上海理工大学光电信息与计算机工程学院

**《数据结构》实验报告**

****

**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：**

目录

[实验1 顺序表的基本操作 3](#_Toc477172901)

[实验2 单链表的基本操作 7](#_Toc477172902)

[实验3 栈和队列的基本操作 12](#_Toc477172903)

[实验4 二叉树的基本操作 21](#_Toc477172904)

[实验5 图的基本操作 24](#_Toc477172905)

[实验6 查找 30](#_Toc477172906)

[实验7 排序 34](#_Toc477172907)

# 实验1 顺序表的基本操作

**一、实验目的**

编程实现顺序表的创建、插入和删除。

（为方便测试，其中元素定义为：typedef int Elemtype）

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：dev c++

**三、实验内容（需写出源程序）**

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 30

typedef struct list

{

int elem[MAXSIZE];

int len;

}\*Linklist;

void creat(Linklist L,int length)

{

int i;

L->len=length;

int value;

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

{

printf("第%d个：",(i+1));

scanf("%d",&value);

L->elem[i]=value;

}

}

void insert(Linklist L,int m,int e)

{

int i;

for(i=L->len-1;i>=m-1;i--)

{

L->elem[i+1]=L->elem[i];

}

L->len++;

L->elem[i+1]=e;

}

void deletedata(Linklist L,int n)

{

int i;

for(i=n;i<L->len;i++)

{

L->elem[i-1]=L->elem[i];

}

L->len--;

}

int main()

{

int i;

int a,b,c,len,l,h;

char m,n;

m='Y';

n='Y';

h='Y';

Linklist L=(Linklist\*)malloc(sizeof(struct list));

printf("请输入你要创建的顺序表的长度:");

scanf("%d",&len) ;

creat(L,len);

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

{

printf("%d\n",L->elem[i]);

}

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

printf("请输入你要执行的操作：1、插入元素;2、删除元素:");

scanf("%d",&l);

switch(l){

case 1:

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

printf("请输入你要插入的元素位置：");

scanf("%d",&a);

if(a<1||a>len){

printf("你的输入有误！");

getchar();

printf("是否继续？Y/N:");

scanf("%c",&m);

}

else{

printf("请输入你要插入元素的值:");

scanf("%d",&b) ;

insert(L,a,b);

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

{

printf("%d\n",L->elem[i]);

}

}break;

}break;

case 2:

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

printf("请输入你要删除的元素位置：");

scanf("%d",&c);

if(c<1||c>len){

printf("输入错误！");

getchar();

printf("是否继续？Y/N：");

scanf("%c",&n);

}

else{

deletedata(L,c);

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

printf("%d\n",L->elem[i]);

break;

}

}

break;

default :printf("你的输入有误!");

}

getchar();

printf("是否继续？Y/N：");

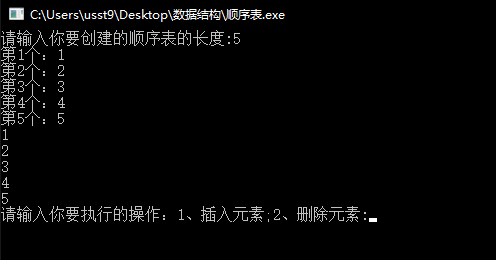
scanf("%c",&h);

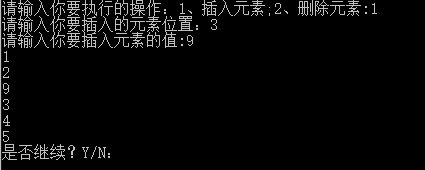
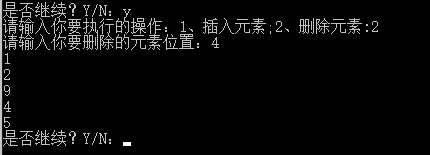
}

return 0;

}

1. **实验结果（写出运行程序后的结果截图）**



* 
* 

# 实验2 单链表的基本操作

**一、实验目的**

编程实现单链表的创建、插入和删除。

（为方便测试，其中元素定义为：typedef int Elemtype）

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：

1. **实验内容（需写出源程序）**

#include<stdio.h>

#include<stdlib.h>

typedef int Elemtype;

typedef struct node{

Elemtype data;

struct node \*next;

