实验06 树及应用

实验周次：第13周 学时：2学时 地点：学院机房

学号：3190707121 姓名：武新纪 班级：人工智能191 序号：21

提示：请务必填写 以上个人信息。

实验题1计50分，实验题2计50分。

**【实验目的】**

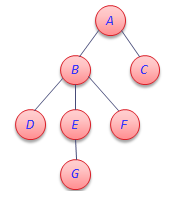
1.掌握树的双亲存储结构及其相关算法设计。

2.掌握树的孩子链存储结构及相关算法设计。

3.掌握树的孩子兄弟链存储结构及其相关算法设计。

**【实验内容】**

给出一棵树T如下图，



**实验题1**. 请以双亲顺序存储结构，然后给出查找某结点如E，的双亲结点的算法设计。要求：1）给出双亲存储结构的类型定义；2）设计算法构建双亲顺序存储结构存储树T；3）给出查找某结点的算法；

**程序:**

#include <stdio.h>

#include <stdlib.h>

#define MaxSize 20

typedef char ElemType; //宏定义树结构中数据类型

typedef struct Snode

{ //结点结构

    ElemType data;

    int parent;

} PNode;

typedef struct

{ //树结构

    PNode tnode[MaxSize];

    int n; //结点个数

} PTree;

PTree CreatePNode(PTree tree)

{ //创建节点

    int i, j;

    char ch;

    printf("请输入节点个数:\n");

    scanf("%d", &(tree.n));

    printf("请输入结点的值其双亲位于数组中的位置下标:\n");

    for (i = 0; i < tree.n; i++)

    {

        getchar();

        scanf("%c %d", &ch, &j);

        tree.tnode[i].data = ch;

        tree.tnode[i].parent = j;

    }

    return tree;

}

void FindParent(PTree tree)

{ //查找每个节点

    char a;

    int isfind = 0, i;

    printf("请输入要查询的结点值:\n");

    getchar();

    scanf("%c", &a);

    for (i = 0; i < tree.n; i++)

    {

        if (tree.tnode[i].data == a)

        {

            isfind = 1;

            int ad = tree.tnode[i].parent;

            printf("%c的父节点为%c,存储位置下标为%d\n", a, tree.tnode[ad].data, ad);

            break;

        }

    }

    if (isfind == 0)

    {

        printf("树中无此节点\n");

    }

}

void DispTree(PTree tree)

{ //显示树

    int i;

    printf("index\tdata\tparent\n");

    for (i = 0; i < tree.n; i++)

        printf("%d\t%c\t%d\n", i, tree.tnode[i].data, tree.tnode[i].parent);

}

int main()

{

    int i;

    PTree tree;

    for (i = 0; i < MaxSize; i++)

    {

        tree.tnode[i].data = ' ';

        tree.tnode[i].parent = 0;

    }

    tree = CreatePNode(tree);

    printf("该树如下:\n");

    DispTree(tree);

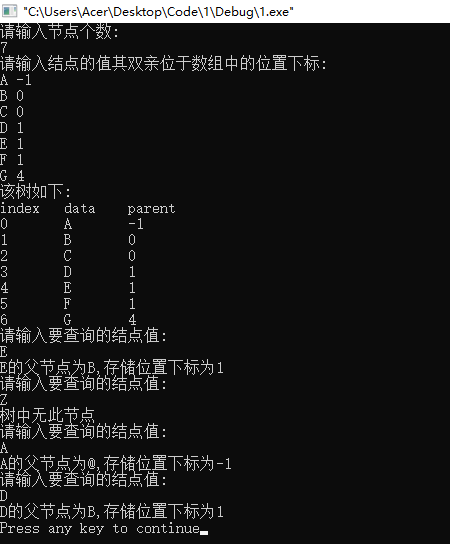
    for (i = 0; i < 4; i++)

        FindParent(tree);

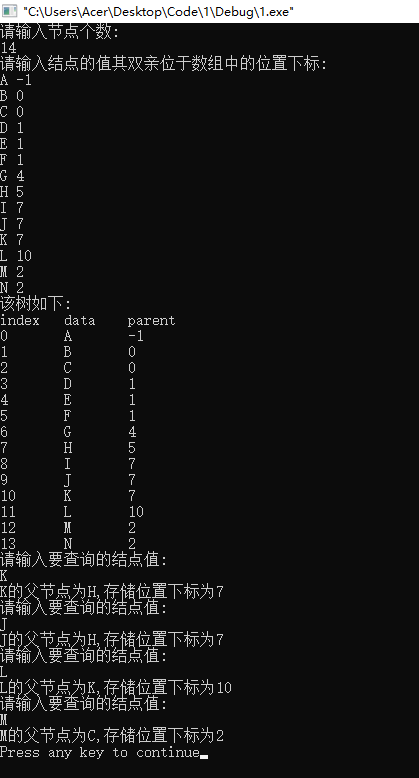
    return 0;

