



東北大學

生物医学工程专业 实验报告

实验课程： 数据结构

实验题目： 实验一~实验八

班级： 生医 1502 姓名： 尚麟静

学号： 20155467 同组人： 无

指导教师： 李建华

实验成绩（教师签字）： _____

实验一 线性表

实验目的：理解线性表的创建、插入和删除操作；掌握顺序表的定义、插入和删除操作时对数据元素的移动。

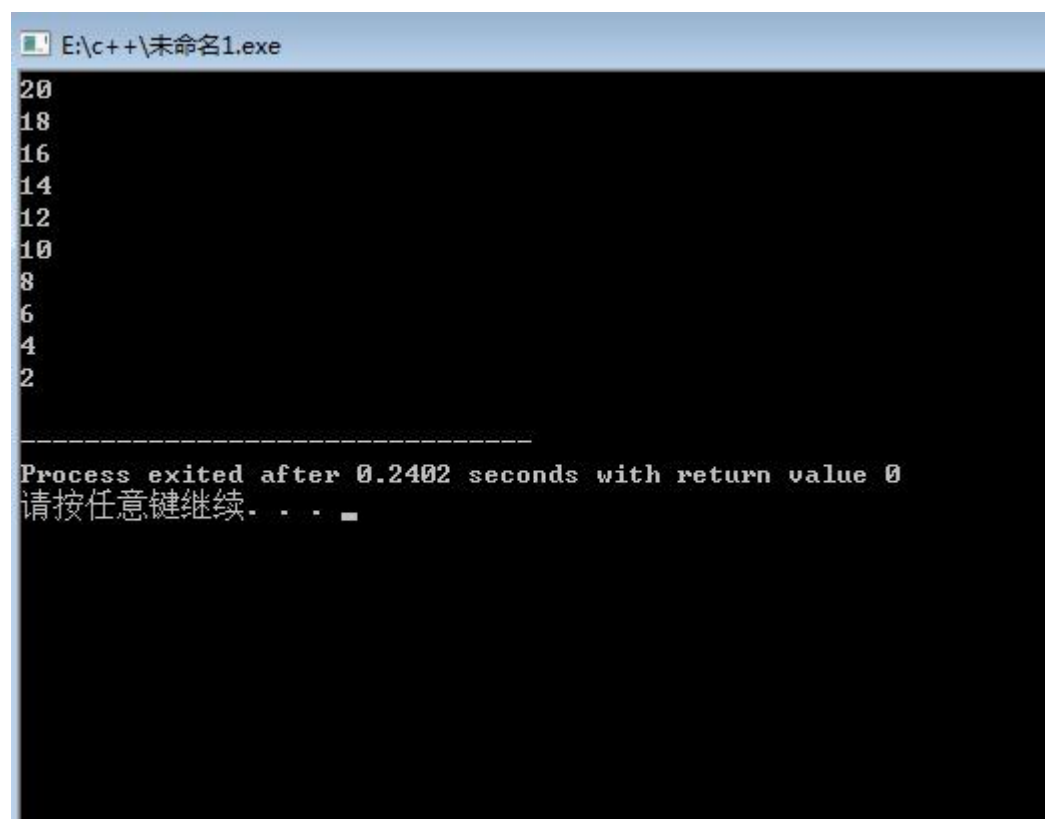
实验内容：1. 通过结构体声明顺序存储的线性表；

2. 在初始化操作中，对上述线性表分配存储空间，如可容纳 100 个元素；
3. 编写插入元素的操作；
4. 编写输出元素的操作；
5. 在主函数中声明一个顺序存储的线性表，通过不断插入新元素，构建保存元素的线性表；依次输出各元素，验证线性表结构的正确性。

