# 实验一 线性表参考答案

参考程序1

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

//\*PROGRAM :顺序结构的线性表 \*

//\*CONTENT :建立,插入,删除,查找 \*

//\*编程语言： Visual c++ 6.0 \*

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 20

typedef int ElemType; //数据元素的类型

typedef struct

{

ElemType a[MAXSIZE];

int length;

}SqList; //顺序存储的结构体类型

SqList a,b,c;

//函数声明

void creat\_list(SqList \*L);

void out\_list(SqList L);

void insert\_sq(SqList \*L,int i,ElemType e);

ElemType delete\_sq(SqList \*L,int i);

int locat\_sq(SqList L,ElemType e);

//主函数

void main()

{

int i,k,loc;

ElemType e,x;

char ch;

do {printf("\n\n\n");

printf("\n 1.建立线性表");

printf("\n 2.插入元素");

printf("\n 3.删除元素");

printf("\n 4.查找元素");

printf("\n 0.结束程序运行");

printf("\n =====================");

printf("\n 请输入要执行的操作: ");

scanf("%d",&k);

switch(k)

{

case 1:{creat\_list(&a);

out\_list(a);

}

break;

case 2:{printf("\n请输入插入位置： ",a.length+1);

scanf("%d",&i);

printf("请输入要插入的元素值： ");

scanf("%d",&e);

insert\_sq(&a,i,e);

out\_list(a);

}

break;

case 3:{printf("\n请输入要删除元素的位置： ",a.length);

scanf("%d",&i);

x=delete\_sq(&a,i);

out\_list(a);

if(x!=-1)

printf("\n删除的元素为：%d\n",x);

else printf("要删除的元素不存在！");

}

break;

case 4:{printf("\n请输入要查找的元素值：");

scanf("%d",&e);

loc=locat\_sq(a,e);

if(loc==-1)

printf("\n未找到指定元素！");

else

printf("\n已找到，元素的位置是: %d ",loc);

}

break;

}/\*switch\*/

}

while(k!=0);

printf("\n 按回车键，返回...\n");

ch=getchar();

}/\*main\*/

//建立线性表

void creat\_list(SqList \*L)

{

int i;

printf("请输入线性表的长度: ");

scanf("%d",&L->length);

for(i=0;i<L->length;i++)

{

printf("数据 %d =",i);

scanf("%d",&(L->a[i]));

}

}

//输出线性表

void out\_list(SqList L)

{

int i;

for(i=0;i<=L.length-1;i++)

printf("%10d",L.a[i]);

}

//在线性表的第i个位置插入元素e

void insert\_sq(SqList \*L,int i,ElemType e)

{

int j;

if(L->length==MAXSIZE)

printf("线性表已满!\n");

else {

if(i<1||i>L->length+1)

printf("输入位置错!\n");

else {

for(j=L->length-1;j>=i-1;j--)

L->a[j+1]=L->a[j];

L->a[i-1]=e;

L->length++;

}

}

}

//删除第i个元素，返回其值

ElemType delete\_sq(SqList \*L,int i)

{

ElemType x;

int j;

if(L->length==0)

printf("空表!\n");

else if(i<1||i>L->length)

{

printf("输入位置错！\n");

x=-1;

}

else

{

x=L->a[i-1];

for(j=i;j<=L->length-1;j++)

L->a[j-1]=L->a[j];

L->length--;

}

return(x);

}

//查找值为e的元素，返回它的位置

int locat\_sq(SqList L,ElemType e)

{

int i=0;

while(i<=L.length-1&&L.a[i]!=e) i++;

if(i<=L.length-1)

return(i+1);

}

参考程序2：

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

//\*PROGRAM :链式结构的线性表 \*

//\*CONTENT :生成,插入,删除,定位,查找 \*

//\*编程语言： Visual c++ 6.0 \*

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

#include <conio.h>

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#define LEN sizeof(LNode) //定义LEN为一个//节点的长度

enum BOOL{False,True}; //定义BOOL型

typedef struct node

{char data; //数据域

struct node \*next;//指向下一个节点的指针

}LNode,\*LinkList;

void CreatList(LinkList &,int); //生成一个//单链表

BOOL ListInsert(LinkList &,int,char); //在单//链表中插入一个元素

BOOL ListDelete(LinkList &,int,char &); //在//单链表中删除一个元素

BOOL ListFind\_keyword(LinkList,char,int &); //按关键字查找一个元素

BOOL ListFind\_order(LinkList,char &,int); //按序号查找一个元素

void ListPrint(LinkList); //显示单链表//所有元素

void main()

{LinkList L;

BOOL temp;

int num,loc,flag=1;

char j,ch;

//---------------------程序解说-----------------------

printf("本程序实现链式结构的线性表的操作。\n");

printf("可以进行插入，删除，定位，查找等操作。\n");

//----------------------------------------------------

printf("请输入初始时链表长度:"); //输入生成单链表时的元素个数

scanf("%d",&num);

CreatList(L,num); //生成单链表

ListPrint(L);

while(flag)

{ printf("请选择:\n");

printf("1.显示所有元素\n"); //显示链//表元素

printf("2.插入一个元素\n"); //插入链//表元素

printf("3.删除一个元素\n"); //删除链//表元素

printf("4.按关键字查找元素\n"); //按//关键字查找

printf("5.按序号查找元素\n"); //按序号//查找

printf("6.退出程序 \n"); //退出

scanf(" %c",&j);

switch(j)

{case '1':ListPrint(L); break;

case '2':{printf("请输入元素(一个字符)和要插入的位置:\n");

printf("格式:字符，位置；例如:a,3\n");

scanf(" %c,%d",&ch,&loc); //输入要插入的元素和要插入的位置

temp=ListInsert(L,loc,ch); //插入

if(temp==False) printf("插入失败!\n"); //插入失败

else printf("插入成功!\n"); //成//功插入

ListPrint(L);

break;