}Lnode;

void creat(Lnode \*head,int n){

Lnode \*p1,\*s;

int i;

p1=head;

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

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

printf("第%d个节点数据:",i);

scanf("%d",&s->data);

p1->next=s;

p1=s;

}

p1->next=NULL;

}

void print(Lnode \*head){

Lnode \*p;

p=head->next;

if(p==NULL){

printf("链表为空！");

}

while(p!=NULL){

printf("数据:%d\t\n",p->data);

p=p->next;

}

}

void destroy(Lnode \*head){

Lnode \*p;

Lnode \*q;

p=head;

while(p->next!=NULL){

q=p->next;

free(p);

p=q;

}

}

void insert(Lnode \*head,int a,int e){

int i;

Lnode \*p;

Lnode \*q;

Lnode \*s;

p=head;

q=p->next;

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

s->data=e;

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

p=q;

q=q->next;

}

s->next=q;

p->next=s;

}

void deletee(Lnode \*head,int m){

int i;

Lnode \*p;

Lnode \*q;

p=head;

q=p->next;

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

p=q;

q=q->next;

}

p->next=q->next;

free(q);

}

int main(){

int m,n;

Lnode \*head;

int a,b,c;

char w,r,s;

w='Y';

r='Y';

s='Y';

head=(Lnode\*)malloc(sizeof(Lnode));

printf("创建链表！\n");

printf("请输入你要创建的链表长度:");

scanf("%d",&m);

creat(head,m);

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

printf("请选择你要进行的操作！\n");

printf("1········插入\n");

printf("2········删除\n");

printf("操作数:");

scanf("%d",&n);

switch(n){

case 1:

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

printf("请输入你要插入元素的位置:");

scanf("%d",&a);

if(a<1||a>m){

printf("你的输入有误!");

getchar();

printf("是否继续？Y/N：");

scanf("%c",&r);

}

else{

printf("请输入你要插入元素的值:");

scanf("%d",&b);insert(head,a,b);print(head); break;

}

}break;

case 2:

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

if(head->next==NULL){

printf("链表为空，无法删除！");

break;

}

else{

printf("请输入你要删除的元素的位置:");

scanf("%d",&c);

if(c<1||c>m){

printf("你的输入有误！");

getchar();

printf("是否继续？Y/N：");

scanf("%c",&s);

}

else{

deletee(head,c);print(head);break;

}

}

}break;

default :printf("你的输入有误！");

}

getchar();

printf("是否继续？Y/N:");

scanf("%c",&w);

}

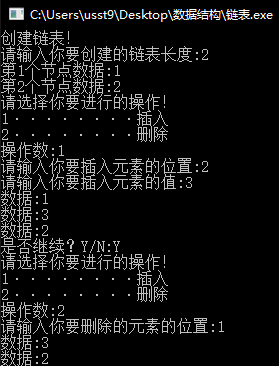
printf("程序已停止运行！");

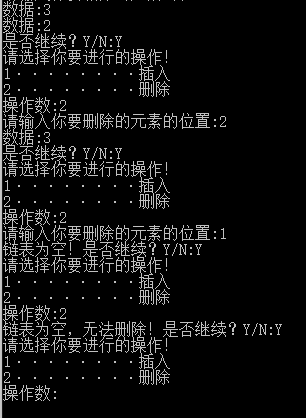
destroy(head);

return 0;

}

**四、实验结果（写出运行程序后的结果截图）**





# 实验3 栈和队列的基本操作

**一、实验目的**

编程实现顺序栈和顺序队列的创建，并能实现入栈、出栈、入队和出队等算法。

（为方便测试，其中元素定义为：typedef int Elemtype）

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：

1. **实验内容（需写出源程序）**

**//栈**

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 100

#define ADDSIZE 10

typedef int Elemtype;

typedef struct stack

{

int len;

Elemtype \*top;

Elemtype \*base;

}SqStack;

void initstack(SqStack \*s)

{

s->base=(Elemtype \*)malloc(MAXSIZE\*sizeof(Elemtype));

if(!s->base) printf("ERROR");

s->top=s->base;

s->len=s->top-s->base;

}

void push(SqStack \*s,Elemtype e)

{

if(s->top-s->base==MAXSIZE){

printf("栈满！");

s->base=(Elemtype \*)realloc(s->base,(ADDSIZE+s->len)\*sizeof(Elemtype));

if(!s->base) printf("ERROR!");

s->top=s->base+ADDSIZE;

