上海理工大学光电信息与计算机工程学院

**《数据结构》课程设计报告**

****

**专　　业 计算机科学与技术**

**学生姓名　　 王保亮**

**学　 号 1520190223**

**年　　级 2015级**

**指导教师 袁健**

**成 绩：**

**教师签字：**

报告格式要求

1. 正文字体中文为宋体，小四，行距为固定值18磅，西文为Times New Rome, 小四，行距为固定值18磅。
2. 章节标题为加粗宋体，三号，段前段后各18磅，行距为单倍行距。
3. 打印时需双面打印。

目录

[作业1 二叉树遍历及应用 4](#_Toc486246743)

[作业2 交通咨询 4](#_Toc486246744)

[作业3 模拟停车场管理 4](#_Toc486246745)

# 

# 作业1 二叉树遍历及应用

1. 任务描述
2. 根据用户从键盘输入的整型数，创建一棵二叉树，每个结点包含结点值和左右孩子的地址。
3. 按二叉树的层次输出各结点的值。
4. 利用先根遍历算法将二叉树中所有结点的左右子树交换，并按二叉树的层次输出各结点的值。
5. 任务要求
6. 实现各项功能函数。
7. main函数中要有菜单选择功能。
8. 编写完整的源程序。
9. 需有完整的测试结果。
10. 数据结构

P[MAXSIZE+1]：储存节点类型的数组。

Btnode:二叉树节点

1. 程序源代码（需对函数或主要功能语句有注释）

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 30

#define max 100

typedef int Elemtype;

typedef struct btnode{

Elemtype data;

struct btnode \*lchild;

struct btnode \*rchild;

}BTnode;

BTnode \*p[MAXSIZE+1];

BTnode \*creat(){

int i,j;

char ch;

BTnode \*t,\*s;

printf("\nenter(i,ch)until enter(0,#):\n");

printf("enter i,ch:");

scanf("%d,%c",&i,&ch);

while(i!=0&&ch!='#'){

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

s->data = ch;

s->lchild = s->rchild = NULL;

p[i]=s;

if(i==1){

t=s;

}

else{

j=i/2;

if(i%2==0){

p[j]->lchild=s;

}

else{

p[j]->rchild=s;

}

}

printf("enter i,ch:");

scanf("%d,%c",&i,&ch);

}

return t;

}

void preOrder(BTnode \*s){

printf("%c",s->data);

if(s->lchild!=NULL){

preOrder(s->lchild);

}

if(s->rchild!=NULL){

preOrder(s->rchild);

}

}

void change(BTnode \*t){

BTnode \*m;

if(t){

m=t->lchild;

t->lchild=t->rchild;

t->rchild=m;

if(t->lchild!=NULL){

change(t->lchild);

}

if(t->rchild!=NULL){

change(t->rchild);

}

}

}

void inOrder(BTnode \*s){

if(s==NULL){

return;

}

if(s->lchild!=NULL){

inOrder(s->lchild);

}

printf("%c",s->data);

if(s->rchild!=NULL){

inOrder(s->rchild);

}

}

void levelOrder(BTnode \*t){

BTnode \*p,\*q[max];

int front,rear;

front=rear=0;

rear=(rear+1)%max;

q[rear]=t;

while(front!=rear){

front=(front+1)%max;

p=q[front];

printf("%c",p->data);

if(p->lchild){

rear=(rear+1)%max;

q[rear]=p->lchild;

}

if(p->rchild){

rear=(rear+1)%max;

q[rear]=p->rchild;

}

}

}

int main(){

BTnode \*s;

s=creat();

int num;

char m;

m='Y';

while(m=='y'||m=='Y'){

printf("请输入你要执行的操作：1、按层次遍历；2、交换左右孩子后按层次遍历:");

scanf("%d",&num);

switch(num){

case 1:printf("按层次遍历:");

levelOrder(s);

break;

case 2:printf("交换左右孩子后按层次遍历:");

change(s);levelOrder(s);change(s);

break;

default:break;

}

getchar();