}

case '3':printf("请输入要删除的元素所在位置:");

scanf("%d",&loc); //输入要删除的节点的位置

temp=ListDelete(L,loc,ch); //删除

if(temp==False) printf("删除失败!\n"); //删除失败

else printf("成功删除了一个元素:%c\n",ch); //删除成功，显示该元素

ListPrint(L);

break;

case '4':if(L->next==NULL) //链表为空

printf("链表为空!\n");

else{printf("请输入要查找的元素(一个字符):");

scanf(" %c",&ch); //输入要查找的元素

temp=ListFind\_keyword(L,ch,loc); //按关键字//查找

if(temp==False) printf("没有找到该元素!\n"); //查找失败

else printf("该元素在链表的第%d个位置。\n",loc);

//成功查找，显示该元素位置

}

break;

case '5':if(L->next==NULL) //链表为空

printf("链表为空!\n");

else{printf("请输入要查找的位置:");

scanf("%d",&loc); //输入//要查找的元素的位置

temp=ListFind\_order(L,ch,loc); //按序号查找

if(temp==False) printf("该位置不存在!\n"); //查找失败

else printf("第%d个元素是：%c\n",loc,ch);

//成功查找，显示该元素

}

break;

default:flag=0;printf("程序结束，按任意键退出!\n");

}

}

getch();

}

void CreatList(LinkList &v,int n)

{//生成一个带头结点的有n个元素的单链表

int i;

LinkList p;

v=(LinkList)malloc(LEN); //生成头结点

v->next=NULL;

printf("请输入%d个字符：例如：abcdefg\n",n);

getchar();

for(i=n;i>0;--i)

{p=(LinkList)malloc(LEN); //生成新结点

scanf("%c",&p->data);

p->next=v->next;

v->next=p;

}

}

BOOL ListInsert(LinkList &v,int i,char e)

{//在单链表的第i各位置插入元素e，成功返回//True，失败返回False

LinkList p,s;

int j=0;

p=v;

while(p&&j<i-1) {p=p->next;++j;} //查找第//i-1个元素的位置

if(!p||j>i-1) return False; //没有找到

s=(LinkList)malloc(LEN); //生成一个新//结点

s->data=e;

s->next=p->next;//将新结点插入到单链表中，即修改指针，完成插入操作

p->next=s;

return True;

}

BOOL ListDelete(LinkList &v,int i,char &e)

{//在单链表中删除第i个元素，成功删除返回//True，并用e返回该元素值，失败返回False

LinkList p,q;

int j=0;

p=v;

while(p->next&&j<i-1) //查找第i-1个元素位//置

{p=p->next;++j;}

if(!(p->next)||j>i-1) return False; //查找//失败

q=p->next;

p->next=q->next; //删除该元素

e=q->data;//e取得该元素值，即修改指针，删除结点q

free(q); //释放该元素空间

return True;

}

BOOL ListFind\_keyword(LinkList v,char e,int &i)

{//在单链表中查找关键字为e的元素，成功返回//True,并用i返回该元素位置，

//失败返回False

i=1;

LinkList p;

p=v->next;

while((p->data!=e)&&(p->next!=NULL))//p指//针指向下一个，直到

{p=p->next; i++;} //找到或到链表尾为止

if(p->data!=e) //该元//素在链表中不存在

return False;

else return True;

}

BOOL ListFind\_order(LinkList v,char &e,int i)

{//在单链表中查找第i个元素，成功返回True，//并用e返回该元素值，

//失败返回False

LinkList p;

int j=0;

p=v;

while(p->next&&j<i) //移动指针，直到找到第i个元素

{p=p->next;++j;}

if(j!=i) return False; //查找失败

else {e=p->data; //查找成功，用e取得//该元素值

return True;

}

}

void ListPrint(LinkList v)

{//显示链表所有元素

LinkList q;

q=v->next;

printf("链表所有元素:");

while(q!=NULL)

{printf("%c ",q->data);q=q->next;}

printf("\n");

}

# 实验二 栈参考答案

参考程序2.1

#include<stdio.h>

#include<stdlib.h>

#define MAXNUM 20

#define ElemType int

typedef struct

{

ElemType stack[MAXNUM];

int top;

}SqStack;

void InitStack(SqStack \*p)

{

if(!p)

printf("内存分配失败！");

p->top=-1;

}

void Push(SqStack \*p,ElemType x)

{

if(p->top<MAXNUM-1)

{

p->top=p->top+1;

p->stack[p->top]=x;

}

else

printf("Overflow!\n");

}

ElemType Pop(SqStack \*p)

{

ElemType x;

if(p->top>=0)

{

x=p->stack[p->top];

printf("之前的栈顶数据元素 %d 已经被删除！\n",p->stack[p->top]);

p->top=p->top-1;

return(x);

}

else

{

printf("Underflow!\n");

return(0);

}

}

ElemType GetTop(SqStack \*p)

{

ElemType x;

if(p->top>=0)

{

x=p->stack[p->top];

printf("\n栈顶元素为:%d\n",x);

return(x);

}

else

{

printf("Underflow!\n");

return(0);

}

}

void OutStack(SqStack \*p)

{

int i;

printf("\n");

if(p->top<0)

printf("这是一个空栈！");

printf("\n");

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

printf("第 %d 个元数据元素为：%6d\n",i,p->stack[i]);

}

void setEmpty(SqStack \*p)

{

p->top=-1;

}

void main()

{

SqStack \*q;

int cord;

