**数据结构实验报告**

**实验题目：输出二叉树形态**

|  |  |  |  |
| --- | --- | --- | --- |
| 班级：2020211305 | 姓名：缪奇志 | 学号：2020211345 | 分工：需求分析、概要设计 |
| 班级：2020211305 | 姓名：倪玮昊 | 学号：2020211346 | 分工：测试结果、用户使用说明 |
| 班级：2020211305 | 姓名：杨智杰 | 学号：2020211356 | 分工：详细设计、调试分析报告 |

1. **需求分析**

题目任务：

用先序递归过程建立二叉树 (存储结构：二叉链表)，要求：输出二叉树的形态。

程序的输入：

输入数据按先序遍历所得序列输入，当某结点左子树或右子树为空时，输入‘\*’号。因此，此处程序的输入只能是一串由小写字母和\*号构成的字符串。

程序的输出：

打印出按照先序递归所构建的二叉树的形态。因此在打印时需要注意打印的格式要和二叉数的形态相匹配。

程序功能实现（主要分为4个模块）：

1. 模块1：输入按照题目要求合法的一串字符 。按照这段按先序递归排序的字符完成二叉树的创建与储存。
2. 模块2：在完成二叉树的构建之后利用递归算法，找出构建的二叉树的树的高度。
3. 模块3：按照层次遍历的顺序，实现二叉数的存储以及按照规定二叉树形态的打印。
4. 功能4：在完成前三个模块之后实现二叉树空间的释放，即清除二叉树。

正确数据输出结果范例：

手机屏幕的截图

中度可信度描述已自动生成

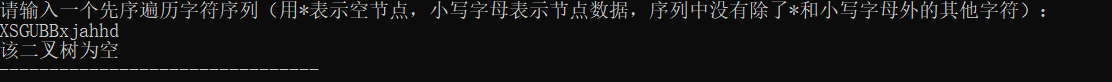
屏幕上有字

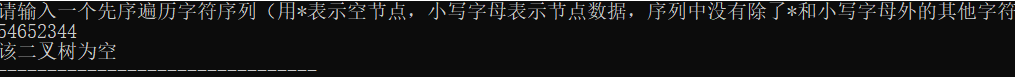
中度可信度描述已自动生成

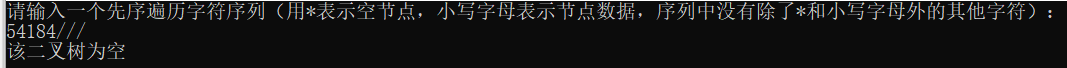
文本

描述已自动生成

错误数据输出结果范例：

****

****

****

1. **概要设计**

**算法思路概述：**

本程序主要由四个主要的功能模块组成。

1. 模块一：按照输入的先序递归顺序的字符串完成二叉树的创建 思路：先创建好对应的数据结构，完成之后合理的利用该数据结构，同时利用递归的思路按照先序遍历完成二叉树元素的录入，同时构建好二叉树之间的逻辑关系。
2. 模块二：计算出模块一所构建的二叉树的树的度。 思路：同样使用递归的算法完成二叉树树的高度的计算，递归同样使用先序递归的思路，每往下一层成功递归一次，计数加一，最后会对根左边和根右边的高度进行比较，较大者就是树的高度。
3. 模块三：按照层级顺序完成二叉树的存储和打印出二叉树的形态。思路：根据模块二查找出的二叉树的高度对每一层的理论最大元素个数依次存储在对应数组元素里，同时利用函数动态申请一组空间用来重新存储按照层次顺序排列的二叉树元素的地址，最后使用两个循环嵌套实现按层次的存储和二叉树形态的打印。
4. 模块四：完成对二叉树所占存储空间的释放。 思路：利用后序递归的思路完成对二叉树所占空间的释放。

**数据结构类型的定义：**

定义一个字符的变量EleTypee用来表示二叉树中的数据元素。

typedef char EleType;

定义结构体BTnode用来表示一个二叉树节点，BiTree用来表示指向一个二叉树的指针。

typedef struct BTnode

{

EleType data;

BTnode\* lc;

BTnode\* rc;

}\* BiTree;

**主程序流程：**

int main()

{

BiTree T = NULL;

bool ERROR = false;

printf("请输入一个先序遍历字符序列（用\*表示空节点，小写字母表示节点数据，序列中没有除了\*和小写字母外的其他字符）：\n");

PreInitial(T, ERROR);

int h = GetHeight(T);

LevelOutput(T, h);

Clear(T);

return 0;

