# 4.1　多边形的扫描转换

## 多边形的定义

多边形是由折线段组成的封闭图形，它由有序顶点的点集来表示。

## 多边形的表示

1. 顶点表示法
2. 点阵表示法
3. 多边形的扫描转换：由顶点表示法变换到点阵表示法的过程

## 多边形填充

主要算法是扫描线算法，自底向上的水平扫描线对多边形区域进行扫描，每条扫描线的填充包括求交、排序、配对和填充四个步骤的处理。

## 区域填充

还可通过种子填充的方式对区域进行填充。

# 4.2 有效边表填充算法

## 填充原理

扫描线算法, 与一般扫描线算法的区别是，用类似画线算法的方法计算与边界的交点，不需要每次执行求交算法。

扫描线与一条边的交点计算：yi+1= yi +1; xi+1=xi+1/k;

## 边界像素处理原则

“左闭右开”和“下闭上开”

## 结构体

### 边记录：

CAET

{//记录边的结构体

double x;//边的x坐标

int yMax;//顶点y坐标最大值

double k; //边斜率的倒数

CAET \* next; //下一条边记录指针

｝

### 桶记录

class CBucket

{//记录扫描线的结构体

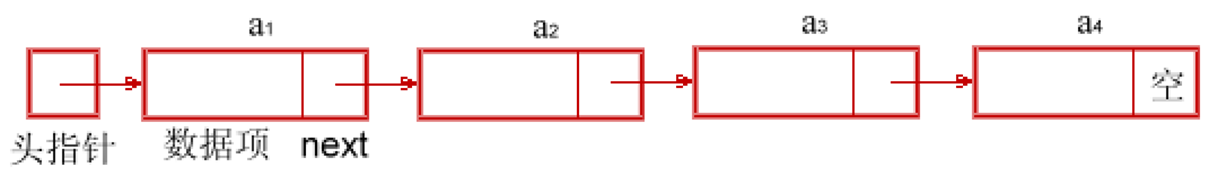
int ScanLine; //扫描线y坐标

CAET \*p; //桶上的边表头指针

CBucket \*next; //下一条桶记录指针

};

## 链表



### （1）桶表

存储扫描线记录的链表。

### （2）边表

存储边记录的链表，从同一扫描线开始的边构成一个链表，其头指针链接到桶记录上。

### （3）有效边表

存储与当前扫描线相交的边记录的链表。

## 链表建立的实例

### 桶表建立算法

* for(i=ScanMin;i<=ScanMax;i++)//建立桶结点
* {
* if(ScanMin==i)//桶头结点
* {
* HeadB=new CBucket;//建立桶的头结点
* CurrentB=HeadB;//CurrentB为CBucket当前结点指针
* CurrentB->ScanLine=ScanMin;
* CurrentB->p=NULL;//没有连接边链表
* CurrentB->next=NULL;
* }
* else//建立桶的其它结点
* {
* CurrentB->next=new CBucket;//新建一个桶结点
* CurrentB=CurrentB->next;//使CurrentB指向新建的桶结点
* CurrentB->ScanLine=i;
* CurrentB->p=NULL;//没有连接边链表
* CurrentB->next=NULL;
* }
* }

*ScanMin=1; ScanMax=12;*

HeadB

y

∙

∙

∙

∙

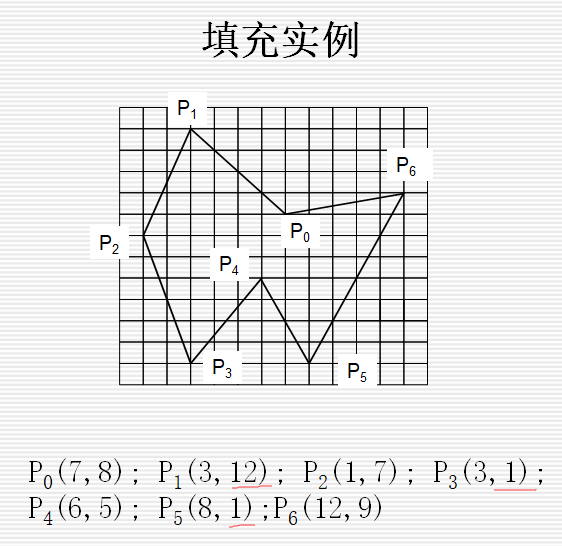
CurrentB

### 边表建立

* 对顶点表进行循环，Pi为当前点，Pi+1为下一点，比较两点的y坐标，y小的为Plow,大的为Phigh; 令经yMin=plow.y; x=Plow.x; yMax=Phigh.y;