实验要求：1. 对重要语句进行备注，以表明对程序的正确理解。

2. Visual C++ 或 Dev C++

程序源码：



```
#include<iostream>
#include<cstdlib>
```

```

#define TURE    1
#define FALSE   0
#define OK      1
#define ERROR   0
#define OVERFLOW -2
#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10

using namespace std;
typedef int ElemType;
typedef int Status;

typedef struct SqList
{
    ElemType* elem;
    int length;
    int listsize;
}SqList;

Status InitList_Sq(SqList &S)
{
    S.elem = (ElemType*)malloc(
        LIST_INIT_SIZE*sizeof(ElemType));
    if(!S.elem)
        exit(OVERFLOW);
    S.length = 0;
    S.listsize = LIST_INIT_SIZE;
    return OK;
}

Status ListInsert_Sq(SqList&S,int i,ElemType b)
{
    if(i<1||i>(S.length+1)) return ERROR;
    ElemType* p; ElemType* q;
    q = &(S.elem[i-1]);
    for(p = &(S.elem[(S.length)-1]);p>=q;--p)
        *(p+1) = *p;
    *q = b;
    ++(S.length);
    return OK;
}

int main()
{

```

```

int i; SqList S1 ;
    InitList_Sq(S1);
    for(i=1;i<10;i++)
    {
        ListInsert_Sq(S1,1,i*2);
    }
    for(i=1;i<=S1.length;i++)
    {
        cout<<S1.elem[i-1]<<' '<<endl;
    }
    return 0;
}

```

实验二 栈与队列

实验目的：掌握栈的创建、入栈和出栈操作；理解顺序栈在入栈和出栈时对 Top 指针的移动。

实验内容：1. 通过结构体声明线性栈；

2. 在初始化操作中，对栈分配存储空间，如可容纳 100 个元素；

3. 编写入栈、出栈、栈空判断（可选）操作；

4. 编写函数，利用栈将 10 进制的数转换为 8 进制的数；

5. 在主函数中对栈进行测试：

声明一个新栈，初始化后。


（1）输入 2,4,6,... ...,98,100 等 50 个元素，然后退栈输出所有元素。

（2）将 10 进制数 5678 转化为 8 进制数并输出。

开发工具：visual C++ 或 Dev C++

程序源码：

一、



```

100 98 96 94 92 90 88 86 84 82 80 78 76 74 72 70 68 66 64 62 60 58 56 54 52 50 48 46 44 42 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2
Press any key to continue

```

```

#include<iostream>
#include<cstdlib>
#include<stdio.h>
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define STACK_INIT_SIZE 100
#define STACKINCREMENT 10
using namespace std;
typedef int SElemType;
typedef int Status;
typedef struct SqStack{
    SElemType *base;
    SElemType *top;
    int stacksize;
}SqStack;
Status InitStack(SqStack &S)
{
    S.base=(SElemType *)malloc(STACK_INIT_SIZE *sizeof(SElemType));
    if(!S.base)exit(OVERFLOW);
    S.top=S.base;
    S.stacksize=STACK_INIT_SIZE;
    return OK;
}
Status Push(SqStack&S,SElemType t)
{
    if(S.top-S.base>=S.stacksize){
        S.base=(SElemType*)realloc(S.base,(S.stacksize+STACKINCREMENT)*sizeof(SElemType));
        if(!S.base)exit(OVERFLOW);
        S.top=S.base+S.stacksize;
        S.stacksize+=STACKINCREMENT;
    }
    *S.top++=t;
    return OK ;
}
//Push
Status Pop(SqStack&S,SElemType&t){
    if(S.top==S.base)return ERROR;
    t=*--S.top;
    return OK;
}
int main()
{
    SqStack S;
    int t;
    int i;

```

```

    InitStack(S);
    for(i=2;i<=STACK_INIT_SIZE ;i+=2)
    {Push(S,i);
    }
    for(i=2;i<=STACK_INIT_SIZE ;i+=2)
    {Pop(S,t);
    cout<<t<<' ';
    }
cout<<endl;
return 0;

}

