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　　 孙晓宇**

**学　 号 1517440223**

**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 1](#_Toc486246743)

[作业2 交通咨询 5](#_Toc486246744)

[作业3 模拟停车场管理 10](#_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

二叉树的根指针binode \*bt

结点队列 binode \*q[MAX]，Q[MAX]

typedef struct node {

char value; //结点的值

struct node \*lchild,\*rchild; //指向左右孩子结点的指针

}binode, \*bitree;

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>

#include<malloc.h>

#define MAX 20

typedef struct node {

char value;

struct node \*lchild,\*rchild;

}binode, \*bitree;

binode \*create\_bt() { //建立二叉树

char pos,ch;

binode \*Q[MAX];

int front, rear;

binode \*root,\*s;

root = NULL;

front = 1;rear = 0;

scanf\_s("%c", &pos,sizeof(pos));

scanf\_s("%c", &ch,sizeof(ch));

while (pos != ' ')

{ rear++;

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

s->value = ch;

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

Q[rear] = s;

if(rear == 1) root = s;

else {if (rear % 2 == 0)

Q[front]->lchild = s;

else{Q[front]->rchild = s;

front++;}}

getchar();

scanf\_s("%c", &pos,sizeof(pos));

scanf\_s("%c", &ch, sizeof(ch));

}

return root;

}

//按层次遍历二叉树

void treelevel(binode \* bt)

{

printf("按层次遍历二叉树：");

binode \*q[MAX], \*p;

int front, rear;

front = rear = 0;

if (!bt) return;

q[rear] = bt; //树根进队

rear = (rear + 1) % MAX;

while (front!=rear) //队列不为空

{ p = q[front]; //出队

front = (front + 1) % MAX;

printf("%C ", p->value);

if (p->lchild) //左子树不为空

{ q[rear] = p->lchild;

rear = (rear + 1) % MAX;

}

if (p->rchild) //右子树不为空

{ q[rear] = p->rchild;

rear = (rear + 1) % MAX;}

}

printf("\n");

}

//交换左右子树

void exchange(binode \*bt)

{

binode \*tmp;

if (bt)

{ tmp = bt->lchild;

bt->lchild = bt->rchild;

bt->rchild = tmp;

exchange(bt->lchild);

exchange(bt->rchild);}

}

void main() {

binode \*bt = NULL;

int key;

printf("\t1-建立二叉树\n");

printf("\t2-按二叉树层次输出各结点的值\n");

printf("\t3-利用先根遍历算法将二叉树中所有结点的左右子树交换\n");

printf("\t0-退出\n");

printf("请选择：");

scanf\_s("%d", &key);

while (key != 0) {

switch (key)

{

case 1:printf("输入建树元素<char>:\n");getchar();bt = create\_bt();break;

case 2:treelevel(bt);break;

case 3:exchange(bt);printf("左右子树交换后");treelevel(bt);break;

default:break;

}

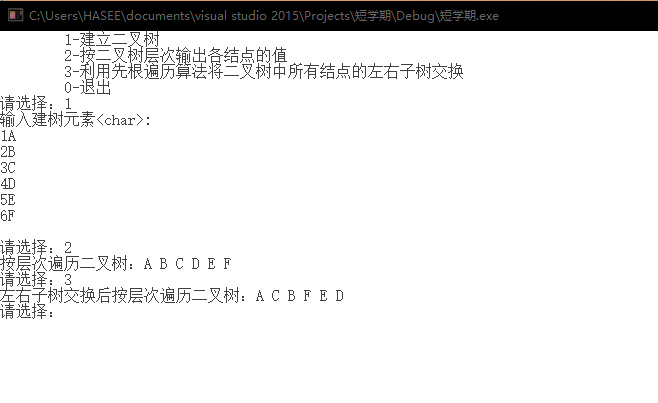
printf("请选择：");

scanf\_s("%d", &key);}

system("pause");

}

1. 测试结果



# 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

邻接矩阵结构体指针Mgraph \*G

typedef struct {

char\* cities[MAX]; //城市的名字

int edges[MAX][MAX]; //城市到城市之间的距离

int n, e; //城市的数量n，道路的数量e

}Mgraph;

布尔类型数组 S[MAX] //判断是否已存入该点到S集合中

数组int dist[MAX] //存储从出发点到该点的最短距离

数组int prev[MAX] //存储以出发点到该点的最短距离的前点的最短距离

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>

#include<string.h>

#define MAX 10

#define MAXINT 32767

char tmp[MAX][20];

typedef struct {

char\* cities[MAX];

int edges[MAX][MAX];

int n, e;