}

**调试结果1**



调试结果2:



**实验题2.**请以**孩子链**作为树的存储结构，设计一个求树T高度的递归算法，要求：1）给出孩子链存储结构类型定义（孩子结点指针域最多3个）；2）设计算法构建孩子链存储结构存储树T；3）给出求树T高度的递归算法：①先给出递归模型（包括递推出口和递归体）；②给出递归算法设计。

**程序:**

#include <stdio.h>

#include <malloc.h>

#define MaxSons 10

#define MaxSize 100

typedef struct node

{

    char data;                  //节点的值

    struct node \*sons[MaxSons]; //指向孩子节点

} TSonNode;                     //孩子链存储结构类型

TSonNode \*CreateTree(char \*str) //由str建立孩子链存储结构

{

    struct

    {

        TSonNode \*nodep; //节点指针

        int num;         //孩子个数

    } St[MaxSize];       //定义顺序栈

    int top = -1;        //栈顶指针

    int i = 0, j;

    char ch = str[i];

    TSonNode \*t = NULL, \*p;

    while (ch != '\0')

    {

        switch (ch)

        {

        case '(':

            top++;

            St[top].nodep = p;

            St[top].num = 0; //当前节点\*p进栈

            break;

        case ')':

            top--;

            break; //退栈

        case ',':

            St[top].num++;

            break; //栈顶节点增加一个孩子

        default:

            p = (TSonNode \*)malloc(sizeof(TSonNode));

            p->data = ch;                 //建立一个节点p存放ch

            for (j = 0; j < MaxSons; j++) //所有孩子指针置为NULL

                p->sons[j] = NULL;

            if (t == NULL) //原为空树

                t = p;

            else //将其作为栈顶节点的一个孩子

                St[top].nodep->sons[St[top].num] = p;

            break;

        }

        i++;

        ch = str[i];

    }

    return t;

}

void DispTree(TSonNode \*t) //输出孩子链存储结构

{

    int i;

    if (t != NULL)

    {

        printf("%c", t->data);

        if (t->sons[0] != NULL) //t节点至少有一个孩子

        {

            printf("("); //输出一个左括号

            for (i = 0; i < MaxSons; i++)

            {

                DispTree(t->sons[i]);

                if (t->sons[i + 1] != NULL) //如果有下一个孩子

                    printf(",");            //输出一个','

                else                        //如果没有下一个孩子

                    break;                  //退出循环

            }

            printf(")"); //输出一个右括号

        }

    }

}

void DestroyTree(TSonNode \*&t) //销毁树t

{

    int i;

    if (t != NULL)

    {

        for (i = 0; i < MaxSons; i++)

        {

            if (t->sons[i] != NULL)      //有子树

                DestroyTree(t->sons[i]); //销毁该子树

            else                         //再没有子树

                break;                   //退出循环

        }

        free(t); //释放根节点

    }

}

int TreeHeight(TSonNode \*t) //求树t高度

{

    TSonNode \*p;

    int i, h, maxh = 0;

    if (t == NULL)

        return 0; //空树返回高度0

    else          //处理非空树

    {

        for (i = 0; i < MaxSons; i++)

        {

            p = t->sons[i]; //p指向t的第i-1个孩子节点

            if (p != NULL)  //若存在第i-1个孩子

            {

                h = TreeHeight(p); //求出对应子树的高度

                if (maxh < h)

                    maxh = h; //求所有子树的最大高度

            }

        }

        return (maxh + 1); //返回maxh+1

    }

}

int main()

{

    TSonNode \*t;

    printf("请以括号和逗号形式表示树:");

    char c[MaxSize];

    scanf("%s", c);

    t = CreateTree(c);

    printf("T:");

    DispTree(t);

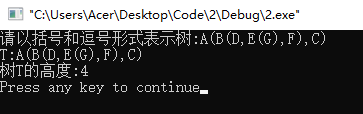
    printf("\n树T的高度:%d\n", TreeHeight(t));