```

```

二、

```

```

12
10
8
6
4
2
请输入一个整数
150
转化为8进制整数：
226

```

```

#include<iostream>
#include<cstdlib>
#include<stdio.h>
#define OK 1
#define ERROR 0
#define OVERFLOW -2
#define STACK_INIT_SIZE 100
#define STACKINCREMENT 10
#define TRUE 1
#define FALSE 0

using namespace std;
typedef int SElemType;
typedef int Status;
typedef struct sqstack{
    SElemType*base;
    SElemType*top;
    int stacksize;
}SqStack;

Status InitStack(SqStack & S){
    S.base=(SElemType*)
        malloc(STACK_INIT_SIZE*sizeof(SElemType));
    if(!S.base)exit(OVERFLOW);
    S.top=S.base;
}

```

```

        S.stacksize=STACK_INIT_SIZE;
        return OK;
    }

    Status Push(SqStack & S,SElemType t){
        *S.top++=t;
        return OK;
    }

    Status Pop(SqStack & S,SElemType & t){
        if(S.top==S.base)
            return ERROR;
        else
            t=*--S.top;
            return OK;
    }

    void conversion(SqStack &S,int N){
        InitStack(S);
        SElemType t;
        while(N){
            Push(S,N%8);
            N=N/8;
        }
        cout<<"转化为 8 进制整数: \n";
        while(S.top!=S.base){
            Pop(S,t);
            cout<<t;
        }
    }

    int main(){
        SqStack s;
        InitStack(s);
        for(int i=1;i<=50;i++){
            Push(s,i*2);
        }
        SElemType t;
        for(int j=1;j<=50;j++){
            Pop(s,t);
            cout<<t<<endl;
        }

        int N;
        cout<<"请输入一个整数\n";
        cin>>N;
        SqStack A;
        InitStack(A);

        conversion(A,N);

        return 0;
    }

```

```
}
```

实验三 串

实验目的：掌握字符串堆分配存储的构造与输出操作；利用求串长、串比较、求子串操作实现串的模式匹配（index）。

实验内容：1. 通过结构体声明堆存储的串，并构造一个串；

2. 实现串的输出；

3. 编写求串长、串比较、求子串操作；

4. 根据 3 中的操作，实现串的模式匹配；

5. 在主函数中声明两个串，通过模式匹配测试一个串是否存在于另一个串中。

开发工具：visual C++ 或 Dev C++

程序源码：

```
#include<iostream>
```

```
#include<cstdlib>
```

```
#define OK 1
```

```
#define ERROR 0
```

```
#define OVERFLOW -2
```

```
using namespace std;
```

```
typedef int Status;
```

```
typedef struct {
```

```
    char*ch;
```

```
    int length;
```

```
}HString;
```

```
Status StrAssign(HString & T, char*chars) {
```

```
    int i, j;
```

```
    char*c;
```

```
    for (i = 0, c = chars; *c; ++i, ++c);
```

```
    if (!i) {
```

```
        T.ch = NULL;
```

```
        T.length = 0;
```

```
    }
```

```
    else {
```

```
        if (!(T.ch = (char*)malloc(i*sizeof(char))))
```

```
            exit(OVERFLOW);
```

```
        for (j = 0; j <= i - 1; ++j) {
```



```

        T.ch[j] = chars[j];
    }
    T.length = i;
}
return OK;
}

int StrLength(HString S) {
    return S.length;
}

int StrCompare(HString S, HString T)
{
    int i;
    for (i = 0; i <= S.length && i < T.length; ++i)
        if (S.ch[i] != T.ch[i])
            return S.ch[i] - T.ch[i];
    return S.length - T.length;
}

Status SubString(HString & Sub, HString S, int pos, int len) {
    int j;
    if (pos < 1 || pos > S.length || len < 0 || len > S.length - pos + 1)
        return ERROR;
    //if(Sub.ch)free(Sub.cn);
    if (!len) {
        Sub.ch = NULL;
        Sub.length = 0;
    }
    else {
        Sub.ch = (char*)malloc(len*sizeof(char));
        for (j = 0; j <= len - 1; ++j)
            Sub.ch[j] = S.ch[j + pos - 1];
        Sub.length = len;
    }
    return OK;
}

int index(HString S, HString T, int pos) {
    int n, m, i;
    HString sub;
    if (pos > 0) {
        n = StrLength(S);
        m = StrLength(T);