ElemType a;

printf("第一次使用必须初始化！\n");

do

{

printf("\n");

printf("\n-----------------------主菜单------------------\n");

printf("\n----------------1 初始化顺序栈---------------\n");

printf("\n----------------2 插入一个元素---------------\n");

printf("\n----------------3 删除栈顶元素---------------\n");

printf("\n----------------4 取栈顶元素-----------------\n");

printf("\n----------------5 置空顺序栈-----------------\n");

printf("\n----------------6 结束程序运行---------------\n");

printf("\n-----------------------------------------------\n");

printf("请从菜单栏中选择（1,2,3,4,5,6）");

scanf("%d",&cord);

printf("\n");

switch(cord)

{

case 1:

{

q=(SqStack \*)malloc(sizeof(SqStack));

InitStack(q);

OutStack(q);

}

break;

case 2:

{

printf("请输入要插入的数据元素:a=");

scanf("%d",&a);

Push(q,a);

OutStack(q);

}

break;

case 3:

{

Pop(q);

OutStack(q);

}

break;

case 4:

{

GetTop(q);

OutStack(q);

}

break;

case 5:

{

setEmpty(q);

printf("\n顺序栈被置空!\n");

OutStack(q);

}

break;

case 6:

exit(0);

}

}while(cord<=6);

}

参考程序2.2

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef int ElemType;

typedef struct StackNode

{

ElemType data;

StackNode \*next;

}StackNode;

typedef struct

{

StackNode \*top;//栈顶指针

}

LinkStack;

void InitStack(LinkStack \*s)

{

s->top=NULL;

printf("\n已经初始化链栈\n");

}

void setEmpty(LinkStack \*s)

{

s->top=NULL;

printf("\n链栈被置空！\n");

}

void pushLstack(LinkStack \*s,ElemType x)

{

StackNode \*p;

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

p->data=x;

p->next=s->top;

s->top=p;

}

ElemType popLStack(LinkStack \*s)

{

ElemType x;

StackNode \*p;

p=s->top;

if(s->top==0)

{

printf("\n栈空，不能出栈！\n");

return(0);

}

x=p->data;

printf("\n当前出栈的数据是：%d",x);

s->top=p->next;

free(p);

return(x);

}

ElemType StackTop(LinkStack \*s)

{

ElemType x;

if(s->top==0)

{

printf("\n链栈空!\n");

return 0;

}

{

x=s->top->data;

printf("\n\n当前链栈的栈顶元素为：%d\n",x);

return(x);

}

}

void Disp(LinkStack \*s)

{

printf("\n链栈中的数据为:\n");

printf("============================\n");

StackNode \*p;

p=s->top;

while(p!=NULL)

{

printf("%d\n",p->data);

p=p->next;

}

printf("============================\n");

}

void main()

{

int i,n,a;

LinkStack \*s;

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

int cord;

printf("第一次使用必须初始化！\n");

do

{

printf("\n");

printf("\n----------主菜单---------\n");

printf("\n---- -1 初始化链栈-----\n");

printf("\n------2 入栈-----------\n");

printf("\n------3 出栈-----------\n");

printf("\n------4 取栈顶元素-----\n");

printf("\n------5 置空链栈-------\n");

printf("\n------6 结束程序运行---\n");

printf("\n-------------------------\n");

printf("请从菜单栏中选择（1,2,3,4,5,6）");

scanf("%d",&cord);

printf("\n");

switch(cord)

{

case 1:

{

InitStack(s);

Disp(s);

}

break;

case 2:

{

printf("请输入将要压入链栈的数据个数:n=");

scanf("%d ",&n);

printf("依次将%d个数据压入链栈:\n",n);

for(i=1;i<=n;i++)

{

scanf("%d",&a);

pushLstack(s,a);

}

Disp(s);

}

break;

case 3:

{

popLStack(s);

Disp(s);

}

break;

case 4:

{

StackTop(s);

printf("\n");

}

break;

case 5:

{

setEmpty(s);

Disp(s);

}

break;

case 6:

exit(0);

}

}while(cord<=6);

}

# 实验三 队列

参考程序3.1

#define MAX 100

#define NULL 0

#include "stdio.h"

#include "string.h"

#include "stdlib.h"

typedef struct tick

{char name[10];

char first[10];

char dist[10];

char time[10];

char flay[10];

int rear,front;

}tickd;

tickd q[MAX+1];

int rear,front;

void enqueue(),dequeue(),clear(),good\_bye();

int menu\_select();

void clear()

{ }

void good\_bye()

{clear();}

int menu\_select()

{char c; int n;

clear();

printf("机票预售系统程序主控模块:\n\n\n");

printf("\t\t 1. 买票登记 \n");

printf("\t\t 2. 删除登记 \n");

printf("\t\t 0. 退 出 \n");

do{printf("\n\t\t请按数字0-2键选择功能:");

c=getchar();

n=c-48;}while((n<0)||(n>2));

return(n);

} /\*MENU\_SELECT\*/

void enqueue()//入队操作

{char s1[10], s2[10], s3[10], s4[10], s5[10];

if (rear==MAX) rear=1;

else if(((rear+1)%MAX)==front)

{printf("队列溢出错误\n"); getchar();exit(0);}

else {rear=(rear+1)%MAX;

printf("\n\n\t\t请登记所买机票信息:\n");

printf("\n\t\t请输入旅客姓名:");scanf("%s",s1);

printf("\n\t\t请输入起飞地点:");scanf("%s",s2);

printf("\n\t\t请输入飞往地点:");scanf("%s",s3);

printf("\n\t\t请输入起飞时间:");scanf("%s",s4);

printf("\n\t\t请输入航班代号:");scanf("%s",s5);

strcpy(q[rear].name,s1);

strcpy(q[rear].first,s2);

strcpy(q[rear].dist,s3);

strcpy(q[rear].time,s4);

strcpy(q[rear].flay,s5);}

}/\*ENQNQUEUE\*/

void dequeue() //删除机票登记信息即出队操作

{if (rear==front)