* 将该结点链接到scanLine=yMin的桶结点的链表中
* （3）桶上的边表按x坐标排序



|  |
| --- |
| 12 |
| 11 |
| 10 |
| 9 |
| 8 |
| 7 |
| 6 |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | - | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ^ |

^

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ^ |

^

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

^

^

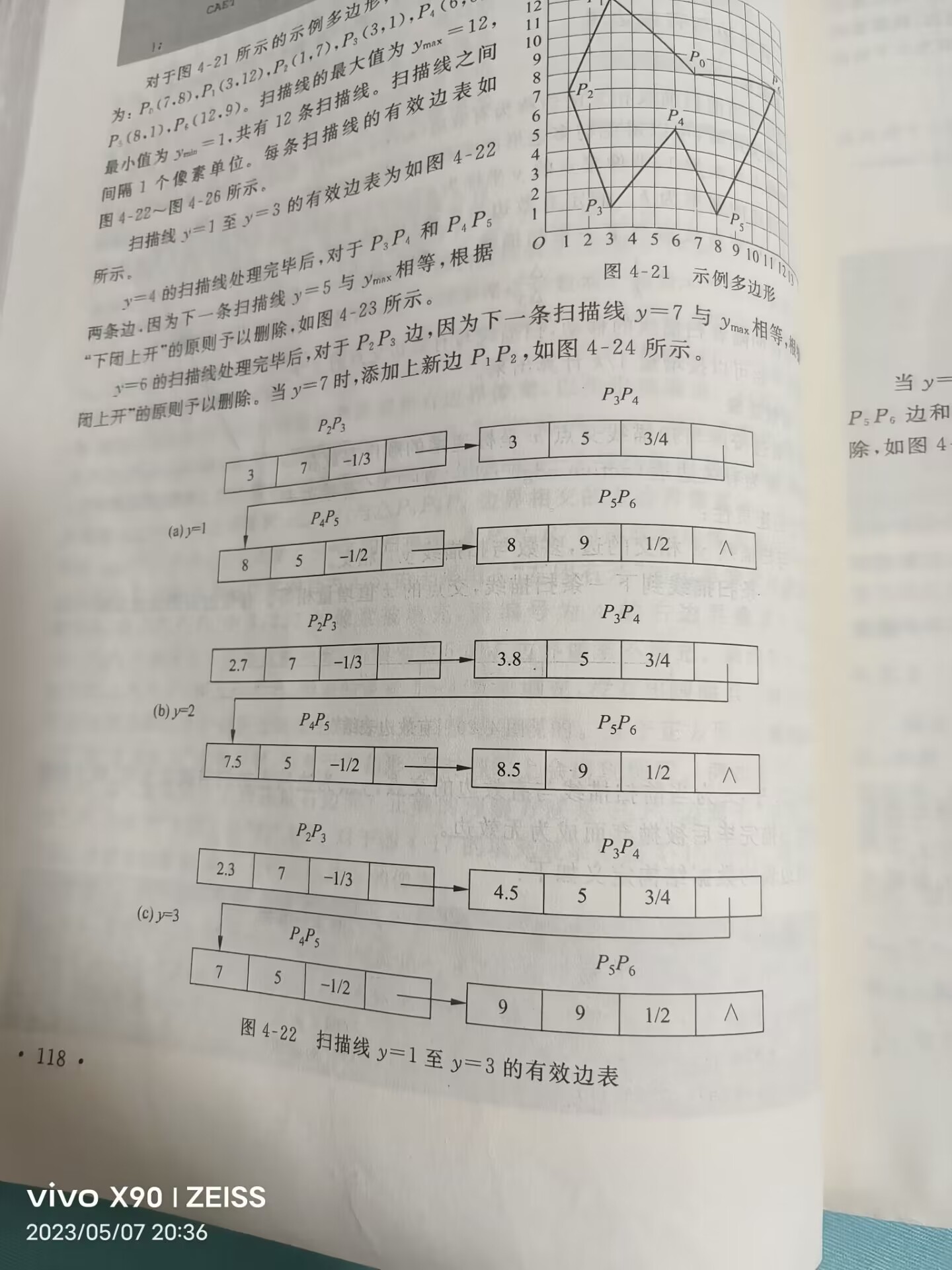
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | Pi | Pi+1 | Plow | Phigh | yMin | x | yMax | 1/k |
| 0 | P0(7,8) | P1(3,12) |  |  |  |  |  |  |
| 1 | P1(3,12) | P2(1,7) |  |  |  |  |  |  |
| 2 | P2(1,7) | P3(3,1) |  |  |  |  |  |  |
| 3 | P3(3,1) | P4(6,5) |  |  |  |  |  |  |
| 4 | P4(6,5) | P5(8,1) |  |  |  |  |  |  |
| 5 | P5(8,1) | P6(12,9) |  |  |  |  |  |  |
| 6 | P6(12,9) | P0(7,8) |  |  |  |  |  |  |