```

```

        i = pos;
        while (i <= n - m + 1) {
            SubString(sub, S, i, m);
            if (StrCompare(sub, T) != 0)
                ++i;
            else
                return i;
        }
    }
    return 0;
}

void StrPrint(HString S) {
    int j;
    for (j = 0; j <= S.length - 1; ++j)
        cout << S.ch[j];
    cout << endl;
}

int main() {
    HString S, S1, S2, sub;
    char*mainstring = "Helloworld";
    StrAssign(S1, mainstring);
    char*substring = "world";
    StrAssign(S2, substring);
    StrPrint(S1);
    StrPrint(S2);
    cout << "The location is" << index(S1, S2, 1) << endl;
    system("pause");
    return 0;
}

```

```

Helloworld
world
The location is6
请按任意键继续. . .

```

实验四 数组

实验目的：掌握利用三元组实现稀疏矩阵的顺序存储；掌握三元组的基本操作，实现矩阵的转置。

- 实验内容：
1. 通过结构体声明三元组类型和稀疏矩阵类型；
 2. 实现矩阵的创建和输出；
 3. 编写矩阵转置函数；
 4. 在主函数中声明一个稀疏矩阵，转置后并输出。

开发工具：visual C++ 或 Dev C++

程序源码：

```
#include <iostream>
#define MAXSIZE 12500
#define OK 1
#define ERROR 0
#define MAXSIZE 10000
using namespace std;
typedef int Status;
typedef int ElemType;
typedef struct{
    int i,j;
    ElemType e;
}Triple;
typedef struct{
    Triple data[MAXSIZE +1];
    int mu,tu,nu;
}TSMatrix;//系数矩阵类型

Status TransposeSMatrix(TSMatrix M,TSMatrix &T){
    T.mu=M.nu;
    T.nu=M.mu;
    T.tu=M.tu;
    if(T.tu){
        int q=1,col,p;
        for(col=1;col<=M.nu;col++){
            for(p=1;p<=M.tu;p++){
                if(M.data[p].j==col){
                    T.data[q].i=M.data[p].j;
                    T.data[q].j=M.data[p].i;
                    T.data[q].e=M.data[p].e;
                    ++q;
                }
            }
        }
    }
}
```

```

    }
    }
}
return OK;
}

void CreateSMatrix(TSMatrix &M,int mu,int nu,int tu){
    M.mu=mu;M.nu=nu;M.tu=tu;
    int k;
    M.data[0].e=0;
    cout<<"行数:"<<mu<<",列数:"<<nu<<endl;
    cout<<"请输入"<<tu<<"个三元组(i j elem):"<<endl;
    for(k=1;k<=tu;k++){
        cin>>M.data[k].i>>M.data[k].j>>M.data[k].e;
    }
}

void PrintSMatrix(TSMatrix &M){

    for(int n=1;n<=M.tu;n++){
        cout<<M.data[n].i<<M.data[n].j<<M.data[n].e<<endl;}
    }

int main(){
    TSMatrix A;
    TSMatrix B;
    CreateSMatrix(A,4,5,3);
    PrintSMatrix(A);
    TransposeSMatrix(A,B);
    PrintSMatrix(B);
    return 0;
}