}

else{

\*(s->top)=e;

s->top++;

s->len++;

}

}

void pop(SqStack \*s)

{

if(s->len==0)

{

printf("栈空！无法执行出栈操作！\n");

}

else

{

s->top--;

s->len--;

if(s->len==0){

printf("栈空！\n");

}else{

printf("已删除栈顶元素！");

}

}

}

void getTop(SqStack \*s)

{

if(s->len==0){

printf("栈为空，无头元素！\n");

}

else{

printf("栈顶元素为:%d\n",\*(s->top-1));

}

}

void display(SqStack \*s)

{

Elemtype \*p=s->top;

if(s->len==0){

printf("栈为空，无法打印！\n");

}

else{

printf("栈自顶向下的元素为:\n");

while(p!=s->base)

{

p--;

printf("%d",\*p);

printf("\n");

}

}

}

int main()

{

int i=0;

int m,n,a,b;

char a1,a2,a3,a4;

a1='Y';

a2='Y';

a3='Y';

a4='Y';

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

initstack(s);

printf("\*\*\*\*\*\*\*栈的基本操作\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*1、入栈\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*2、出栈\*\*\*\*\*\*\*\*\*\*\n");

while(a2=='Y'||a2=='y'){

printf("请输入你要进行的操作:");

scanf("%d",&a);

switch(a){

case 1:

printf("请输入你要入栈的元素:");

scanf("%d",&n);

push(s,n);

getTop(s);

display(s);

break;

case 2:

pop(s);

getTop(s);

display(s);

break;

default:printf("你的输入有误！");

break;

}

getchar();

printf("是否继续？Y/N:");

scanf("%c",&a2);

}

printf("程序已停止运行！");

return 0;