{printf("\n\t\t队列已空，操作错误\n");getchar(); exit(0);}

else{

front=(front+1)%MAX;

printf("\n\n\t\t删除机票登记:\n\n\t\t");

printf("旅客姓名:%s\n\t\t",q[front].name);

printf("旅客姓名:%s\n\t\t",q[front].first);

printf("起飞地点:%s\n\t\t",q[front].dist);

printf("飞往地点:%s\n\t\t",q[front].time);

printf("起飞时间:%s\n\t\t",q[front].flay);

} getchar();

}/\*DEQUEUE\*/

void main()

{int kk;

rear=0; front=0;

do{kk=menu\_select();

switch(kk)

{case 1:{enqueue();break;}

case 2:{dequeue();break;}

case 0:{good\_bye();exit(0);}

}getchar();getchar();

}while(kk!=0);

}/\*MAIN\*/

填空题。

1、int append(sqqueue \*q,Elemtype x)

{

if(q->rear+1)%MAXNUM==q->front)

printf("\n队列满！");

return FALSE;

}

q->queue[q->rear]=x;

q->rear=(q->rear+1)%MAXUM;

return TRUE;

}

2、int Delete(sqqueue \*q)

{

Elemtype x;

if(q->front==q->rear)

printf("\n队列空！");

return FALSE;

}

x=q->queue[q->front];

q->front=(q->front+1)%MAXUM;

printf("\n队列元素%d出队！\n",x);

return TRUE;

}

3、void Lappend(Lqueue \*q,int x)

{

Qnodetype \*s;

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

s->data=x;

s->next=NULL;

q->rear->next=s;

q->rear=s;

}

4、

ElemType Ldelete(Lqueue \*q)

{

Qnodetype \*p;

ElemType x;

if(q->front==q->rear)

{

printf("队列为空！\n");

return 0;

}

else

{

p=q->front->next;

q->front->next=p->next;

if(p->next==NULL)

q->rear=q->front;

x=p->data;

free(p);

}

printf("出链队列元素:%d\n",x);

return(x);

}

# 实验四

参考程序4.1

#include <dos.h>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

enum BOOL{False,True};

typedef struct BiTNode //定义二叉树节点结构

{char data; //数据域

struct BiTNode \*lchild,\*rchild; //左右孩子指针域

}BiTNode,\*BiTree;

void CreateBiTree(BiTree &); //生成一个二叉树

void PreOrder(BiTree); //先序递归遍历二叉树

void InOrder(BiTree); //中序递归遍历二叉树

void PostOrder(BiTree); //后序递归遍历二叉树

void main()

{BiTree T;

char ch,j;

int flag=1;

BOOL temp;

//---------------------程序解说-----------------------

printf("本程序实现二叉树的操作。\n");

printf("可以进行建立二叉树，递归先序、中序、后序遍历等操作。\n");

//----------------------------------------------------

printf("请将先序遍历二叉树的结果输入以建立二叉树。\n");

printf("对于叶子结点以空格表示。\n");

printf("例如:abc de g f (回车)，建立如下二叉树：\n");

printf(" a \n");

printf(" / \n");

printf(" b \n");

printf(" / \\ \n");

printf(" c d \n");

printf(" / \\ \n");

printf(" e f \n");

printf(" \\ \n");

printf(" g \n");

CreateBiTree(T); //初始化队列

getchar();

while(flag)

{ printf("请选择: \n");

printf("1.递归先序遍历\n");

printf("2.递归中序遍历\n");

printf("3.递归后序遍历\n");

printf("4.退出程序\n");

scanf(" %c",&j);

switch(j)

{case '1':if(T)

{printf("先序遍历二叉树:");

PreOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

case '2':if(T)

{printf("中序遍历二叉树:");

InOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

case '3':if(T)

{printf("后序遍历二叉树:");

PostOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

default:flag=0;printf("程序运行结束，按任意键退出!\n");

}

}

getch();

}

void CreateBiTree(BiTree &T)

{char ch;

scanf("%c",&ch); //读入一个字符

if(ch==' ') T=NULL;

else {T=(BiTNode \*)malloc(sizeof(BiTNode)); //生成一个新结点

T->data=ch;

CreateBiTree(T->lchild); //生成左子树

CreateBiTree(T->rchild); //生成右子树

}

}

void PreOrder(BiTree T)

{

if(T)

{

printf("%c",T->data); //访问结点

PreOrder(T->lchild);

PreOrder(T->rchild);

}

}

void InOrder(BiTree T)

{if(T)

{

InOrder(T->lchild);

printf("%c",T->data); //访问结点

InOrder(T->rchild);

}

}

void PostOrder(BiTree T)

{

if(T)

{

PostOrder(T->lchild);

PostOrder(T->rchild);

printf("%c",T->data); //遍历右子树

}

}

参考程序4.2

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

//\*PROGRAM :非递归遍历二叉树 \*

//\*CONTENT :建立，先序、中序、后序遍历二叉树 \*

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

#include <dos.h>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX 100 //定义堆栈最大容量

enum BOOL{False,True};

enum RVISIT{Rchildnovisit,Rchildvisit};

//在后序遍历二叉树时用来指示是否已访问过右子//树

typedef struct BiTNode //定义二叉树节点结构

{char data; //数据域

struct BiTNode \*lchild,\*rchild; //左右孩子指针域

}BiTNode,\*BiTree;

typedef struct //定义堆栈结构

{BiTree elem[MAX]; //栈区

int top; //栈顶指针

}BiTreeStack;

void Initial(BiTreeStack &); //初始化一个堆栈

BOOL Push(BiTreeStack &,BiTree); //将一个元素入//栈

BOOL Pop(BiTreeStack&,BiTree &); //将一个元素 //出栈

BOOL Gettop(BiTreeStack ,BiTree &); //取得堆栈栈//顶元素

BOOL StackEmpty(BiTreeStack); //判断堆栈是 //否已空

void CreateBiTree(BiTree &); //生成一个二叉树

void PreOrder(BiTree); //先序非递归遍 //历二叉树

void InOrder(BiTree); //中序非递归遍 //历二叉树

void PostOrder(BiTree); //后序非递归遍历//二叉树

void main()