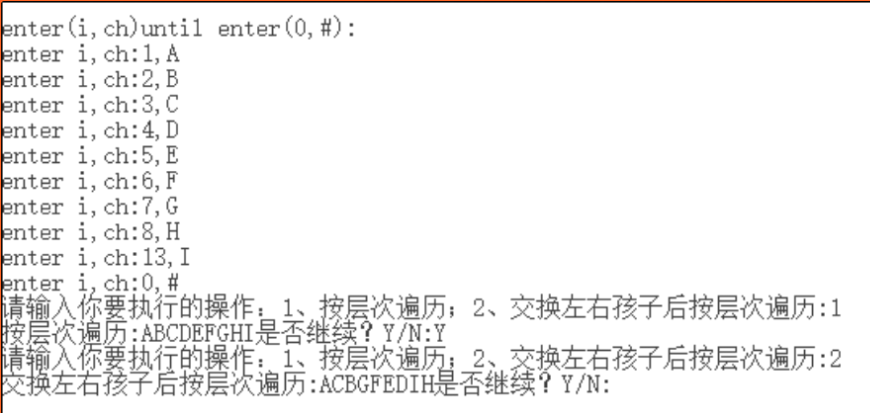
printf("是否继续？Y/N:");

scanf("%c",&m);

}

return 0;

}

1. 测试结果（建议用画图软件将黑底白字的截屏图的反色图附上）

# 作业2 交通咨询

1. 任务描述
2. 根据用户从键盘输入的城市名以及城市间的道路连接距离，创建一个简单的城市交通网络图。
3. 输出用户咨询的两个城市间的最短路径和距离。
4. 任务要求
5. 编写完整的源程序。
6. 需有完整的测试结果。
7. 数据结构

s[]：s[i]为1表示i点已经比较过；

path[i]表示i节点的前驱点；

dist[i]表示从v0到i的距离；

vex[]表示存放节点数据的数组；

arc[][]表示存放两节点的权值；

1. 程序源代码（需对函数或主要功能语句有注释）

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<string>

#define MAXVEX 100

#define MAXSIZE 100

#define MAXINT 32767

#define MAXNUM 10

using namespace std;

typedef int Edgetype;

typedef int Elemtype;

typedef string Vertextype;

typedef struct graph {

Vertextype vex[MAXVEX]; //存放节点数据

Edgetype arc[MAXVEX][MAXVEX]; //存放两节点的权值

int numNodes;

int numEdges;

}MGraph;

//输入城市名字，返回城市对应输入时的编号，方便程序调用

int name(MGraph \*Gp, string a) {

int i;

for (i = 0; i < Gp->numNodes; i++) {

if (Gp->vex[i] == a)

return i;

}

}

void CreateMGraph(MGraph \*Gp) { //图的创建

int i, j;

int m;

cout << "请输入城市数量:";

cin >> Gp->numNodes;

string a;

cout << "请输入城市名字(0-" << Gp->numNodes << "):" << endl;

for (i = 0; i<Gp->numNodes; i++) { //输入网的各节点数据

cin >> a;

Gp->vex[i] = a;

}

for (i = 0; i<Gp->numNodes; i++) { //初始化邻接矩阵

for (j = 0; j<Gp->numNodes; j++) {

Gp->arc[i][j] = MAXINT;

}

}

string x,y;

cout << "请输入两城市的名字及他们之间的距离(城市名字与距离之间均用空格隔开)(以城市名字为#结束）:"<<endl;

//通过输入城市名字和对应权值，修改邻接矩阵中的数据

while (x != "#") {

cin >> x >> y >> m;

Gp->arc[name(Gp,x)][name(Gp, y)] = m;

}

}

//dijksrra算法，

void dijkstra(MGraph \*Gp, int v0, int path[], int dist[]) {

int i, j, k, mindist;

int u;

int s[MAXNUM]; //s[i]=1表示i已经选过

//初始 s，path,dist；path[i]表示i的前驱点，dist[i]表示从起始点到i点的权值

for (i = 0; i<Gp->numNodes; i++) {

s[i] = 0;

dist[i] = Gp->arc[v0][i];

if (i != v0&&dist[i]<MAXINT) {

path[i] = v0;

}

else

path[i] = -1;

}

dist[v0] = 0; //起始点到本身的权值为0

s[v0] = 1; //先把起始点取出

u = v0; //u是中间变量，始终存储权值最小的点

for (i = 0; i<Gp->numNodes - 1; i++) {

mindist = MAXINT;

for (j = 0; j<Gp->numNodes; j++) {

if (!s[j] && dist[j]<mindist) {

u = j;

mindist = dist[j];

