];//火车运行方向

int point;//火车当前所处的位置

int sum//火车运行轨道的点的个数

};

typedef struct TRAIN Train;

Train train[3];

Train \* tra=train;

struct BLOCK{

int commom=0;//0非公共，1、2代表不同的公共轨道

int direction=0;// 1左2下3右4上

};

接下来是一些关键函数算法实现的代码或伪代码：

函数：int pausetime()；

参数：int time；//计算火车暂停或停靠的时间

接收到暂停或停靠信号;

while (火车暂停或停靠) {

if (时间片推进一个)

time++;

}

return time;

函数：void kinit\_railway()//轨道初始化

{

int lenA,lenB,lenC;

struct block \* railwayA,railwayB,railwayC;

printf("输入轨道A的长度:\n");

scanf("%d",&lenA);

railwayA =(struct block \*) malloc(lenA,sizeof(struct block));

printf("输入轨道B的长度:\n");

scanf("%d",&lenB);

railwayB =(struct block \*) malloc(lenB,sizeof(struct block));

printf("输入轨道C的长度:\n");

scanf("%d",&lenC);

railwayC =(struct block \*) malloc(lenC,sizeof(struct block));

//创建三条轨道的结构数组。

int cor\_pointA[4], cor\_pointB[4],cor\_pointC[4];

int com\_pointA[6],com\_pointB[6],com\_pointC[6];

A printf("从西南角开始，顺时针输入轨道A的四个拐点:\n");

gets(cor\_pointA);

fflush(stdin);

int i;

for(int j=0;j<4;j++)

{

for(i=cor\_pointA[j];i!=cor\_pointA[j+1];i++)

{

if(i>=lenA)

{

i=-1;

continue;

}

railwayA[i].direction=j+1;

}

}

for(j=0;j<4;j++)

railwayA.direction[cor\_pointA[j]]=5;

//方向

printf("输入轨道A的公共轨道的起始点到结束点:\n");

gets(com\_pointA);

for(int j=0;j<6;j+2)

{

for(i=com\_pointA[j];i!=1+com\_pointA[j+1];i++)

{

if(i>=lenA)

{

i=-1;

continue;

}

railwayA[i].common=1;

}

}

//公共轨道

printf("从西北角开始，顺时针输入轨道B的四个拐点:\n");

gets(cor\_pointB);

fflush(stdin);

for(int j=0;j<4;j++)

{

for(i=cor\_pointB[j];i!=cor\_pointB[j+1];i++)

{

if(i>=lenB)

{

i=-1;

continue;

}

railwayB[i].direction=j+1;

}

}

for(j=0;j<4;j++)

railwayB.direction[cor\_pointB[j]]=5;

//方向

printf("输入轨道B的公共轨道的起始点到结束点:\n");

gets(com\_pointB);

for(int j=0;j<6;j+2)

{

for(i=com\_pointB[j];i!=1+com\_pointB[j+1];i++)

{

if(i>=lenB)

{

i=-1;

continue;

}

railwayB[i].common=1;

}

}

//公共轨道

printf("从西北角开始，顺时针输入轨道C的四个拐点:\n");

gets(cor\_pointC);

fflush(stdin);

for(int j=0;j<4;j++)

{

for(i=cor\_pointC[j];i!=cor\_pointC[j+1];i++)

{

if(i>=lenC)

{

i=-1;

continue;

}

rCilwayC[i].direction=j+1;

}

}

for(j=0;j<4;j++)

railwayC.direction[cor\_pointC[j]]=5;

//方向

printf("输入轨道C的公共轨道的起始点到结束点:\n");

gets(com\_pointC);

for(int j=0;j<6;j+2)

{

for(i=com\_pointC[j];i!=1+com\_pointC[j+1];i++)

{

if(i>=lenC)

{

i=-1;

continue;

}

railwayC[i].common=1;

}

}

//公共轨道

}

函数：void trans()；//火车状态的转换

{

switch 火车状态

{

case 停止:

if 顺时针启动

火车状态=顺时针行驶；

if 逆时针启动

火车状态=逆时针行驶；

break;

case 顺时针行驶:

if 公共轨道有车&&火车在公共轨道入口

火车状态=公共口暂停；

if 用户暂停

火车状态=任意点暂停；

if 结束信号

exit(1)；

break;

case 逆时针行驶:

if 公共轨道有车&&火车在公共轨道入口

火车状态=公共口暂停；

if 用户暂停

火车状态=任意点暂停；

if 结束信号

exit(1)；

break;

case 公共口暂停:

if 公共口有车

break；

if 公共轨道无车&&火车顺时针运行

火车状态=顺时针行驶；

if 公共轨道无车&&火车逆时针运行

火车状态=逆时针运行；

break；

case 任意点暂停:

if 在公共轨道口

火车状态=公共口暂停；

if 用户恢复&&火车顺时针运行

火车状态=顺时针行驶；

if 用户恢复&&火车逆时针运行

火车状态=逆时针行驶；

break；

}

}

函数：void direction(Train \* tra,Block \* rail ){//判断火车的运行方向

if(tra.direction==1){

if(rail[tra.point]==1)

strcpy(tra.run,"north");

else if(rail[train.point]==2)

strcpy(tra.run,"west");

else if(rail[tra.point]==3)

strcpy(tra.run,"south");

else if(rail[tra.point]==4)

strcpy(tra.run,"east");

else

continue;

}

else{

if(rail[tra.point]==1)

strcpy(tra.run,"south");

else if(rail[tra.point]==2)

strcpy(tra.run,"east");

else if(rail[tra.point]==3)

strcpy(tra.run,"north");

else if(rail[tra.point]==4)

strcpy(tra.run,"west");

else

continue;

}

}函数：void time\_count()；

int i=0;

while(i<1000){

i++;

Sleep(1000);

system("cls");

printf("T=%ds",i);

}

函数：int trainpoint(Train \* tra)//计算火车当前所处的位置

{

double position=tra.point+.2\*tra.speed;

if(position>=tra.sum)

position=position-tra.sum;

return position;

}