{BiTree T;

char ch,j;

int flag=1;

BOOL temp;

//--------------------程序解说-----------------

printf("本程序实现二叉树的非递归遍历操作。\n");

printf("可以实现建立二叉树，非递归先序、中序、后序遍历二叉树\n");

//---------------------------------------------

printf("请将先序遍历二叉树的结果输入以建立二叉树。\n");

printf("对于叶子结点以空格表示。\n");

printf("例如:abc de g f (回车)，建立如下二叉树：\n");

printf(" a \n");

printf(" / \n");

printf(" b \n");

printf(" / \\ \n");

printf(" c d \n");

printf(" / \\ \n");

printf(" e f \n");

printf(" \\ \n");

printf(" g \n");

CreateBiTree(T); //生成一棵二叉树

getchar();

while(flag)

{ printf("请选择: \n");

printf("1.非递归先序遍历\n");

printf("2.非递归中序遍历\n");

printf("3.非递归后序遍历\n");

printf("4.退出程序\n");

scanf(" %c",&j);

switch(j)

{case '1':if(T)

{printf("先序遍历二叉树:");

PreOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

case '2':if(T)

{printf("中序遍历二叉树:");

InOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

case '3':if(T)

{printf("后序遍历二叉树");

PostOrder(T);

printf("\n");

}

else printf("二叉树为空!\n");

break;

default:flag=0;printf("程序运行结束，按任意键结束!\n");

}

}

getch();

}

void Initial(BiTreeStack &S)

{S.top=-1; //栈顶指针初始化为-1

}

BOOL Push(BiTreeStack &S,BiTree ch)

{//将元素ch入栈，成功返回True,失败返回False

if(S.top>=MAX-1) return False;//判断是否栈满

else {S.top++; //栈顶指针top加一

S.elem[S.top]=ch; //入栈

return True;

}

}

BOOL Pop(BiTreeStack &S,BiTree &ch)

{//将栈顶元素出栈,成功返回True，并用ch返回该//元素值，失败返回False

if(S.top<=-1) return False;//判断是否栈空

else {S.top--; //栈顶指针减一

ch=S.elem[S.top+1];

return True;

}

}

BOOL Gettop(BiTreeStack S,BiTree &ch)

{//取得栈顶元素，成功返回True，并用ch返回该 //元素值，失败返回False

if(S.top<=-1)

return False;

else {ch=S.elem[S.top];//显示栈顶元素

return True;

}

}

BOOL StackEmpty(BiTreeStack S)

{//判断堆栈是否已空，若空返回True,不空返回//False

if(S.top<=-1) return True;

else return False;

}

void CreateBiTree(BiTree &T)

{//生成一棵二叉树，该二叉树以T为根结点

char ch;

scanf("%c",&ch); //读入一个字符

if(ch==' ') T=NULL;

else {T=(BiTNode \*)malloc(sizeof(BiTNode)); //生成一个新结点

T->data=ch;

CreateBiTree(T->lchild); //生成左子树

CreateBiTree(T->rchild); //生成右子树

}

}

void PreOrder(BiTree T)

{//先序非递归遍历以T为根结点的二叉树

BiTreeStack S;

BiTree p;

Initial(S);

p=T;

while(p||!StackEmpty(S))

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

Push(S,p);

p=p->lchild;

}

else {Pop(S,p);

p=p->rchild;

}

}

printf("\n");

}

void InOrder(BiTree T)

{//中序非递归遍历以T为根结点的二叉树

BiTreeStack S;

BiTree p;

Initial(S);

p=T;

while(p||!StackEmpty(S))

{ if(p) {Push(S,p); p=p->lchild;}

else {Pop(S,p);

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

p=p->rchild;

}

}

printf("\n");

}

void PostOrder(BiTree T)

{//后序非递归遍历以T为根结点的二叉树

BiTreeStack S;

BiTree p,q;

RVISIT tag;

Initial(S);

p=T;

do {

while(p)

{Push(S,p); p=p->lchild;}

q=NULL; tag=Rchildvisit;

while(!StackEmpty(S)&&tag)

{Gettop(S,p);

if(p->rchild==q)

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

Pop(S,p);

q=p;

}

else {

p=p->rchild;

tag=Rchildnovisit;}

}

}while(!StackEmpty(S));

printf("\n");

}

# 实验五

参考程序5.1

#include <dos.h>

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{unsigned int weight; //结点权值

unsigned int parent,lchild,rchild; //结点的父指针，左//右孩子指针

}HTNode,\*HuffmanTree; //动态分配数组存//储哈夫曼树

typedef char \*\*HuffmanCode; //动态分配数组存储//哈夫曼编码表

void CreateHuffmanTree(HuffmanTree &,unsigned int\*,int ); //生成一棵哈夫曼树

void HuffmanCoding(HuffmanTree,HuffmanCode &,int ); //对哈夫曼树进行编码

void PrintHuffmanCode(HuffmanCode,unsigned int\*,int); //显示哈夫曼编码

void Select(HuffmanTree,int,int&,int&); //在数组中 //寻找权值最小的两个结点

void main()

{HuffmanTree HT; //哈夫曼树HT

HuffmanCode HC; //哈夫曼编码表HC

int n,i; //n是哈夫曼树叶子结点数

unsigned int \*w; //w存放叶子结点权值

char j='y';