}

**各层次模块层次调用关系：**

main先调用了PreInitial函数实现按照先序递归的顺序完成二叉树的构建，再接着调用GetHeight函数完成对指定二叉树的树的高度的计算，再调用 LevelOutput函数实现二叉树按照层次的存储以及最后打印出二叉树的形态，最终调用Clear函数完成二叉树的空间释放。

1. **详细设计**

**各种函数的设计：**

1. 按照先序遍历构造二叉树：

void PreInitial(BiTree& T, bool& Error) // Error判断输入字符是否非法

{

if (Error)

then return;

EleType e = getchar();

if (e == '\*')

then return;

else if (e <'a'||e>'z')

then Error <- true;

return;

if (!T)

then

T = (BTnode\*)malloc(sizeof(BTnode));

T->data <- e;

T->lc <- NULL;

T->rc <- NULL;

PreInitial(T->lc, Error);

PreInitial(T->rc, Error);

}

2.求出指定二叉树的树的高度：

int GetHeight(BiTree T)

{

if (!T)

then return 0;

int len1 <- GetHeight(T->lc) + 1;

int len2 <- GetHeight(T->rc) + 1;

return len1 >= len2 ? len1 : len2;

}

3. 按照层序输出二叉树形态：

void LevelOutput(BiTree T, int H)

{

if (!T)

then

printf("该二叉树为空");

return;

printf("该二叉树结构如下：\n");

int pow2[100] <- { 1 };

for i <- 1 to H /\*通过循环写出完全二叉树每一行的元素个数\*/

pow2[i] <- pow2[i - 1] \* 2;

int space <- pow2[H]; /\*space指的是二叉树最后一行再加一行的最大元素个数，后续主要用来保证打印格式正确\*/

BTnode\*\* temp <- (BTnode\*\* )malloc(sizeof(BTnode\* ) \* pow2[H]);

temp[1] <- T;

for i <- 1 to H /\*通过两个循环嵌套实现二叉树每个元素按层次存入到temp数组\*/

{

for j <- pow2[i] to pow2[i+1]

{

if (temp[j])

then

temp[2 \* j] <- temp[j]->lc;

temp[2 \* j + 1] <- temp[j]->rc;

else

temp[2 \* j] <- NULL;

temp[2 \* j + 1] <- NULL;

}

}

putchar('\n');

for i <- 1 to H

{

for j <- pow2[i] to pow2[i+1] /\*输出一行\*/

{

for k <- 0 to space-1 // 输出空格（格式）

putchar(' ');

if (temp[j])

then putchar(temp[j]->data);

else

then putchar(' ');

for k <- 0 to space // 输出空格（格式）

putchar(' ');

}

putchar('\n');

space /= 2;

}

}

4. 完成二叉树的清除：

{

if (!T)

then return;

if (T->lc)

then Clear(T->lc);

if (T->rc)

then Clear(T->rc);

free(T);

T <- NULL;

}

**函数调用关系图：**

手机屏幕的截图

中度可信度描述已自动生成

1. **调试分析报告**

**调试过程中遇到的问题：**

在创建二叉树的时候忘记传入结果导致主函数里的二叉树没有初始化，导致结果错误。同时为了实现二叉树的形态的正确输出，花费了很大的经历，最后采用空格的办法帮助实现二叉树形态的打印。

**算法时空分析：**

PreInitial:o(n)

GetHeight:o(n)

LevelOutput:o(h\*2^h)

Clear:o(n)

main:o(h\*2^h)

**改进设想：**

在打印二叉树时希望可以减少一次储存数据，这样可以减少程序所占的内存空间，同时可以实现二叉树打印时每个元素和它的子节点使用线段连接，这样可以使得二叉树的形态更加明显。

**经验体会**：

对先序遍历以及其他二叉树遍历方式有了进一步深刻的认识，在输入时应当注意严格按照先序递归的序列，同时在面对二叉树的各种操作时，灵活采用递归算法可以提供比较节俭的代码以及较简单的思路，但是需要注意的是在使用递归算法时应该要注意到设置正确的算法确保递归能顺利返回。

1. **用户使用说明**

步骤如下：

用户打开程序，输入按照要求的指定的字符串，要求字符串是按照先序递归的排列二叉树输入，同时要求只能含有小写字母和\*符号，小写字母即表示二元组数据，如果元素为空就使用\*表示。

1. **测试结果**

文本

描述已自动生成

手机屏幕的截图

中度可信度描述已自动生成

文本

描述已自动生成

屏幕上有字

中度可信度描述已自动生成