}Mgraph; //定义图的邻接矩阵储存类型

void CreateGraph(Mgraph \*G)//邻接矩阵将图存储

{ int x = 0;

printf("输入交通图的城市数n，道路数e：");

scanf\_s("%d %d", &G->n, &G->e);

for (int i = 0;i<G->n;i++) //邻接矩阵置0,得到所有城市的名称

{ printf("输入城市%d的名称：",i+1);

scanf\_s("%s", tmp[x], sizeof(tmp[x]));

G->cities[x]= tmp[x];

x++;

for (int j = 0;j<G->n;j++)

G->edges[i][j] = MAXINT;

}

for (int k = 0;k<G->e;k++)

{ char tmp[20];

int i, j, d;

printf("输入两城市及之间的距离\n");

printf("输入城市1的名称：");

scanf\_s("%s", tmp,sizeof(tmp));//输入城市1名称

for (int y = 0;y < G->n;y++)

{ if (strcmp(G->cities[y], tmp) == 0)

i = y;

}

printf("输入城市2的名称：");

scanf\_s("%s", tmp,sizeof(tmp));//输入城市2名称

for (int y = 0;y < G->n;y++)

{ if (strcmp(G->cities[y], tmp) == 0)

j = y;

}

printf("输入两城市之间的距离：");

scanf\_s("%d", &d);//两城市之间的距离

G->edges[i][j] = d;

G->edges[j][i] = d;

}

}

void pre(int v, int x, Mgraph \*G, int prev[])

{ if (x != v)

{ pre(v, prev[x], G, prev);

printf("%s->", G->cities[x]);

}

else printf("%s->", G->cities[v]);

}

void Dijkstra(Mgraph \*G)

{

int dist[MAX];

int prev[MAX];

int v;

bool S[MAX]; // 判断是否已存入该点到S集合中

char tmp[20];

printf("输入城市1的名称：");

scanf\_s("%s", tmp, sizeof(tmp));//输入城市1名称

for (int i = 0; i < G->n; i++)

{ if (strcmp(G->cities[i],tmp) == 0)

v = i;

}

for (int i = 0; i < G->n; i++)

{ if (v != i)

{ dist[i] = G->edges[v][i];

S[i] = false; // 初始都未用过该点

if (dist[i] == MAXINT) prev[i] = -1;

else prev[i] = v;}

}

dist[v] = 0;

S[v] = true;

for (int i = 1; i < G->n; i++)

{ int mindist = MAXINT;

int u = v; // 找出当前未使用的点j的dist[j]最小值

for (int j = 0; j < G->n; j++)

if ((!S[j]) && dist[j]<mindist)

{ u = j; // u保存当前邻接点中距离最小的点的号码

mindist = dist[j];

}

S[u] = true;

for (int j = 0; j < G->n; j++)

if ((!S[j]) && G->edges[u][j]<MAXINT)

{ f (dist[u] + G->edges[u][j] < dist[j]) //在通过新加入的u点路径找到离v点更短的路径

{ dist[j] = dist[u] + G->edges[u][j]; //更新dist

prev[j] = u; //记录前驱顶点

}}

}

printf("输入城市2的名称：");

scanf\_s("%s", tmp, sizeof(tmp));//输入城市2名称

for (int i = 0; i < G->n; i++)

{ if (strcmp(G->cities[i], tmp) == 0)

{ printf("两城市之间的最短距离为：");

printf("%d\n",dist[i]);

printf("两城市之间的最短路径为：");

pre(v,prev[i],G,prev);

printf("%s", G->cities[i]);

}}

}

void main()

{ Mgraph G;

CreateGraph(&G);

printf("交通图创建完成\n\n");

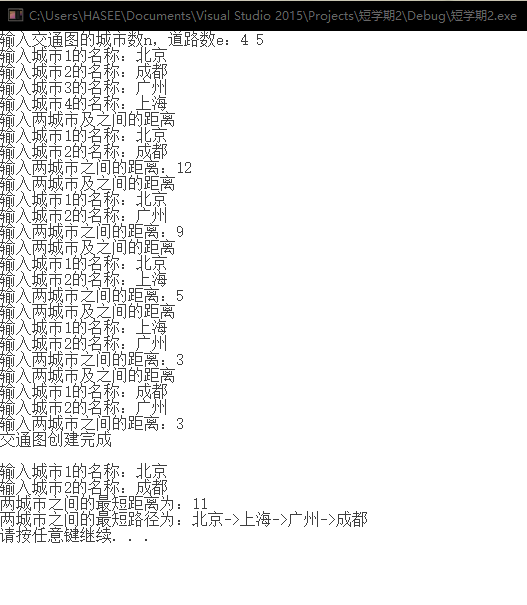
Dijkstra(&G);

printf("\n");

system("pause");

