# 实验五 二叉树

【实验项目】：完成二叉树的基本运算，从而了解其基本特征，基本运算算法

## 一、实验目的

1. 理解和掌握二叉树的类型定义方法。
2. 掌握二叉树的基本运算，包括创建、二叉树深度，叶子结点的计算，二叉树的各种遍历方法，学习利用二叉树解决实际问题

## 二、实验要求（本次实验要求上交）

**【题目1】----验证型**

(1)      编写一个程序，实现二叉树的各种运算，并在此基础上设计一个主程序完成如下功能。要求尽量能不用递归进行运算。（参考代码见文件夹5-1）

（1）        输出二叉树B；

（2）        输出二叉树B的深度；

（3）        输出H结点的左、右孩子结点值；

（4）        输出二叉树B的宽度；

（5）        输出二叉树B的结点个数；

（6）        输出二叉树B的叶子结点个数；

**【题目2】**

编写一个程序，实现二叉树的层次遍历的算法，要求自己以括号表示法录入一棵二叉树，对该二叉树进行遍历，并输出遍历结果，并统计度为2的结点的数目。

|  |
| --- |
| 请将源程序附录于此： |
| #include <stdio.h>  #include <malloc.h>  #define MaxSize 100  typedef char ElemType;  typedef struct node  {  ElemType data; //数据元素  struct node \*lchild; //指向左孩子  struct node \*rchild; //指向右孩子  } BTNode;  void CreateBTNode(BTNode \*&b,char \*str) //由str串创建二叉链  {  BTNode \*St[MaxSize],\*p=NULL;  int top=-1,k,j=0;  char ch;  b=NULL; //建立的二叉树初始时为空  ch=str[j];  while (ch!='\0') //str未扫描完时循环  {  switch(ch)  {  case '(':top++;St[top]=p;k=1; break; //为左节点  case ')':top--;break;  case ',':k=2; break; //为右节点  default:p=(BTNode \*)malloc(sizeof(BTNode));  p->data=ch;p->lchild=p->rchild=NULL;  if (b==NULL) //p指向二叉树的根节点  b=p;  else //已建立二叉树根节点  {  switch(k)  {  case 1:St[top]->lchild=p;break;  case 2:St[top]->rchild=p;break;  }  }  }  j++;  ch=str[j];  }  }  int LevelOrder(BTNode \*b)  {  BTNode \*p;  BTNode \*qu[MaxSize];  int front,rear,i=0;  front = rear = -1;  rear++;  qu[rear] = b;  while(front != rear)  {  front = (front+1)%MaxSize;  p = qu[front];  printf("%c",p->data);  if(p->rchild!=NULL && p->lchild!=NULL)  {  i++;  }  if(p->lchild!=NULL)  {  rear = (rear+1)%MaxSize;  qu[rear] = p->lchild;  }  if(p->rchild!=NULL)  {  rear = (rear+1)%MaxSize;  qu[rear] = p->rchild;  }  }  return i;  }  void DispBTNode(BTNode \*b) //以括号表示法输出二叉树  {  if (b!=NULL)  {  printf("%c",b->data);  if (b->lchild!=NULL || b->rchild!=NULL)  {  printf("(");  DispBTNode(b->lchild);  if (b->rchild!=NULL) printf(",");  DispBTNode(b->rchild);  printf(")");  }  }  printf("\n");  }  void DestroyBTNode(BTNode \*&b)  {  if (b!=NULL)  {  DestroyBTNode(b->lchild);  DestroyBTNode(b->rchild);  free(b);  }  }  int main(){  int i,j,m,n;  BTNode \*b,\*p,\*s,\*R;  char \*str="A(B(D,E(H(J,K(L,M(,N))))),C(F,G(,I)))";  CreateBTNode(b,str);  i=LevelOrder(b);  DispBTNode(b);  printf("\n度数为2的节点的个数是:%d\n",i);  DestroyBTNode(b);  return 0;  } |
| 请将运行结果图附录于此 |
| C:\Users\Administrator\AppData\Roaming\Tencent\Users\1061892125\QQ\WinTemp\RichOle\6AX_RYG1B7KH$1OTTGC6FEA.png |