}

}

s[u] = 1;

/\*如果从起始点到i点距离大于从起始

点到中间点加上从中间点到i的距离，

那就更新从起始点到i的权值\*/

for (k = 0; k<Gp->numNodes; k++) {

if (!s[k] && Gp->arc[u][k]<MAXINT && (dist[k]>mindist + Gp->arc[u][k])) {

dist[k] = mindist + Gp->arc[u][k];

path[k] = u;

}

}

}

}

//输出要求的量城市的最短路径及最短距离

void display(MGraph \*Gp,int v0,int v1,int path[], int dist[]) {

int i, next;

for (i = 0; i<Gp->numNodes; i++) {

if(i==v1){

if (dist[i] < MAXINT&&i != v0) {

cout << Gp->vex[i] << "<--";

next = path[i];

while (next != v0) {

cout << Gp->vex[next] << "<--";

next = path[next];

}

cout << Gp->vex[v0] << ":" << dist[i] << endl;

}

else

if (i != v0) cout << Gp->vex[i] << "<--" << Gp->vex[v0] << ":no path" << endl;

}

}

}

int main() {

int path[MAXNUM];

int dist[MAXNUM];

string a;

MGraph G;

string b;

CreateMGraph(&G);

char c='y';

while(c=='y'||c=='Y'){

cout << "请输入你要查询的两个城市:";

cin >> a>>b;

dijkstra(&G, name(&G,a), path, dist);

cout << a << "到其他城市的最短距离的路线分别为:\n";

display(&G, name(&G,a),name(&G,b), path, dist);

getchar();

cout<<"是否继续查询？Y/N:";

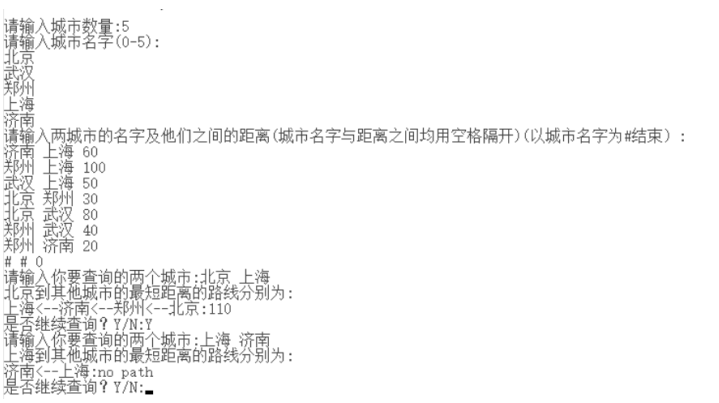
cin>>c;

}

system("pause");

return 0;

}

1. 测试结果（建议用画图软件将黑底白字的截屏图的反色图附上）

# 作业3 模拟停车场管理

1. 任务描述

设有一个可以停放n辆汽车的狭长停车场，它只有一个大门可以供车辆进出。车辆按到达停车场时间的先后次序从停车场最里面向门口处停放（最先到达的车停在最里面）。请编程模拟停车场管理。

1. 如果停车场已放满n辆车，则后面的车辆只能在停车场大门外的便道上等待。
2. 如果停车场内有车开走，则排在便道上的第一辆车就可进入停车场。
3. 停车场内如有某辆车要开走，在它之后进入停车场的车都必须先退出停车场到旁边的一个临时停车区暂停为它让路。待其开出停车场后，这些车辆再依原来的次序进入。
4. 每辆车进入停车场时记录车号和进入时间，当车离开停车场时需按停车时间收费（按停的小时数收费，不满1小时的不收费）。
5. 停在便道上的车离开不收费。
6. 任务要求
7. 实现各项功能函数。
8. main函数中要有菜单选择功能。包括显示停车场状态；车辆到达；车辆离开；退出。
9. 编写完整的源程序。
10. 需有完整的测试结果。
11. 数据结构

Stack s1 栈1，表示车库

Stack s2 栈2，表示临时停车区

LinkQueue lq 链队，表示便道

Struct Car 表示汽车的结构体，存放汽车的信息

1. 程序源代码（需对函数或主要功能语句有注释）

#include<iostream>

#include<stdlib.h>

#include<string.h>

#include<string>

#define size 3

#define MAX 10000

using namespace std;

typedef struct Car { //存储汽车信息的结构体，包括车牌号和驶入时间

char number[10];

int firtime;