}

1. 测试结果



# 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

车结点堆栈LinkStack top,wtop;

typedef struct Snode {

char id[20]; //车牌号

char time[10]; //车进入停车场的时间，默认为NULL

char date[10]; //车进入停车场的日期，默认为NULL

struct Snode \*next;

}stackNode, \*LinkStack;

链式队列LinkQueue Q

typedef struct {

Qnode \*front; //指向队列的头的指针

Qnode \*rear; //指向队列的尾的指针

}LinkQueue;

//把链式队列的头尾节点指针定义为一个单独的结构体，避免在新增节点时占用过多的空间

队列的结点

typedef struct node

{

char id[20]; //便道上的车的车牌号

char time[10]; //车进入停车场时，将上一辆车离开的时间存入，并将数据分享给新的栈结点

char date[10]; //车进入停车场时，将上一辆车离开的日期存入，并将数据分享给新的栈结点

struct node \*next;

}Qnode, \*q\_node;

1. 程序源代码

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#include<string>

int count = 0;

typedef struct Snode {

char id[20];

char time[10];

char date[10];

struct Snode \*next;

}stackNode, \*LinkStack;

typedef struct node

{

char id[20];

char time[10];

char date[10];

struct node \*next;

}Qnode, \*q\_node;

//把链式队列的头尾节点指针定义为一个单独的结构体，避免在新增节点时占用过多的空间

typedef struct {

Qnode \*front;

Qnode \*rear;

}LinkQueue;

//初始化链式队列

void InitQueue(LinkQueue \*Q)

{ Q->front = Q->rear = (Qnode \*)malloc(sizeof(Qnode));

if (!Q->front)

{ printf("error!");

return;

}

Q->rear->next = NULL;//将链式队列的尾节点始终指向NULL

}

void InQueue(LinkQueue \*Q)//入队

{ Qnode \*p;

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

printf("请输入车号：");

scanf\_s("%s", p->id, sizeof(p->id));

p->next = NULL;

Q->rear->next = p;

Q->rear = p;

}

bool isEmptyQ(LinkQueue \*Q)//判断队列是否为空

{ if (Q->front->next == NULL)

return true;

else return false;

}

void PrintQueue(LinkQueue \*Q)//输出队

{ Qnode \*p;

p = Q->front->next;

if (!p)

{ printf("便道为空！\n");

return;

}

else

{ printf("便道的状态由里至外为:\n");

while (p)

{ printf("车号:");

printf("%s\n", p->id);

p = p->next;

} }

}

void InitStack(LinkStack &topPt) //初始化栈

{

topPt = NULL;

}

int isEmpty(LinkStack top) //判空栈

{ return top == NULL;

}

void in\_push(LinkStack &topPt, stackNode \*p)

{ p->next = topPt;

topPt = p;

}

void push(LinkStack &topPt, LinkQueue \*Q,int n) //进栈

{

if (count >= n)

{ printf("停车场已满，请在便道上等待！\n");

InQueue(Q);

return;

}

stackNode \*p;

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

printf("请输入车号：");

scanf\_s("%s", p->id, sizeof(p->id));

printf("请输入进入时间：");

scanf\_s("%s", p->time, sizeof(p->time));

printf("请输入进入日期：");

scanf\_s("%s", p->date, sizeof(p->date));

in\_push(topPt, p);

count++;

}

stackNode\* pop(LinkStack &topPt) //出栈

{ stackNode \*p;

if (isEmpty(topPt))

return NULL;

p = topPt;

topPt = p->next;

return p;

}

void OutStack(LinkStack &topPt,LinkStack &wtopPt, LinkQueue \*Q,int fee)//离开停车场

{

if (isEmpty(topPt))

{ printf("停车场为空！\n");

return;

}

int arrive\_t[2], leave\_t[2];

int arrive\_d[2], leave\_d[2];

bool isFind = false;

bool isFindQ = true;

char tmp[20];

char time[20];

char date[20];

char time\_t[20];

char date\_t[20];

char arrive\_time[20];

char arrive\_date[20];

printf("请输入离开停车场的车牌号：");

scanf\_s("%s", &tmp,sizeof(tmp));

printf("请输入离开时间：");

scanf\_s("%s", &time, sizeof(time));

printf("请输入离开日期：");

scanf\_s("%s", &date, sizeof(date));

strcpy\_s(time\_t, time);

strcpy\_s(date\_t, date);