**【题目3】**

编写一个程序，实现二叉树的中序遍历的非递归算法。并统计叶子结点的数目。

|  |
| --- |
| 请将源程序附录于此： |
| #include <stdio.h>  #include <malloc.h>  #define MaxSize 100  typedef char ElemType;  typedef struct node  {  ElemType data; //数据元素  struct node \*lchild; //指向左孩子  struct node \*rchild; //指向右孩子  } BTNode;  void CreateBTNode(BTNode \*&b,char \*str) //由str串创建二叉链  {  BTNode \*St[MaxSize],\*p=NULL;  int top=-1,k,j=0;  char ch;  b=NULL; //建立的二叉树初始时为空  ch=str[j];  while (ch!='\0') //str未扫描完时循环  {  switch(ch)  {  case '(':top++;St[top]=p;k=1; break; //为左节点  case ')':top--;break;  case ',':k=2; break; //为右节点  default:p=(BTNode \*)malloc(sizeof(BTNode));  p->data=ch;p->lchild=p->rchild=NULL;  if (b==NULL) //p指向二叉树的根节点  b=p;  else //已建立二叉树根节点  {  switch(k)  {  case 1:St[top]->lchild=p;break;  case 2:St[top]->rchild=p;break;  }  }  }  j++;  ch=str[j];  }  }  void InOrder1(BTNode \*b) //中序遍历的非递归算法  {  BTNode \*St[MaxSize],\*p;  int top = -1;  if(b!=NULL)  {  p = b;  while(p!=NULL || top>-1) //处理\*p节点的左子树  { //扫描\*p的所有左节点并进栈  while(p!=NULL)  {  top++;  St[top] = p;  p = p->lchild;  }  //执行到此处时，栈顶元素没有左孩子或左子树均已访问过  if(top>-1)  {  p = St[top]; //出栈 \*p节点  top--;  printf("%c",p->data); //访问之  p = p->rchild; //转向处理\*p的右孩子节点  }  }  printf("\n");  }  }  void DispBTNode(BTNode \*b) //以括号表示法输出二叉树  {  if (b!=NULL)  {  printf("%c",b->data);  if (b->lchild!=NULL || b->rchild!=NULL)  {  printf("(");  DispBTNode(b->lchild);  if (b->rchild!=NULL) printf(",");  DispBTNode(b->rchild);  printf(")");  }  }  printf("\n");  }  int LeafNodes(BTNode \*b) //求二叉树b的叶子节点个数  {  int num1,num2;  if (b==NULL)  return 0;  else if (b->lchild==NULL && b->rchild==NULL)  return 1;  else  {  num1=LeafNodes(b->lchild);  num2=LeafNodes(b->rchild);  return (num1+num2);  }  }  void DestroyBTNode(BTNode \*&b)  {  if (b!=NULL)  {  DestroyBTNode(b->lchild);  DestroyBTNode(b->rchild);  free(b);  }  }  int main(){  int i;  BTNode \*b,\*p,\*s,\*R;  char \*str="A(B(D,E(H(J,K(L,M(,N))))),C(F,G(,I)))";  CreateBTNode(b,str);  InOrder1(b);  i = LeafNodes(b);  printf("叶子节点的数目是:%d\n",i);  return 0;  } |
| 请将运行结果图附录于此 |
| C:\Users\Administrator\AppData\Roaming\Tencent\Users\1061892125\QQ\WinTemp\RichOle\9`)_$~L1H8G[~D~EM7@KIYU.png |

## 实验上传方法：

将源程序文件和本word文档（添加了相应的源程序和截图）上传。多个文件请将其放入一个文件夹压缩后上传。请于指定时间前上交，过时不补！

**附：二叉树部分习题。完成习题练习，加强对二叉树的理解。习题部分不交**