}Car;

struct node {

Car c;

struct node \*next;

};

class LinkQueue {

private:

int len;

struct node \*front;

struct node \*rear;

public:

LinkQueue(struct node \*p) { //链队列

len = 0;

p->next = NULL;

front = rear = p;

}

~LinkQueue() {}

void EnQueue(Car xc) { //入队

struct node \*p = (struct node \*)malloc(sizeof(struct node));

p->c = xc;

rear->next = p;

rear = rear->next;

len++;

rear->next = NULL;

}

Car DelQueue() {// 带头节点的链队

Car xc;

xc = front->next->c;

front = front->next;

len--;

return xc;

}

int IsEmpty() {

if (len == 0) {

return 1;

}

return 0;

}

void show() { //显示便道中车辆的车牌号

struct node \*p;

p = front->next;

string a;

if(!IsEmpty()){

cout << "便道中车辆车牌号从头到尾为:" << endl;

while (p != NULL) {

a = p->c.number;

cout << a << endl;

p = p->next;

}

}

else{

cout<<"便道为空！\n";

}

}

};

class Stack { //栈

public:

Stack() {

top = -1;

}

bool push(Car c);

Car pop();

void clear();

bool isEmpty();

bool isFull();

void print();

~Stack() {}

private:

Car data[size];

int top;

};

bool Stack::push(Car c) {

if (top == size - 1) {

return false;

}

top++;

data[top] = c;

return true;

}

Car Stack::pop() { //返回值为Car类型，可以直接作为临时停车区的栈的入栈函数的参数；

Car c;

c = data[top--];

return c;

}

void Stack::clear() {

top = -1;

}

bool Stack::isEmpty() {

if (top == -1) {

return true;

}

return false;

}

bool Stack::isFull() {

if (top == size - 1) {

return true;

}

return false;

}

void Stack::print() {//显示车库中车辆的车牌号

int i;

Car c;

if (!isEmpty()) {

cout << "车库中从前到后车辆的车牌号为:" << endl;

for (i = top; i >= 0; i--) {

c = data[i];

cout << c.number << endl;

}

}

else {

cout << "车库为空！" << endl;

}

}

int main() {

int a, b;

char c;

c = 'y';

Stack s1, s2;

char k;

k = 'y';

int z;

struct node \*p = (struct node \*)malloc(sizeof(struct node));

LinkQueue \*lq = new LinkQueue(p);

char x[10];

while (k == 'Y' || k == 'y') {

cout << " ★★★★★★★欢迎使用车库管理系统★★★★★★" << endl;

cout << " ★ ★" << endl;

cout << " ★ \*\*\*\*1,显示停车场状态\*\*\*\* ★" << endl;

cout << " ★ \*\*\*\*2,车辆到达 \*\*\*\* ★" << endl;

cout << " ★ \*\*\*\*3,车辆离开 \*\*\*\* ★" << endl;

cout << " ★ \*\*\*\*4,退出 \*\*\*\* ★" << endl;

cout << " ★ ★" << endl;

cout << " ★★★★★★★★★★★★★★★★★★★★★★★" << endl;

cout << "请输入你要进行的操作:";

cin >> a;

system("cls");

switch (a) {

case 1:

s1.print();

lq->show();

break;

case 2:

int i;

Car c1;

for (i = 0; i<MAX; i++) {

if (!s1.isFull()) {

cout << "车库未满，请驶入车库！" << endl;

cout << "请输入你的车牌号:";

cin >> x;

strcpy(c1.number, x);

cout << "请输入你进入时的时间:";

cin >> b;

c1.firtime = b;

s1.push(c1);

z++;

cout<<"现在车库里面有"<<z<<"辆车!"<<endl;

}

else {

cout << "车库已满，请驶入便道！" << endl;

cout << "请输入你的车牌号:";

cin >> x;

strcpy(c1.number, x);

cout << "请输入你进入时的时间:";

cin >> b;

c1.firtime = b;

lq->EnQueue(c1);

}

cout << "是否继续？Y/N:";

cin >> c;

if (c != 'Y'&&c != 'y')

break;

}

break;

case 3:

{

int i;

char g='y';