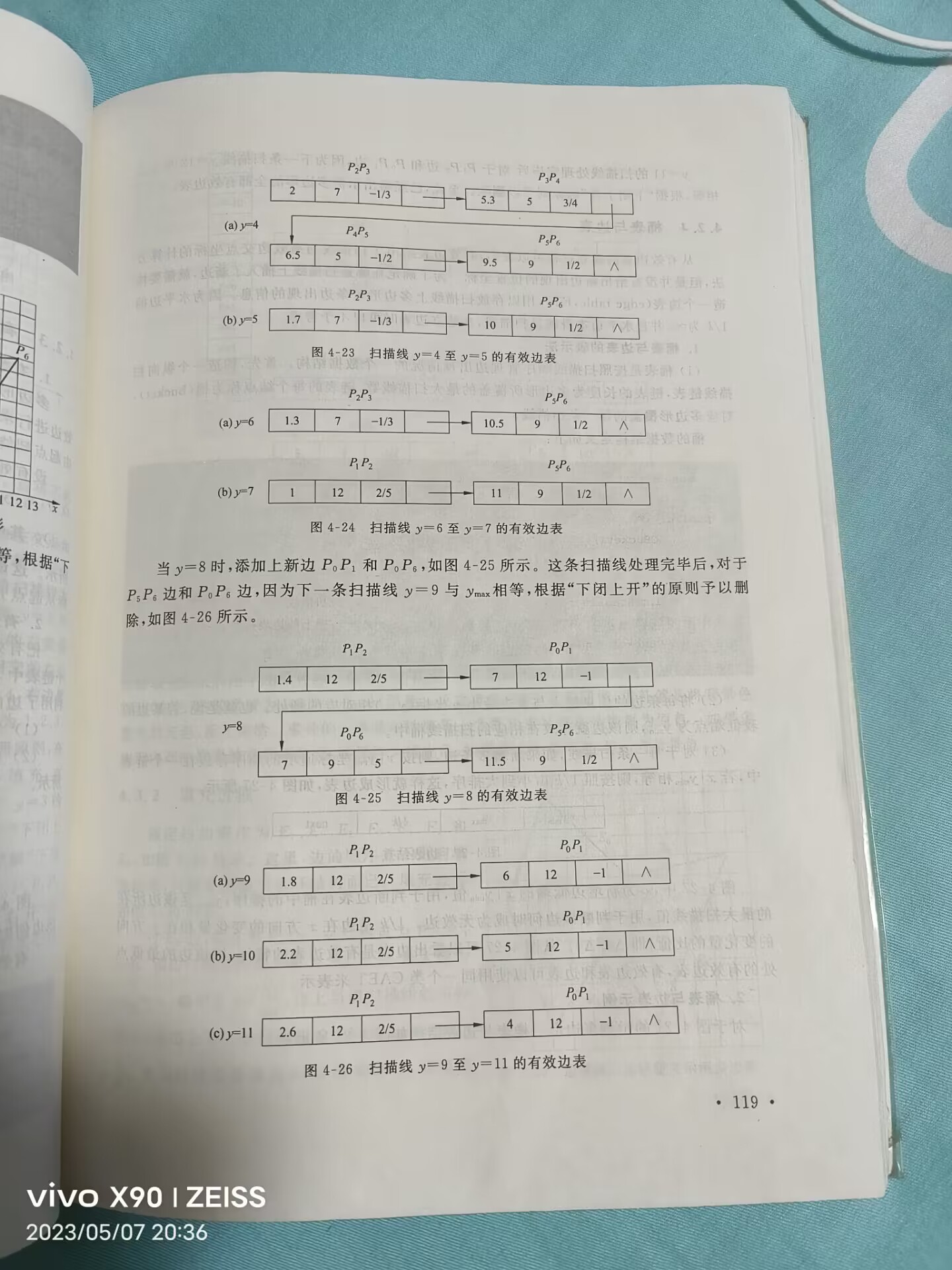
1/k=(Phigh.x-Plow.x)/( (Phigh.y-Plow.y)=(3-7)/(12-8)=-1