//程序解说

printf("本程序将演示构造哈夫曼树.\n");

printf("首先输入叶子结点数目.\n例如:8\n");

printf("然后输入每个叶子结点的权值.\n");

printf("例如:5 29 7 8 14 23 3 11\n");

printf("程序会构造一棵哈夫曼树并显示哈夫曼编码.\n");

printf(" 5---0110\n 29---10\n 7---1110\n 8---1111\n 14---110\n");

printf(" 23---00\n 3---0111\n 11---010\n");

while(j!='N'&&j!='n')

{printf("请输入叶子结点数目:");

scanf("%d",&n); //输入叶子结点数

if(n<=1) {printf("该数不合理!\n");continue;}

w=(unsigned int\*)malloc(n\*sizeof(unsigned int)); //开辟空间存放权值

printf("请输入各叶子结点的权值:\n");

for(i=0;i<n;i++) scanf("%d",&w[i]); //输入各叶子结点权值

CreateHuffmanTree(HT,w,n); //生成哈夫曼树

HuffmanCoding(HT,HC,n); //进行哈夫曼编码

PrintHuffmanCode(HC,w,n); //显示哈夫曼编码

printf("哈夫曼树构造完毕，还要继续吗?(Y/N)");

scanf(" %c",&j);

}

}

void CreateHuffmanTree(HuffmanTree &HT,unsigned int \*w,int n)

{//w存放n个结点的权值，构造一棵哈夫曼树HT

int i,m;

int s1,s2;

HuffmanTree p;

if(n<=1) return;

m=2\*n-1; //n个叶子结点的哈夫曼树，有2\*n-1 //个结点

HT=(HuffmanTree)malloc((m+1)\*sizeof(HTNode)); //开辟2\*n各结点空间，0号单元不用

for(p=HT+1,i=1;i<=n;++i,++p,++w) //进行初始化

{p->weight=\*w;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

for(;i<=m;++i,++p)

{p->weight=0;

p->parent=0;

p->lchild=0;

p->rchild=0;

}

for(i=n+1;i<=m;++i) //建哈夫曼树

{Select(HT,i-1,s1,s2);

//从HT[1...i-1]中选择parent为0且weight //最小的两个结点，其序号分别为s1和s2

HT[s1].parent=i; HT[s2].parent=i; //修改s1和//s2结点的父指针parent

HT[i].lchild=s1; HT[i].rchild=s2; //修改i结点 //的左右孩子指针

HT[i].weight=HT[s1].weight+HT[s2].weight; //修改权值

}

}

void HuffmanCoding(HuffmanTree HT,HuffmanCode &HC,int n)

{//将有n个叶子结点的哈夫曼树HT进行编码， 所//编的码存放在HC中

//方法是从叶子到根逆向求每个叶子结点的哈夫 //曼编码

int i,c,f,start;

char \*cd;

HC=(HuffmanCode)malloc((n+1)\*sizeof(char \*)); //分配n个编码的头指针向量

cd=(char \*)malloc(n\*sizeof(char)); //开辟一个求 //编码的工作空间

cd[n-1]='\0'; //编码结束符

for(i=1;i<=n;++i) //逐个地求哈夫曼编码

{start=n-1; //编码结束位置

for(c=i,f=HT[i].parent;f!=0;c=f,f=HT[f].parent) //从叶子到根逆向求编码

if(HT[f].lchild==c) cd[--start]='0'; //若是左孩子编为'0'

else cd[--start]='1'; //若是右孩子编为'1'

HC[i]=(char \*)malloc((n-start)\*sizeof(char)); //为第i个编码分配空间

strcpy(HC[i],&cd[start]); //将编码从cd复制到HC中

}

free(cd); //释放工作空间

}

void PrintHuffmanCode(HuffmanCode HC,unsigned int \*w,int n)

{//显示有n个叶子结点的哈夫曼树的编码表

int i;

printf("HuffmanCode is :\n");

for(i=1;i<=n;i++)

{printf(" %3d---",w[i-1]);

puts(HC[i]);

}

printf("\n");

}

void Select(HuffmanTree HT,int t,int&s1,int&s2)

{//在HT[1...t]中选择parent不为0且权值最小的两//个结点，其序号分别为s1和s2

int i,m,n;

m=n=10000;

for(i=1;i<=t;i++)

{if((HT[i].parent==0)&&((HT[i].weight<m)||(HT[i].weight<n)))

if(m<n)

{n=HT[i].weight;s2=i;}

else {m=HT[i].weight;s1=i;}

}

if(s1>s2) //s1放较小的序号

{i=s1;s1=s2;s2=i;}

}

# 实验六

实验6.1

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

typedef int VexType;

typedef VexType Mgraph[MAX][MAX];

void creat\_mg(Mgraph G);

void output\_mg(Mgraph G);

Mgraph G1;

int n,e,v0;

void main()

{

creat\_mg(G1);

output\_mg(G1);

}

void creat\_mg(Mgraph G)

{

int i,j,k;

printf("\n 请输入无向图的顶点数和边数，如: 6,5:");

scanf("%d,%d",&n,&e);

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

G[i][j]=0;

for(k=1;k<=e;k++)

{

printf("\n请输入每条边的两个顶点编号，如：2,5 :");

scanf("%d,%d",&i,&j);

G[i][j]=1;

G[j][i]=1;

}

}

void output\_mg(Mgraph G)

{

int i,j;

for(i=1;i<n;i++)

{

printf("\n");

for(j=1;j<=n;j++)

printf("%5d",G[i][j]);

}

printf("\n");

}

实验6.2

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

typedef int VexType;

typedef struct Vnode

{

VexType data;

struct Vnode \*next;

}Vnode;

typedef Vnode Lgraph[MAX];

typedef struct

{

int V[MAX];

int front;

int rear;

}Queue;

void creat\_L(Lgraph G);

void output\_L(Lgraph G);

void dfsL(Lgraph G,int v);

Lgraph Ga;

int n,e,visited[MAX];

void main()