}

**//顺序队**

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX 30**

**typedef int Elemtype;**

**typedef struct queue{**

**Elemtype front;**

**Elemtype rear;**

**Elemtype data[MAX];**

**}\*SqQueue;**

**SqQueue InitQueue(){**

**SqQueue q;**

**q=(SqQueue)malloc(sizeof(SqQueue));**

**q->front = q->rear = -1;**

**return q;**

**}**

**void getHead(SqQueue q){**

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

**printf("队列为空，无头元素!\n");**

**}**

**else**

**printf("头元素为:%d\n",q->data[q->front]);**

**}**

**int getLength(SqQueue q){**

**return q->rear-q->front;**

**}**

**Insert(SqQueue q,Elemtype e){**

**if(q->rear== MAX-1){**

**printf("队满，无法插入元素！\n");**

**}**

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

**q->front = q->rear=0;**

**q->data[q->rear]=e;**

**q->rear++;**

**}**

**else{**

**q->data[q->rear]=e;**

**q->rear++;**

**}**

**}**

**void DelElement(SqQueue q,Elemtype y){**

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

**printf("队列为空，无法删除！\n");**

**}**

**y=q->data[q->front];**

**q->front++;**

**}**

**void clear(SqQueue q){**

**q->front=q->rear=-1;**

**printf("队列已清空！\n");**

**}**

**void destroy(SqQueue q){**

**free(q);**

**printf("队列已销毁！\n");**

**}**

**print(SqQueue q){**

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

**printf("队列为空！\n");**

**}**

**else{**

**int i;**

**printf("队列元素从头到尾为:\n");**

**for(i=q->front;i<q->rear;i++){**

**printf("%d\n",q->data[i]);**

**}**

**}**

**}**

**int main(){**

**int i;**

**int m,n,y;**

**char a,b;**

**int k;**

**a='y';**

**b='y';**

**SqQueue q=InitQueue();**

**while(a=='y'||a=='Y'){**

**printf("请输入你要进行的操作:1,入队；2,出队:");**

**scanf("%d",&k);**

**switch(k){**

**case 1:**

**printf("请输入你要入队的元素个数:");**

**scanf("%d",&m);**

**while(b=='Y'||b=='y'){**

**if(m>MAX){**

**printf("要入队的元素个数大于队列长度！\n");**

**getchar();**

**printf("是否重新输入要入队的元素个数？Y/N：");**

**scanf("%c",&a);**

**}**

**else break;**

**}**

**for(i=0;i<m;i++){**

**printf("请输入你要入队的第%d个元素:",(i+1));**

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

**Insert(q,n);**

**}**

**getHead(q);**

**print(q);**

**printf("队列长度为:%d\n",getLength(q));**

**break;**

**case 2:**

**DelElement(q,y);**

**print(q);**

**getHead(q);**

**printf("队列长度为:%d\n",getLength(q));**

**break;**

**default: break;}**

**getchar();**

**printf("是否继续？Y/N:");**

**scanf("%c",&a) ;**

**}**

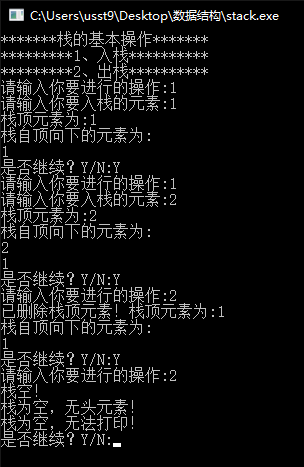
**clear(q);**

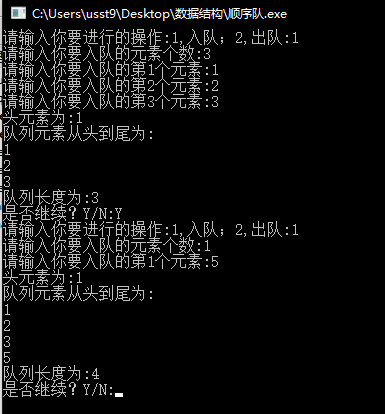
**destroy(q);**

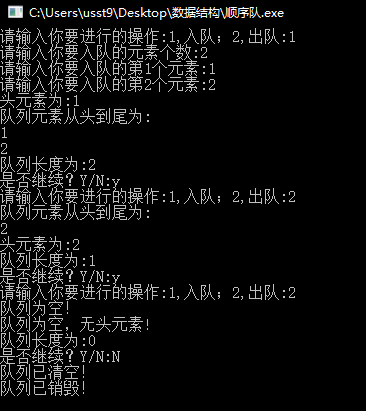
**return 0;**

**}**

**四、实验结果（写出运行程序后的结果截图）**







# 实验4 二叉树的基本操作

**一、实验目的**

利用二叉链表方法编程实现建立二叉树，以及二叉树的前序和中序遍历算法。

（为方便测试，其中元素定义为：typedef int Elemtype）

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：

1. **实验内容**（需写出源程序）

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 30

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 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);

}

}

int main(){

BTnode \*s;

s = creat();

printf("先序遍历:");

preOrder(s);

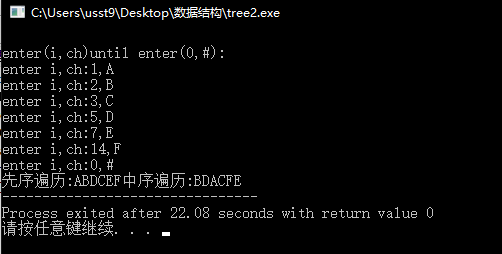
printf("中序遍历:");

inOrder(s);

return 0;

}

**四、实验结果**（写出运行程序后的结果截图）



# 实验5 图的基本操作

**一、实验目的**

用邻接矩阵表示法存储一个图，编程实现图的深度优先搜索和广度优先搜索遍历算法。

（为方便测试，其中元素定义为：typedef int Elemtype）

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：DEV C++

1. **实验内容**（需写出源程序）

1、图的深度优先搜索:

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAXVEX 100

#define MAXSIZE 100

typedef int Edgetype;

typedef int Elemtype;

typedef char Vertextype;

typedef struct graph{

Vertextype vex[MAXVEX];

Edgetype arc[MAXVEX][MAXVEX];

int numNodes,numEdges;

}MGraph;

void CreateMGraph(MGraph \*Gp){

int i,j,k;

printf("请输入顶点数:");

scanf("%d",&Gp->numNodes);

printf("请输入边数:");

scanf("%d",&Gp->numEdges);

printf("请输入顶点信息:\n");

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

scanf("%s",&Gp->vex[i]);

}

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

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

Gp->arc[i][j]=0;

}

}

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

printf("请输入边(vi,vj)的标号i,j:");

