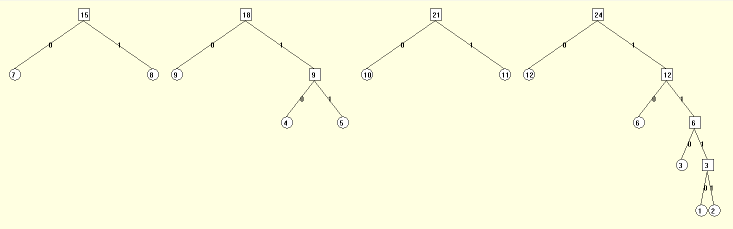
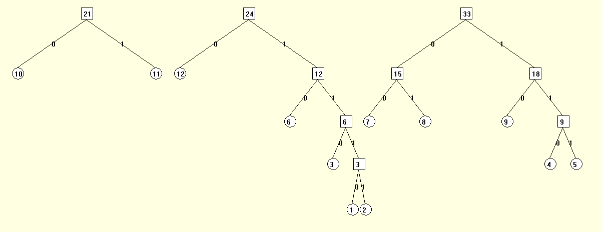
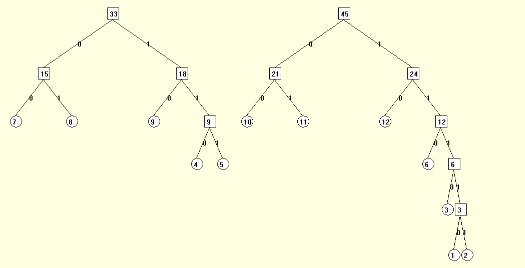
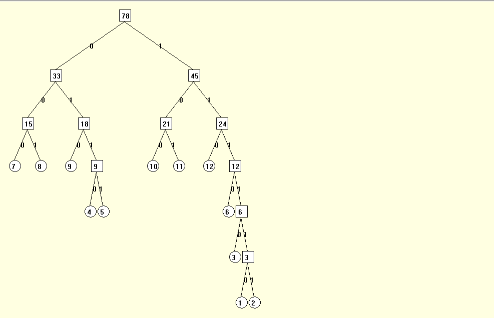
**如何使用Win32API绘制树**









一、这个已经有几年时间了，刚开始学习charlie的《windows程序设计》的时候做的。  
现在看来，代码很乱，虽然后来还整理过几次，现在这方面没什么兴趣了，有兴趣的可自由下载。

二、绘制二叉树的伪代码

int drawBinaryTree(BinaryTree bt){

Queue main;

Queue slave;

mian.enQueue(head node of bt);

while (!empty(main)){

slave.clear();

for each node in main{

drawNode(node);

slave.enQueue(sons of node);

}

main.clear();

copy all nodes in slave ==> main;

}

return 0;

}

三、一些数据结构

////

void init(int nElem);//初始化

void push(int elem); //数据入栈auxBuffer

int StackEmpty(); //SeniorBuffer无数据则返回真值

void copy(int \*src, int \*dest);//copy auxBuffer to SeniorBuffer

void destroy();

//绘制哈夫曼树时存储结点序号的栈,采用双缓冲结构

typedef struct tagSTACK{

int \* SeniorBuffer; //用于暂存结点序号

int sp1; //元素个数

int \* auxBuffer; //缓冲区

int sp2; //缓冲区元素个数

void ( \*init )(int nElem); //初始化

void ( \*push )(int elem); //数据入栈auxBuffer

int ( \*StackEmpty )(); //SeniorBuffer无数据则返回真值

void ( \*copy )(int \*src, int \*dest);//copy auxBuffer to SeniorBuffer

void ( \*destroy)();

} STACK;

函数指针，可以模仿C++成员函数调用语法，《面向对象软件构造》附录内也有有提到。

四、绘制森林

最开始的绘制效果结点多的时候容易碰撞，通过观察得出结论：绘制头节点时，手背要尽可能长，两背夹角应该尽可能大，越往下手背越短，夹角变小。  
不过过也不能一直变短变小，到一定程度可以固定下来。  
这样几百个节点也比最初效果好了。  
这段代码很琐碎，我自己都不想看。

void PrepareForest(ForestDemension\* \_G\_fd){

\_G\_fd->MaxNodeNO = \_G\_fd->ForestWidth = \_G\_fd->ForestHeight = 0;

}

void EnumForestStructMember(PHuffmantreeNode ht, int NodeNO, ForestDemension\* \_G\_fd){

int node = NodeNO;

\_G\_fd->rNodeNO = \_G\_fd->lNodeNO = 0;

while( node && ht[node].lchild ) {\_G\_fd->lNodeNO++; node = ht[node].lchild;}

node = NodeNO;

while( node && ht[node].rchild ) {\_G\_fd->rNodeNO++; node = ht[node].rchild;}

if (\_G\_fd->lNodeNO > \_G\_fd->MaxNodeNO) \_G\_fd->MaxNodeNO = \_G\_fd->lNodeNO;

if (\_G\_fd->rNodeNO > \_G\_fd->MaxNodeNO) \_G\_fd->MaxNodeNO = \_G\_fd->rNodeNO;

}

int GetTreeOrigX( ForestDemension \*\_G\_fd ){

double m = 0;

if ( \_G\_fd->lNodeNO > -1 ) m = m + EDGE;

if ( \_G\_fd->lNodeNO > 0 ) m = m + 138;

if ( \_G\_fd->lNodeNO > 1 ) m = m + 56;

if ( \_G\_fd->lNodeNO > 2 ) m = m + 26;

if ( \_G\_fd->lNodeNO > 3 ) m = m + 13.7 \* (\_G\_fd->lNodeNO-3);

return (int)(m) + \_G\_fd->ForestWidth ;

}

void SetForestWidth( ForestDemension \*\_G\_fd ){

double m = 0;

if ( \_G\_fd->lNodeNO > -1 ) m = m + EDGE;

if ( \_G\_fd->lNodeNO > 0 ) m = m + 138;

if ( \_G\_fd->lNodeNO > 1 ) m = m + 56;

if ( \_G\_fd->lNodeNO > 2 ) m = m + 26;

if ( \_G\_fd->lNodeNO > 3 ) m = m + 13.7 \* (\_G\_fd->lNodeNO-3);

if ( \_G\_fd->rNodeNO > -1 ) m = m + EDGE;

if ( \_G\_fd->rNodeNO > 0 ) m = m + 138;

if ( \_G\_fd->rNodeNO > 1 ) m = m + 56;

if ( \_G\_fd->rNodeNO > 2 ) m = m + 26;

if ( \_G\_fd->rNodeNO > 3 ) m = m + 13.7 \* (\_G\_fd->lNodeNO-3);

\_G\_fd->ForestWidth += (int)m;

}

void SetForestHeight( ForestDemension \*\_G\_fd ){

double m = 0.1\*TREELENGTH;

if ( \_G\_fd->MaxNodeNO < HIERARCHY ) \_G\_fd->MaxNodeNO = HIERARCHY;

if ( \_G\_fd->MaxNodeNO > -1 ) m = m + 18\*EDGE;

if ( \_G\_fd->MaxNodeNO > 0 ) m = m + 80+EDGE;

if ( \_G\_fd->MaxNodeNO > 1 ) m = m + 56+EDGE;

if ( \_G\_fd->MaxNodeNO > 2 ) m = m + 46+EDGE;

if ( \_G\_fd->MaxNodeNO > 3 ) m = m + (51+EDGE) \* (\_G\_fd->MaxNodeNO-3);

if ( (int)m > \_G\_fd->ForestHeight ) \_G\_fd->ForestHeight = (int)m;

}