Car c;

int leavetime;

while(g=='y'||g=='Y'){

if (!s1.isEmpty()) {

int rander = rand() % 2;

if (!lq->IsEmpty()) {

if (rander == 1) {

int rander2 = rand() % z + 1;

cout << "车库中第" << rander2 << "辆车出库！" << endl;

z--;

Car car1;

for (i = 1; i < rander2; i++) {

s2.push(s1.pop());

}

car1 = s1.pop();

for (i = 1; i < rander2; i++) {

s1.push(s2.pop());

}

cout << "请输入您的离开时间:";

cin >> leavetime;

if (leavetime - car1.firtime < 100) {

cout << "您使用车库的时间小于一小时，不收取费用！" << endl;

}

else {

cout << "请缴纳您使用车库的费用，费用为:" << ((leavetime - car1.firtime) / 100) \* 10 << endl;

}

if (!lq->IsEmpty()) {

cout << "便道上的第一辆车入库！" << endl;

c = lq->DelQueue();

c.firtime = leavetime + 1;

s1.push(c);

}

}

else {

cout << "便道上汽车离开，不收取费用！" << endl;

lq->DelQueue();

}

}

else {

int rander2 = rand() % z + 1;

cout << "车库中第" << rander2 << "辆车出库！" << endl;

z--;

Car car1;

for (i = 1; i < rander2; i++) {

s2.push(s1.pop());

}

car1 = s1.pop();

for (i = 1; i < rander2; i++) {

s1.push(s2.pop());

}

cout << "请输入您的离开时间:";

cin >> leavetime;

if (leavetime - car1.firtime < 100) {

cout << "您使用车库的时间小于一小时，不收取费用！" << endl;

}

else {

cout << "请缴纳您使用车库的费用，费用为:" << ((leavetime - car1.firtime) / 100) \* 10<<endl;

}

}

cout << "是否继续？Y/N:";

cin >> g;

}

else {

cout << "车库已空！" << endl;break;

}

}

}

break;

case 4:

cout<<"谢谢使用车库管理系统！"<<endl;

break;

}

getchar();

cout << "是否继续查看？Y/N:";

cin >> k;

system("cls");

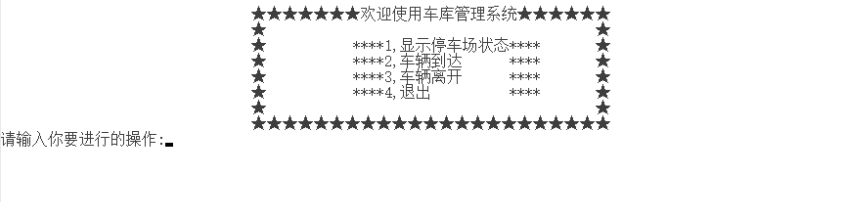
}

system("pause");

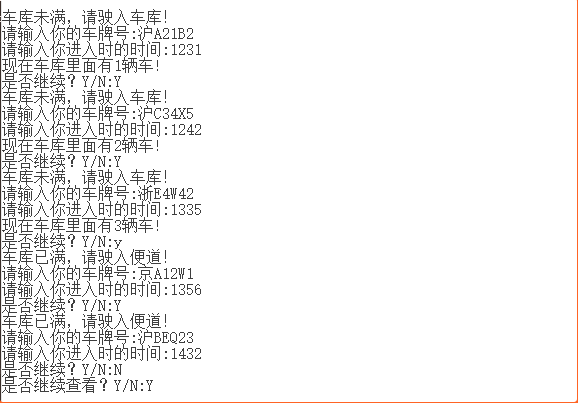
return 0;

}

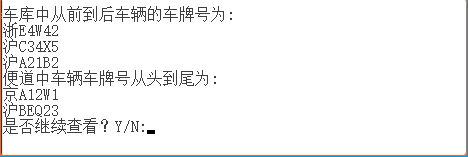
1. 测试结果（建议用画图软件将黑底白字的截屏图的反色图附上）

程序界面：

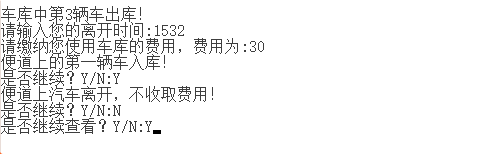
汽车入库：



查看状态：



车辆离开：



退出：