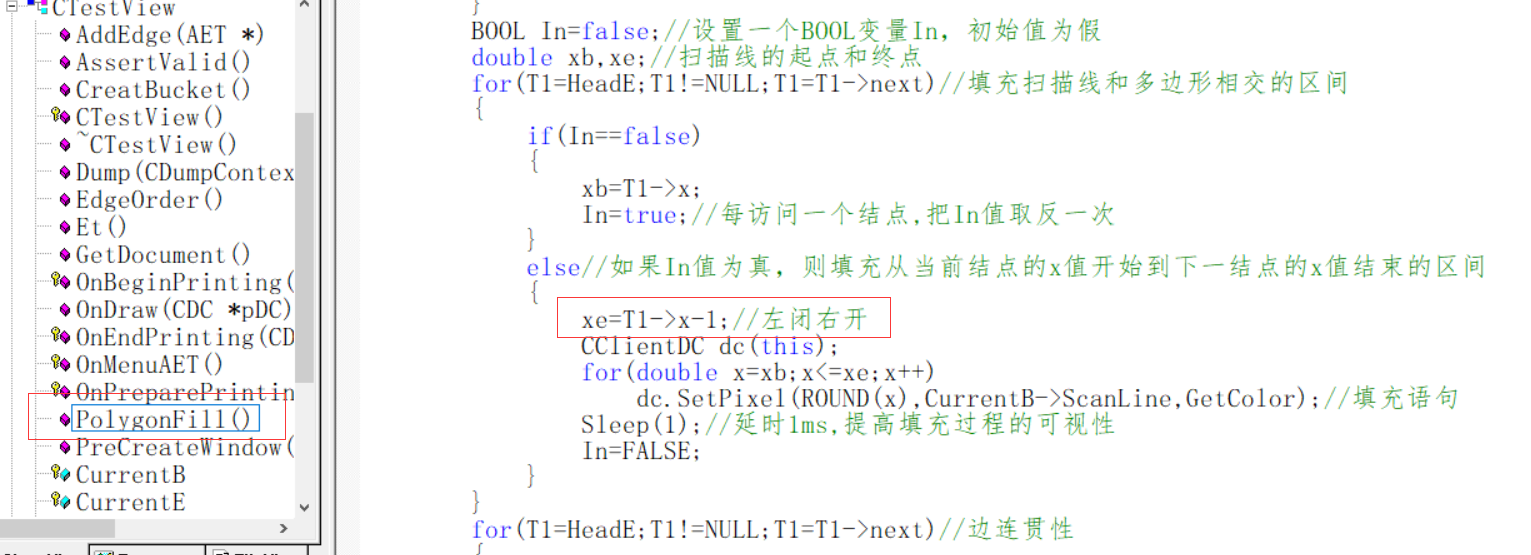
## 填充

* （4）桶结点循环填充
* y坐标=scanLine;
* ①将边表中的边插入到AET中
* ②删除结束的边，即scanLine=yMax的边
* ③AET排序
* ④对AET循环遍历，两两节点配对填充；
* ⑤对AET中每条边x值加1/k;

填充过程见下面两个图片。







|  |
| --- |
| 12 |
| 11 |
| 10 |
| 9 |
| 8 |
| 7 |
| 6 |
| 5 |
| 4 |
| 3 |
| 2 |
| 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 12 | 2/5 | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 9 | 1/2 | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 12 | -1 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 9 | 5 | ^ |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 5 | -1/2 |  |

^

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 5 | 3/4 |  |

^

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 7 | -1/3 |  |

^

^

scanLine

AET