# 树结构的课堂作业

2009853E-I011-0084

毛一凡

## 图示 描述已自动生成储存结构

（图建议放缩至175%观看）

用列表存储10个顶级合成物合成森林（主要目的：方便遍历时较容易返回顶级合成物的名称）。剩下的结构用链表储存。

## 代码

**构建链表结点**

**typedef** struct Obj**{**

const char **\***name**;**

struct Obj **\***fchild**,** **\***schild**,** **\***tchild **;**

**}**Obj**;**

**使用函数嵌套的方式建立起树结构**

/\*

一级材料

\*/

int init\_ironingot(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h){

(h->name) = "铁锭";

h->fchild = NULL;

h->schild = NULL;

h->tchild = NULL;

\*p = h;

}

return OK;

}

int init\_goldingot(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h) {

(h->name) = "金锭";

h->fchild = NULL;

h->schild = NULL;

h->tchild = NULL;

}

\*p = h;

return OK;

}

int init\_copperingot(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h) {

h->name = "铜锭";

h->fchild = NULL;

h->schild = NULL;

h->tchild = NULL;

}

\*p = h;

return OK;

}

int init\_log(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h) {

h->name = "原木";

h->fchild = NULL;

h->schild = NULL;

h->tchild = NULL;

\*p = h;

}

return OK;

}

int init\_gossamer(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h) {

(h->name) = "蜘蛛丝";

h->fchild = NULL;

h->schild = NULL;

h->tchild = NULL;

}

\*p = h;

return OK;

}

/\*

二级材料

\*/

int init\_board(Obj\*\* p) {

Obj\* h = (Obj\*)malloc(sizeof(Obj));

if (h) {

(h->name) = "木板";

init\_log(&(h->fchild));

h->schild = NULL;

h->tchild = NULL;

}

\*p = h;

return OK;

}

/\*

三级材料

\*/

int init\_stick(Obj \*\*p) {

Obj \*h = (Obj\*)malloc(sizeof(Obj));

if (h) {

(h->name) = "木棒";

init\_board(&(h->fchild));

h->schild = NULL;

h->tchild = NULL;

}

\*p = h;

return OK;

}

/\*

四级材料

\*/

int init\_ironaxe(Obj \*\*p) {

Obj \*h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

(h1->name) = "铁斧头";

init\_stick(&(h1->schild));

init\_ironingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铁斧头初始化成功\n");

return OK;

}

int init\_ironshovel(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

(h1->name) = "铁锹";

init\_stick(&(h1->schild));

init\_ironingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铁锹初始化成功\n");

return OK;

}

int init\_ironhoe(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

(h1->name) = "铁锄头";

init\_stick(&(h1->schild));

init\_ironingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铁锄头初始化成功\n");

return OK;

}

int init\_fishingpole(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "钓鱼竿";

init\_stick(&(h1->schild));

init\_gossamer(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("钓鱼竿初始化成功\n");

return OK;

}

int init\_copperaxe(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "铜斧头";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铜斧头初始化成功\n");

return OK;

}

int init\_coppershovel(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "铜锹";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铜锹初始化成功\n");

return OK;

}

int init\_copperhoe(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "铜锄头";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("铜锄头初始化成功\n");

return OK;

}

int init\_goldaxe(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "金斧头";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("金斧头初始化成功\n");

return OK;

}

int init\_goldshovel(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "金锹";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("金锹初始化成功\n");

return OK;

}

int init\_goldhoe(Obj\*\* p) {

Obj\* h1 = (Obj\*)malloc(sizeof(Obj));

if (h1) {

h1->name = "金锄头";

init\_stick(&(h1->schild));

init\_copperingot(&(h1->fchild));

h1->tchild = NULL;

}

\*p = h1;

printf("金锄头初始化成功\n");

return OK;

}

**整合所有函数**

int init\_total(Obj \*\*p) {

init\_ironaxe(p);

init\_ironshovel(p + 1);

init\_ironhoe(p + 2);

init\_fishingpole(p + 3);

init\_copperaxe(p + 4);

init\_coppershovel(p + 5);

init\_copperhoe(p + 6);

init\_goldaxe(p + 7);

init\_goldshovel(p + 8);

init\_goldhoe(p + 9);

return 0;

}

**定义类似于二叉树后序遍历的函数，用于查找用户输入的材料位置**(后续遍历更容易找到底层的材料)

int PostOrderTrvaverse(Obj\* T, char \*p) {

int flag = 0;//用flag表示是否找到用户输入的材料，找到为1， 没找到为零

if (T == NULL || flag == 1) {

return flag;

}

if (strcmp(p,T->name) == 0){

flag = 1;

}

if(flag != 1){

flag = PostOrderTrvaverse(T->fchild, p);

}

if (flag != 1) {

flag = PostOrderTrvaverse(T->schild, p);

}

if (flag != 1) {

flag =PostOrderTrvaverse(T->tchild, p);

}

return flag;

}

**整合后置函数**

int searchtop(char\* p, Obj\*\* list) {

int flag;

if (!list) {

exit(-1);

}

printf("\n可以最终生成。。。\n");

for (int i = 0; i < 10; i++) {

flag = PostOrderTrvaverse(\*(list + i), p);

//printf("%d", flag);

if (flag == 1) {

printf("%s\t", (\*(list + i))->name);

flag = 0;

}

}

if (flag == 0) {

printf("\n没有找到");

}

printf("\n");

return 0;

}

**用户交互**

int menu() {

printf("\n一级原材料有：铁锭 金锭 铜锭 原木 蜘蛛丝\n");

printf("二级原材料有：木板\n");

printf("三级原材料有：木棒\n");

printf("四级原材料有：铁斧头 铁锹 铁锄头 钓鱼竿 铜斧头 铜锹 铜锄头 金斧头 金锹 金锄头\n");

return 0;

}

int main() {

Obj\* totallist[10];

init\_total(totallist);

char c[10];

menu();

printf("输入需要查询的材料：");

scanf("%s", c);

searchtop(c, totallist);

return 0;

}

**运行结果**

**文本

描述已自动生成文本

描述已自动生成文本

描述已自动生成**