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

Gp->arc[i][j]=1;

Gp->arc[j][i]=1;

}

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

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

printf("%d\t",Gp->arc[i][j]);

}

printf("\n");

}

}

int visited[MAXVEX] = {0};

void DFS(MGraph \*Gp,int i){

int j;

visited[i]=1;

printf("%c",Gp->vex[i]);

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

if(Gp->arc[i][j]==1&&visited[j]==0){

DFS(Gp,j);

}

}

}

int main(){

MGraph G;

int i;

CreateMGraph(&G);

printf("深度优先搜索:");

DFS(&G,0);

}

2、图的广度优先搜索：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAXVEX 100

#define MAXSIZE 100

typedef int Edgetype;

typedef int Elemtype;

typedef char Vertextype;

typedef struct graph{

Vertextype vex[MAXVEX];

Edgetype arc[MAXVEX][MAXVEX];

int numNodes,numEdges;

}MGraph;

typedef struct queue{

Elemtype front;

Elemtype rear;

Elemtype data[MAXSIZE];

}\*SqQueue;

int isEmpty(SqQueue q){

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

return 1;

}

return 0;

}

SqQueue InitQueue(){

SqQueue q;

q=(SqQueue)malloc(sizeof(SqQueue));

q->front = q->rear = -1;

return q;

}

void Insert(SqQueue q,Elemtype e){

if(q->rear== MAXSIZE-1){

printf("队满，无法插入元素！\n");

}

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

q->front = q->rear=0;

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

q->rear++;

}

else{

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

q->rear++;

}

}

void DelElement(SqQueue q,Elemtype y){

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

printf("队列为空，无法删除！\n");

}

y=q->data[q->front];

q->front++;

}

void CreateMGraph(MGraph \*Gp){

int i,j,k;

printf("请输入顶点数:");

scanf("%d",&Gp->numNodes);

printf("请输入边数:");

scanf("%d",&Gp->numEdges);

printf("请输入顶点信息:\n");

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

scanf("%s",&Gp->vex[i]);

}

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

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

Gp->arc[i][j]=0;

}

}

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

printf("请输入边(vi,vj)的标号i,j:");

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

Gp->arc[i][j]=1;

Gp->arc[j][i]=1;

}

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

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

printf("%d\t",Gp->arc[i][j]);

}

printf("\n");

}

}

int visited[MAXVEX];

void BFS(MGraph \*Gp){

int i,j;

int y;

SqQueue q = InitQueue();

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

if(visited[i]==0){

visited[i]=1;

printf("%c",Gp->vex[i]);

Insert(q,i);

while(!isEmpty(q)){

DelElement(q,y);

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

if(Gp->arc[i][j==1]&&visited[j]==0){

visited[j]=1;

printf("%d",Gp->vex[j]);

Insert(q,j);

}

}

}

}

}

}

int main(){

MGraph MG;

CreateMGraph(&MG);

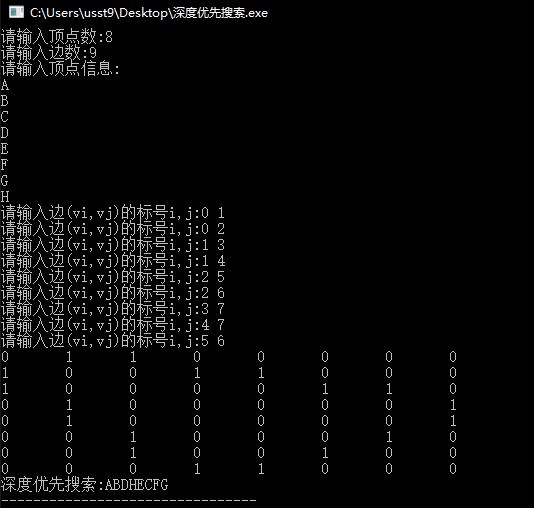
printf("广度优先搜索:");

BFS(&MG);