    DestroyTree(t);

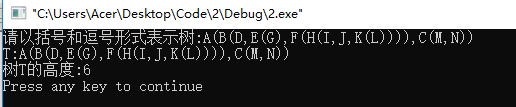
    return 0;

}

**调试结果1：**



**调试结果2:**



**实验题3**. 请以**孩子兄弟链**作为树的存储结构，设计一个求树T高度的递归算法，要求：1）给出孩子兄弟链存储结构类型定义；2）设计算法构建孩子兄弟链存储结构存储树T；3）给出求树T高度的递归算法：①先给出递归模型（包括递推出口和递归体）；②给出递归算法设计。

**程序:**

#include <stdio.h>

#include <malloc.h>

#define MaxSize 100

typedef struct tnode

{

    char data;        //节点的值

    struct tnode \*hp; //指向兄弟

    struct tnode \*vp; //指向孩子节点

} TSBNode;            //孩子兄弟链存储结构类型

TSBNode \*CreateTree(const char \*str) //由str建立孩子兄弟链存储结构

{

    struct

    {

        TSBNode \*nodep; //节点指针

        int num;        //孩子个数

    } St[MaxSize];      //顺序栈

    int top = -1;       //栈顶指针

    int i = 0, j;

    char ch = str[i];

    TSBNode \*t = NULL, \*p, \*q;

    while (ch != '\0')

    {

        switch (ch)

        {

        case '(':

            top++;

            St[top].nodep = p;

            St[top].num = 0; //当前节点p进栈

            break;

        case ')':

            top--;

            break; //退栈

        case ',':

            St[top].num++;

            break; //栈顶节点增加一个孩子

        default:

            p = (TSBNode \*)malloc(sizeof(TSBNode));

            p->data = ch; //建立一个节点p存放ch

            p->hp = p->vp = NULL;

            if (t == NULL) //原为空树

                t = p;

            else //将其作为栈顶节点的一个孩子

            {

                if (St[top].num == 0) //第一个孩子用vp指向它

                    St[top].nodep->vp = p;

                else //其他孩子用栈顶节点的孩子节点的hp指向它

                {

                    q = St[top].nodep->vp;

                    for (j = 1; j < St[top].num; j++)

                        q = q->hp;

                    q->hp = p;

                }

            }

            break;

        }

        i++;

        ch = str[i];

    }

    return t;

}

void DispTree(TSBNode \*t) //输出孩子兄弟链存储结构

{

    TSBNode \*p;

    if (t != NULL)

    {

        printf("%c", t->data);

        if (t->vp != NULL) //有孩子时输出一个'('

        {

            printf("(");

            p = t->vp; //p指向节点t的第一个孩子

            while (p != NULL)

            {

                DispTree(p);

                p = p->hp;

                if (p != NULL)

                    printf(",");

            }

            printf(")"); //输出一个')'

        }

    }

}

void DestroryTree(TSBNode \*&t) //销毁树t

{

    if (t != NULL)

    {

        DestroryTree(t->vp);

        DestroryTree(t->hp);

        free(t); //释放根节点

    }

}

int TreeHeight(TSBNode \*t)

{

    TSBNode \*p;

    int h, maxh = 0;

    if (t == NULL)

        return 0; //空树返回0

    else

    {

        p = t->vp;        //指向第1个孩子节点

        while (p != NULL) //扫描t的所有子树

        {

            h = TreeHeight(p); //求出p子树的高度

            if (maxh < h)

                maxh = h; //求所有子树的最大高度

            p = p->hp;    //继续处理t的其他子树

        }

        return (maxh + 1); //返回maxh+1

    }

}

int main()

{

    TSBNode \*t;

    char c[MaxSize];

    printf("请以括号和逗号形式表示树:");

    scanf("%s", c);

    t = CreateTree(c);

    printf("树T为:");

    DispTree(t);

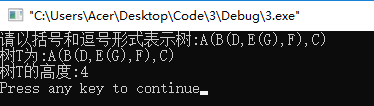
    printf("\n树T的高度:%d\n", TreeHeight(t));

    DestroryTree(t);

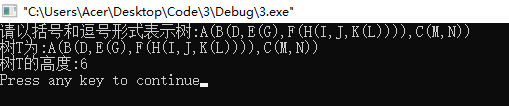
    return 0;

}

**调试结果1：**

****

**调试结果2：**

****