{

int v1,i;

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

visited[i]=0;

creat\_L(Ga);

output\_L(Ga);

printf("\n请输入深度优先遍历的出发点:");

scanf("%d",&v1);

printf("\n深度优先遍历的结果为:");

dfsL(Ga,v1);

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

visited[i]=0;

printf("\n请输入广度优先遍历的出发点:");

scanf("%d",&v1);

printf("\n广度优先遍历的结果为:");

dfsL(Ga,v1);

}

void creat\_L(Lgraph G)

{

Vnode \*p,\*q;

int i,j,k;

printf("\n请输入图的顶点数和边数:");

scanf("%d,%d",&n,&e);

for(i=1;i<=n;i++)

{

G[i].data=i;

G[i].next=NULL;

}

for(k=1;k<=e;k++)

{

printf("请输入每条边的关联顶点编号:");

scanf("%d,%d",&i,&j);

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

p->data=i;

p->next=G[j].next;

G[j].next=p;

q=(Vnode \*)malloc(sizeof(Vnode));

q->data=j;

q->next=G[i].next;

G[i].next=q;

}

}

void output\_L(Lgraph G)

{

int i;

Vnode \*p;

for(i=1;i<=n;i++)

{

printf("\n与[%d]关联的顶点有:",i);

p=G[i].next;

while(p!=NULL)

{

printf("%5d",p->data);

p=p->next;

}

}

}

void initqueue(Queue \*q)

{

q->front=-1;

q->rear=-1;

}

int quempty(Queue \*q)

{

if(q->front==q->rear)

{

return 1;

}

else

{

return 0;

}

}

void enqueue(Queue \*q,int e)

{

if((q->rear+1)%MAX==q->front)

printf("队列满!\n");

else

{

q->rear=(q->rear+1)%MAX;

q->V[q->rear]=e;

}

}

int dequeue(Queue \*q)

{

int t;

if(q->front==q->rear)

{

printf("队列空!\n");

return 0;

}

else

{

q->front=(q->front+1)%MAX;

t=q->V[q->front];

return t;

}

}

void dfsL(Lgraph G,int v)

{

Vnode \*p;

printf("%d->",G[v].data);

visited[v]=1;

p=G[v].next;

while(p)

{

v=p->data;

if(visited[v]==0)

dfsL(G,v);

p=p->next;

}

}

void bfsL(Lgraph g,int v)

{

int x;

Vnode \*p;

Queue \*q=(Queue \*)malloc(sizeof(Queue));

initqueue(q);

printf("\n %d->",g[v].data);

visited[v]=1;

enqueue(q,v);

while(!quempty(q))

{

x=dequeue(q);

p=g[x].next;

while(p)

{

v=p->data;

if(visited[v]==0)

{

printf("%d->",g[v].data);

visited[v]=1;

enqueue(q,v);

}

p=p->next;

}

}

printf("\n");

}

# 实验七

参考程序7.1

参考程序1

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

//\*PROGRAM :哈希表的综合操作 \*

//\*CONTENT :Insert,Search,Deltet \*

//\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

#include <dos.h>

#include <conio.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 12 //哈希表的最大容量，与所采//用的哈希函数有关

enum BOOL{False,True};

enum HAVEORNOT{NULLKEY,HAVEKEY,DELKEY};

//哈希表元素的三种状态，没有记录、有记录、//有过记录但已被删除

typedef struct //定义哈希表的结构

{int elem[MAXSIZE]; //数据元素体

HAVEORNOT elemflag[MAXSIZE]; //元素状态标//志，没有记录、有记录、有过记录但已被删除

int count; //哈希表中当前元素的个数

}HashTable;

typedef struct

{int keynum; //记录的数据域，只有关键字一项

}Record;

void InitialHash(HashTable&); //初始化哈希表

void PrintHash(HashTable); //显示哈希表中的所//有元素

BOOL SearchHash(HashTable,int,int&); //在哈希表 //中查找元素

BOOL InsertHash(HashTable&,Record); //在哈//希表中插入元素

BOOL DeleteHash(HashTable&,Record); //在哈//希表中删除元素

int Hash(int); //哈希函数

void main()

{HashTable H; //声明哈希表H

char ch,j='y';

int position;

Record R;

BOOL temp;

textbackground(3); //设定屏幕颜色

textcolor(15);

clrscr();

//-------------------------程序说明-------------------------------

printf("This program will show how to operate to a HashTable.\n");

printf("You can display all elems,search a elem,\ninsert a elem,delete a elem.\n");

//----------------------------------------------------------------

InitialHash(H);

while(j!='n')

{printf("1.display\n");

printf("2.search\n");

printf("3.insert\n");

printf("4.delete\n");

printf("5.exit\n");

scanf(" %c",&ch); //输入操作选项

switch(ch)

{case '1':if(H.count) PrintHash(H); //哈希表//不空

else printf("The HashTable has no elem!\n");

break;

case '2':if(!H.count) printf("The HashTable has no elem!\n"); //哈希表空

else

{printf("Please input the keynum(int) of the elem to search:");

scanf("%d",&R.keynum); //输入待查//记录的关键字

temp=SearchHash(H,R.keynum,position);

//temp=True:记录查找成功, //temp=False:没有找到待查记录

if(temp) printf("The position of the elem is %d\n",position);

else printf("The elem isn't exist!\n");

}

break;

case '3':if(H.count==MAXSIZE) //哈希表//已满

{printf("The HashTable is full!\n");

break;

}

printf("Please input the elem(int) to insert:");

scanf("%d",&R.keynum); //输入要插入//的记录

temp=InsertHash(H,R);

//temp=True:记录插入成功；//temp=False:已存在关键字相同的记录

if(temp) printf("Sucess to insert the elem!\n");

else printf("Fail to insert the elem.The same elem has been exist!\n");

break;

case '4':printf("Please input the keynum of the elem(int) to delet:");

scanf("%d",&R.keynum); //输入要删除记//录的关键字

temp=DeleteHash(H,R);