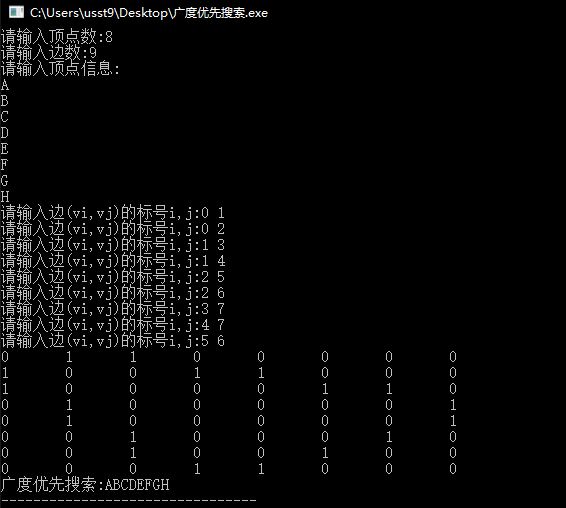
return 0;

}

1. **实验结果**（写出运行程序后的结果截图）
2. 图的深度优先搜索：



1. 图的广度优先搜索：



**实验6 查找**

**一、实验目的**

编程实现对一组无序的整型数，按要求查找指定的元素，并在屏幕上显示是否找到，若找到显示其所在的次序。

编程实现对一组有序的整型数（从小到大），按折半查找算法找指定的元素，并在屏幕上显示是否找到，若找到显示其所在的次序。

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：dev c++

1. **实验内容**（需写出源程序）

//查找

#include<stdio.h>

int sq\_find(int arr[],int n,int key){

int i;

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

if(arr[i]==key)

return (i+1);

}

return -1;

}

int main(){

int i;

int a,c;

int arr[100];

char m;

m='y';

printf("请输入一组乱序的数！");

printf("你要输入的数的个数:");

scanf("%d",&a);

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

printf("第%d个:",(i+1));

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

}

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

printf("请输入你要查找的数:");

scanf("%d",&c);

if((sq\_find(arr,a,c))==-1){

printf("未找到！\n");

}

else{

printf("已找到，是第%d个。\n",sq\_find(arr,a,c));

}

getchar();

printf("是否继续查找？Y/N:");

scanf("%c",&m);

}

return 0;

}

//折半查找

#include<stdio.h>

int bi\_find(int arr[],int e,int begin ,int last){

int mid;

if(begin>last)

return -1;

while(begin<=last){

mid=(begin+last)/2;

if(e==arr[mid])

return mid;

else if(arr[mid]<e)

begin = mid+1;

else if(arr[mid]>e)

last = mid-1;

}

return -1;

}

int main(){

int a;

int i,b;

char c;

c='y';

int arr[100];

printf("请输入一组从小到大的的整数!\n");

printf("请输入这组数的个数:");

scanf("%d",&b);

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

printf("第%d个:",(i+1));

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

}

//int arr[10] = {1,2,3,4,5,6,7,8,9,10};

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

printf("请输入你要查找的元素:");

scanf("%d",&a);

if((bi\_find(arr,a,0,b))==-1){

printf("未找到！");

}

else{

printf("已找到,是第%d个！\n",(bi\_find(arr,a,0,b)+1));

}

getchar();

printf("是否继续查找？Y/N:");

scanf("%c",&c);

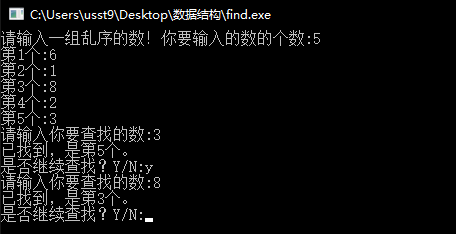
}

return 0;

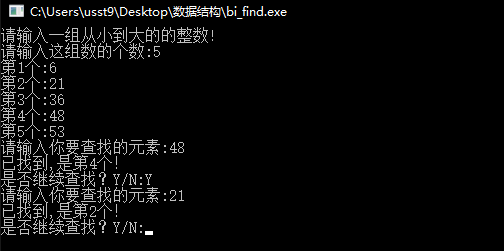
}

1. **实验结果**（写出运行程序后的结果截图）

查找：



**折半查找**



# 实验7 排序

**一、实验目的**

编程实现对一组无序的整型数，本别采用直接插入排序、简单选择排序、堆排序算法从小到大排序，并在屏幕上显示排序结果。

**二、实验软硬件要求**

硬件：一台安装了windows操作系统的计算机。

软件：dev c++

1. **实验内容**（需写出源程序）

1、直接插入排序：

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 30

typedef char Datatype;

typedef struct{

int key;

Datatype oth;