```

```

行数: 4, 列数: 5
请输入3个三元组 (i j elem):
233
546
721
271
323
456

```

实验五 树

实验目的：掌握先序遍历二叉树方法实现二叉树的二叉链表存储；掌握二叉树的三种遍历方法。

- 实验内容：
1. 通过结构体声明二叉树结点；
 2. 创建并存储二叉树；
 3. 实现二叉树的先序、中序和后序遍历。

开发工具：Dev C++或 Visual C++

程序源码：#include<iostream>

#include<cstdlib>

#define OK 1

#define ERROR 0

#define OVERFLOW -2

using namespace std;

typedef int Status;

typedef char TElemType;

typedef struct BiTNode{

TElemType data;

struct BiTNode*lchild,*rchild;

}BiTNode,*BiTree;

Status PrintElem(TElemType e){

cout<<e<<" ";

return OK;

}

Status CreateBiTree(BiTree&T){

TElemType ch;

ch=cin.get();

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

else{

if(!(T=(BiTNode*)malloc(sizeof(BiTNode))))exit(OVERFLOW);

T->data=ch;

CreateBiTree(T->lchild);

CreateBiTree(T->rchild);

}

return OK;

}

Status Preorder(BiTree T,Status(*Visit)(TElemType e)){

if(T){

if(PrintElem(T->data))

if(Preorder(T->lchild,Visit))

```

        if(Preorder(T->rchild,Visit))return OK;
        return ERROR;
    }else return OK;
}
int main()
{
    BiTree T;
    cout<<"input preorder string:"<<endl;
    CreateBiTree(T);
    cout<<"Preorder:"<<endl;
    Preorder(T,PrintElem);
    return 0;
}

```

```

Status PrintElem(TElemType e){
    cout<<e<<" ";
    return OK;
}

Status CreateBiTree(BiTree&T){
    TElemType ch;
    ch=cin.get();
    if(ch=='-')T=NULL;
    else{
        if(!(T=(BiTNode*)malloc(sizeof(BiTNode))))exit(OVERFLOW);
        T->data=ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }
    return OK;
}

Status Preorder(BiTree T,Status(*Visit)(TElemType e)){
    if(T){
        if(PrintElem(T->data))
            if(Preorder(T->lchild,Visit))
                if(Preorder(T->rchild,Visit)) return OK;
        return ERROR;
    }
    else return OK;
}

int main()
{
    BiTree T;
    cout<<"input preorder string:"<<endl;
    CreateBiTree(T);
    cout<<"Preorder:"<<endl;
    Preorder(T,PrintElem);

    return 0;
}

```

```

#include<iostream>
#include<cstdlib>
#define OK 1
#define ERROR 0
#define OVERFLOW -2
using namespace std;
typedef int Status;

```

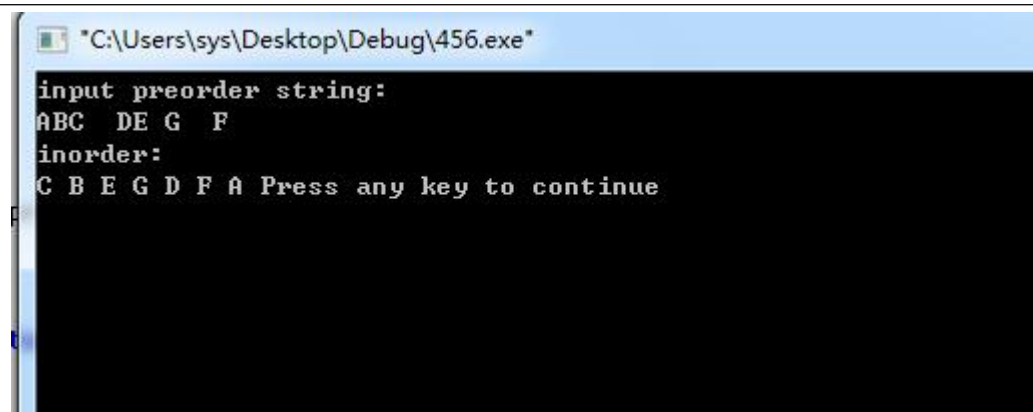
```

typedef char TElemType;
typedef struct BiTNode{
    TElemType data;
    struct BiTNode*lchild,*rchild;
}BiTNode,*BiTree;
Status PrintElem(TElemType e){
    cout<<e<<" ";
    return OK;
}
Status CreateBiTree(BiTree&T){
    TElemType ch;
    ch=cin.get();
    if(ch==' ')T=NULL;
    else{
        if(!(T=(BiTNode*)malloc(sizeof(BiTNode))))exit(OVERFLOW);
        T->data=ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }
    return OK;
}

Status inorder(BiTree T,Status(*Visit)(TElemType e)){
    if(T){
        if(inorder(T->lchild,Visit))
            if(PrintElem(T->data))
                if(inorder(T->rchild,Visit))return OK;
                return ERROR;
        }else return OK;
    }
}

int main()
{
    BiTree T;
    cout<<"input preorder string:"<<endl;
    CreateBiTree(T);
    cout<<"inorder:"<<endl;
    inorder(T,PrintElem);
    return 0;
}