//temp=True:记录删除成功；//temp=False:待删记录不存在

if(temp) printf("Sucess to delete the elem!\n");

else printf("The elem isn't exist in the HashTable!\n");

break;

default: j='n';

}

}

printf("The program is over!\nPress any key to shut off the window!\n");

getchar();getchar();

}

void InitialHash(HashTable &H)

{//哈希表初始化

int i;

H.count=0;

for(i=0;i<MAXSIZE;i++) H.elemflag[i]=NULLKEY;

}

void PrintHash(HashTable H)

{//显示哈希表所有元素及其所在位置

int i;

for(i=0;i<MAXSIZE;i++) //显示哈希表中记录所在//位置

if(H.elemflag[i]==HAVEKEY) //只显示标志为HAVEKEY(存放有记录)的元素

printf("%-4d",i);

printf("\n");

for(i=0;i<MAXSIZE;i++) //显示哈希表中记录值

if(H.elemflag[i]==HAVEKEY)

printf("%-4d",H.elem[i]);

printf("\ncount:%d\n",H.count); //显示哈希表当前 //记录数

}

BOOL SearchHash(HashTable H,int k,int &p)

{//在开放定址哈希表H中查找关键字为k的数据元//素，若查找成功，以p指示

//待查数据元素在表中的位置，并返回True；否则，//以p指示插入位置，并返回False

int p1;

p1=p=Hash(k); //求得哈希地址

while(H.elemflag[p]==HAVEKEY&&k!=H.elem[p])

//该位置中填有记录并且关键字不相等

{p++; //冲突处理方法：线性探测再散列

if(p>=MAXSIZE) p=p%MAXSIZE; //循环搜索

if(p==p1) return False; //整个表已搜索完，没//有找到待查元素

}

if(k==H.elem[p]&&H.elemflag[p]==HAVEKEY) //查找成功，p指示待查元素位置

return True;

else return False; //查找不成功

}

BOOL InsertHash(HashTable &H,Record e)

{//查找不成功时插入元素e到开放定址哈希表H //中，并返回True，否则返回False

int p;

if(SearchHash(H,e.keynum,p)) //表中已有与e有相//同关键字的元素

return False;

else

{H.elemflag[p]=HAVEKEY; //设置标志为HAVEKEY，表示该位置已有记录

H.elem[p]=e.keynum; //插入记录

H.count++; //哈希表当前长度加一

return True;

}

}

BOOL DeleteHash(HashTable &H,Record e)

{//在查找成功时删除待删元素e，并返回True，否//则返回False

int p;

if(!SearchHash(H,e.keynum,p)) //表中不存在待删 //元素

return False;

else

{H.elemflag[p]=DELKEY; //设置标志为//DELKEY，表明该元素已被删除

H.count--; //哈希表当前长度减一

return True;

}

}

int Hash(int kn)

{//哈希函数：H(key)=key MOD 11

return (kn%11);

}

2、

int BinSearch1(Seqlist A[],int n,KeyType k)

int low=0,high=n-1,mid;

while(low<=high)

{

mid=(low+high)/2;

if(high==k)

{

printf("所找的元素%d下标为%d\n",k,mid);

return mid;

}

else if(A[mid].key>k)

{

high=mid-1;

}

else

{

low=mid+1;

}

return -1;

}

# 实验八

参考程序8.1

#include"stdio.h"

int num=0;

void print\_data(int data[],int first,int last)

{

int i=0;

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

printf("\*");

for(i=first;i<=last;i++)

printf("%3d",data[i]);

for(i=last;i<=8;i++)

printf("\*");

printf("\n");

}

void merge(int array[],int first,int last)/\*一趟归并\*/

{

int mid,i1,i2,i3;

int temp[10];

int i,j;

mid=(first+last)/2;

i1=0;

i2=first;

i3=mid+1;

while(i2<=mid&&i3<=last)

{

if(array[i2]<array[i3])

temp[i1++]=array[i2++];

else

temp[i1++]=array[i3++];

}

if(i2<=mid)

while(i2<=mid)

temp[i1++]=array[i2++];

if(i3<=last)

while(i3<=last)

temp[i1++]=array[i3++];

for(i=first,j=0;i<=last;i++,j++)

array[i]=temp[j];

print\_data(array,first,last);

}

void mergesort(int data[],int first,int last)/\*归并排序\*/

{

int mid;

if(first<last)

{

mid=(first+last)/2;

mergesort(data,first,mid);

mergesort(data,mid+1,last);

print\_data(data,first,last);

merge(data,first,last);

}

}

void main()

{

int a[]={1,8,6,4,10,5,3,2,22};/\*可根据实际情况初始化\*/

mergesort(a,0,8);

}

参考程序8.2

#include<iostream.h>

#include"stdio.h"

#define MAX 10

int QuickSort(int a[],int l,int r)

{

int pivot;//枢轴

int i=l;

int j=r;

int tmp;

pivot=a[(l+r)/2];//取数组中间的数为枢轴

do

{

while(a[i]<pivot)

i++;//i右移

while(a[j]>pivot)

j--;//j左移

if(i<=j)

{

tmp=a[i];

a[i]=a[j];

a[j]=tmp;//交换a[i]和a[j]

i++;

j--;

}

}

while(i<=j);

if(l<j)

QuickSort(a,l,j);

if (i<r)

QuickSort(a,i,r);

return 1; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{ int array[MAX];

int i;

printf("请输入%d个整数:",MAX);

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

scanf("%d",&array[i]);

QuickSort(array,0,MAX-1);

printf("快速排序后:");

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

printf("%d ",array[i]);

printf("\n");

return 0;

}