}RcdType;

RcdType r[MAXSIZE-1];

void InsertSort(RcdType r[],int n){

int i,j;

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

r[0]=r[i];

j=i-1;

while(r[0].key<r[j].key){

r[j+1]=r[j];

j--;

r[j+1]=r[0];

}

}

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

printf("%d,%c\n",r[i].key,r[i].oth);

}

}

int main(){

int i,a;

printf("请输入你要进行排序的元素个数:");

scanf("%d",&a);

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

printf("第%d个元素的键值:",(i+1));

scanf("%d",&r[i+1].key);

printf("第%d个元素的数据:",(i+1));

scanf("%s",&r[i+1].oth);

}

InsertSort(r,a);

return 0;

}

2、简单选择排序：

#include<stdio.h>

#include<stdlib.h>

#define MAX 30

typedef char Datatype;

typedef struct{

int key;

Datatype data;

}RcdType;

RcdType r[MAX-1];

void SelectSort(RcdType r[],int n){

int i,j,k;

RcdType t;

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

k=i;

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

if(r[j].key<r[k].key)

k=j;

if(i!=k){

t=r[i];

r[i]=r[k];

r[k]=t;

}

}

}

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

printf("%d,%c\n",r[i].key,r[i].data);

}

}

int main(){

int i,a;

RcdType r[MAX-1];

printf("请输入你要进行排序的元素个数:");

scanf("%d",&a);

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

printf("请输入第%d个元素键值:",(i+1));

scanf("%d",&r[i+1].key);

printf("请输入第%d个元素的数据:",(i+1));

scanf("%s",&r[i+1].data);

}

SelectSort(r,a);

return 0;

}

3、堆排序：

#include<stdio.h>

#include<stdlib.h>

#define MAX 30

typedef char DataType;

typedef struct

{

int key;

char data;

}RcdType;

RcdType r[MAX-1];

void Heap(RcdType r[],int i,int m)

{

int j;

RcdType x;

x=r[i];

j=2\*i;

while(j<=m){

if(j<m){

if(r[j].key>r[j+1].key)

j++;

if(r[j].key<x.key){

r[i]=r[j];

i=j;

j=2\*i;

}

else break;

}

else{

if(r[i].key>r[j].key){

r[i]=r[j];

i=j;

j=2\*i;

}

else break;

}

}

r[i]=x;

}

void HeapSort(RcdType r[],int n){

int i;

RcdType x;

for(i=n/2;i>=1;i--){

Heap(r,i,n);

}

for(i=n;i>=1;i--){

x=r[1];

r[1]=r[i];

r[i]=x;

Heap(r,1,i-1);

}

}

int main()

{

int i;

int a;

printf("请输入要进行排序的元素个数:");

scanf("%d",&a);

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

printf("请输入第%d个元素的键值:",i);

scanf("%d",&r[i].key);

printf("请输入第%d个元素的数据:",i);

scanf("%s",&r[i].data);

}

printf("堆排序之前的顺序:");

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

printf("%d,%c\t",r[i].key,r[i].data);

}

printf("\n堆排序之后的顺序:");

HeapSort(r,a);

for(i=a;i>=1;i--){

printf("%d,%c\t",r[i].key,r[i].data);

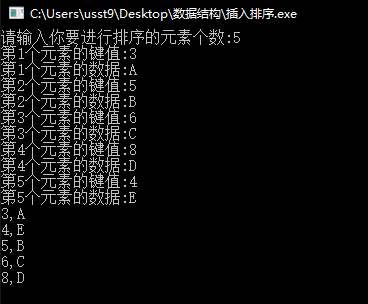
}

return 0;

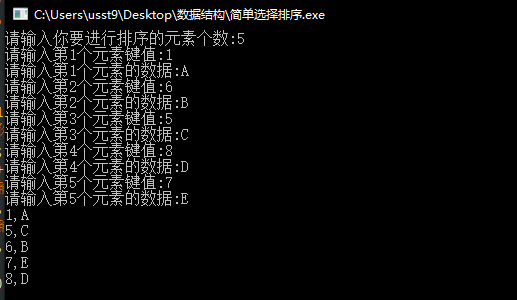
}

**四、实验结果**（写出运行程序后的结果截图）

1、直接插入排序：



2、简单选择排序：



3、堆排序：