```



```
"C:\Users\sys\Desktop\Debug\456.exe"
input preorder string:
ABC DE G F
inorder:
C B E G D F A Press any key to continue
```

```
#include<iostream>
#include<cstdlib>
#define OK 1
#define ERROR 0
#define OVERFLOW -2
using namespace std;
typedef int Status;
typedef char TElemType;
typedef struct BiTNode{
    TElemType data;
    struct BiTNode*lchild,*rchild;
}BiTNode,*BiTree;
Status PrintElem(TElemType e){
    cout<<e<<" ";
    return OK;
}
Status CreateBiTree(BiTree&T){
    TElemType ch;
    ch=cin.get();
    if(ch==' ')T=NULL;
    else{
        if(!(T=(BiTNode*)malloc(sizeof(BiTNode))))exit(OVERFLOW);
        T->data=ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }
    return OK;
}
Status Postorder(BiTree T,Status(*Visit)(TElemType e)){
    if(T){
        if(Postorder(T->lchild,Visit))
            if(Postorder(T->rchild,Visit))
                if(PrintElem(T->data))return OK;
        return ERROR;
    }
```



```

    }else return OK;
}
int main()
{
    BiTree T;
    cout<<"input preorder string:"<<endl;
    CreateBiTree(T);
    cout<<"Postorder:"<<endl;
    Postorder(T,PrintElem);
    return 0;
}

```



实验六 图

实验目的：掌握利用邻接矩阵存储图；掌握顶点和边的类型定义，实现邻接矩阵的输出。

- 实验内容：
1. 通过邻接矩阵存储图的边；通过一维数组存储图的顶点；
 2. 声明无向图的类型；
 3. 定义无向图的创建函数和输出函数。
 4. 在主函数中声明创建一个无向图，实现邻接矩阵存储并输出。

开发工具：Dev C++或 Visual C++

程序源码：

```

#include "stdafx.h"
#include "iostream"
#include "iomanip"
#define INFINITY 0
#define MAX_VERTEX_NUM 20

```

```

using namespace std;

typedef int VRType;
typedef int infoType;
typedef char VerTexType;
typedef struct ArcCell{
VRType adj;
infoType *info;
}ArcCell,AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];

typedef struct{
VerTexType vexs[MAX_VERTEX_NUM];
AdjMatrix arcs;
int vexnum,arcnum;
}MGraph;

int LocateVex(MGraph G,VerTexType v)
{
for(int i=0;i<G.vexnum;i++)
if(G.vexs[i]==v)
return i;
return -1;
}

void CreateGraph(MGraph &G){
int weigh;
int i=0,j=0,k=0;
char vex1,vex2;
cout<<"请输入两个数"<<endl;
cin>>G.vexnum>>G.arcnum;
cout<<endl;
cout<<"您输入的第一个数字是: "<<G.vexnum;
for(i=0;i<G.vexnum;i++)
cin>>G.vexs[i];
for(i=0;i<G.vexnum;i++)
for(j=0;j<G.vexnum;j++)
G.arcs[i][j].adj=INFINITY;
cout<<endl;
cout<<"您输入的第二个数字是: "<<G.arcnum;
for(k=0;k<G.arcnum;k++)
{
cout<<k<<":";
cin>>vex1; cin>>vex2; cin>>weigh;
i=LocateVex(G,vex1); j=LocateVex(G,vex2);