Qnode \*p, \*q;

p = Q->front;

q = NULL;

while (p->next)

{ q = p;

q = q->next;

if (strcmp(q->id, tmp) == 0)

{ p->next = q->next;

free(q);

printf("从便道上离开，不收费！\n");

isFind = true;

return;

}

p = p->next;

}

while (!isEmpty(topPt)&&(strcmp(topPt->id, tmp) != 0))

in\_push(wtopPt, pop(topPt));

if (isEmpty(topPt))

isFindQ = false;

if (!isFind&&!isFindQ)

{ printf("未找到该车!\n");

return;

}

strcpy\_s(arrive\_time,topPt->time);

strcpy\_s(arrive\_date,topPt->date);

pop(topPt);

count--;

while (!isEmpty(wtopPt))

in\_push(topPt, pop(wtopPt));

//时间拆分转化为数值

int x = 0;

char delims[] = ":";

char delimst[] = "/";

char \*buf;

char \*result = NULL;

result = strtok\_s(arrive\_time, delims, &buf);

while (result != NULL) {

arrive\_t[x++] = atoi(result);

result = strtok\_s(NULL, delims, &buf);

}

x = 0;

result = strtok\_s(time, delims, &buf);

while (result != NULL) {

leave\_t[x++] = atoi(result);

result = strtok\_s(NULL, delims, &buf);

}

x = 0;

result = strtok\_s(arrive\_date, delimst, &buf);

while (result != NULL) {

arrive\_d[x++] = atoi(result);

result = strtok\_s(NULL, delimst, &buf);

}

x = 0;

result = strtok\_s(date, delimst, &buf);

while (result != NULL) {

leave\_d[x++] = atoi(result);

result = strtok\_s(NULL, delimst, &buf);

}

int t = leave\_t[0] - arrive\_t[0];

if (leave\_t[1] - arrive\_t[1] < 0)

t--;

if (t < 0 && leave\_d[1] - arrive\_d[1] <= 0)

{ printf("error!");

return;

}

else

{ t += 24 \* (leave\_d[1] - arrive\_d[1]);

printf("计费停车时间为%d小时，费用为%d元\n", t, fee \* t);

}

if (!isEmptyQ(Q))

{ printf("便道上有等待车辆，便道上的车辆进入停车场，进入时间为%s,进入日期为:%s\n", time\_t,date\_t);

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

Qnode \*qq = Q->front->next;

strcpy\_s(p->id, qq->id);

strcpy\_s(p->time, time\_t);

strcpy\_s(p->date, date\_t);

in\_push(topPt, p);

count++;

Q->front = Q->front->next;

}

}

void showStack(LinkStack &topPt, LinkStack &wtopPt)//显示停车场与便道状态

{ if (isEmpty(topPt))

{ printf("停车场为空！\n");

return;

}

while(!isEmpty(topPt))

in\_push(wtopPt, pop(topPt));

stackNode \* p = wtopPt;

printf("停车场的状态由里至外为:\n");

while (p)

{ printf("车号:");

printf("%s\t", p->id);

printf("进入时间:");

printf("%s\t", p->time);

printf("进入日期:");

printf("%s\n", p->date);

p = p->next;

}

while (!isEmpty(wtopPt))

in\_push(topPt, pop(wtopPt));

}

void main()

{

LinkQueue Q;

LinkStack top,wtop;

InitQueue(&Q);

InitStack(top);

InitStack(wtop);

int n,x,fee;

printf("请输入停车场可以停放汽车的数量n：");

scanf\_s("%d", &n);

printf("请输入停车场每小时的费用fee：");

scanf\_s("%d", &fee);

printf("\n\t1-显示停车场状态\n");

printf("\t2-车辆到达\n");

printf("\t3-车辆离开\n");

printf("\t0-退出\n\n");

printf("请选择：");

scanf\_s("%d", &x);

while (x != 0)

{

switch (x)

{

case 1:showStack(top, wtop);PrintQueue(&Q);break;

case 2:push(top,&Q,n);break;

case 3:OutStack(top,wtop,&Q,fee);break;

default:break;

}

printf("请选择：");

scanf\_s("%d", &x);

}

system("pause");

}

1. 测试结果