```

```

G.arcs[i][j].adj=weigh;G.arcs[j][i].adj=weigh;
cout<<endl;
}
}
void PrintGraph(MGraph &G){
int i,j;
for(i=0;i<G.vexnum;i++){
    for(j=0;j<G.vexnum;j++){
        cout<<setiosflags(ios_base::left)<<setw(10)<<G.arcs[i][j].adj;
    }
    cout<<setiosflags(ios_base::left)<<endl;
}
}
void main(){
MGraph A;
VerTexType v= 0;
CreateGraph(A);
LocateVex(A,v);
PrintGraph(A);
system("PAUSE");
}

```

实验七 查找

实验目的：掌握折半查找算法；理解其时间复杂度与顺序查找的不同。

实验内容：1. 构建一个顺序存储的数表（或数组），声明 low, high 和 middle 变量；

2. 0 号单元不用，当操作返回 0 时，意味着该数不存在数表（或数组）中；

3. 在 low<=high 时，比较 middle 位置的值和查找的关键字，不断缩小查找区间；

4. 当 low>high 时，意味查找不成功。

开发工具：Dev C++或 Visual C++

程序源码：

```

#include <iostream>
#include <cstdlib>
using namespace std;
#define NUM 15
int ST[15] = { 20,17,51,75,19,97,62,92,11,10,13,14,05,21,84 };
int BinarySearch(int T[], int len, int key)
{
    int low, high, mid, i;

```

```

low = 1, high = len;
for (i = 1; i < 10; i++)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (key < T[mid])
        {
            high = mid - 1;
        }
        else if (key > T[mid])
        {
            low = mid + 1;
        }
        else
        {
            return mid;
        }
    }
    else {
        return 0;
    }
}
}

int main() {
    int x;
    int y;
    cout << "折半查找如下:" << endl;
    cout << "请输入所要查找的数据:" << endl;
    cin >> x;
    y = BinarySearch(ST, 15, x);

    if (0 == y)
    {
        cout << "查找失败!" << endl;
    }
    else
    {
        cout << "查找的数据的位置为: " << y + 1 << endl;
    }
}

```

折半查找如下：

请输入所要查找的数据：

92

查找的数据的位置为：8

实验八 排序

实验目的：掌握直接插入排序算法；理解其算法原理和适用情况。

实验内容：1. 构建一个数组，包含 N 个待排序的数据；

2. 0 号单元不用，用于存放监视哨；

3. 通过比较和移动实现数据排序，并输出；

4. 估计算法的时间复杂性。

开发工具：Dev C++或 Visual C++

程序源码：

```
#include <iostream>
#include <iomanip>
#define MAXSIZE 20
using namespace std;

void Print(int M[], int len)
{
    for (int i=1;i<len;i++)
        cout<<M[i]<<" "<<endl;
}

void InsertionSort(int M[], int len){
```

```

        cout<<"IN ORDER"<<" ";
        int j;
        for(int i=2;i<=len;++i)
            if(M[i]<M[i-1]){
                M[0]=M[i];
                M[i]=M[i-1];
                for(j=i-2;M[0]<M[j];--j)
                    M[j+1]=M[j];
                M[j+1]=M[0];
            }
        Print(M,len);
    }

void main(){

    int y[7]={-1,41,30,35,20,10,15};
    for(int l=0;l<7;l++)
        cout<<y[l]<<" ";
    cout<<endl;
    InsertionSort(y,7);
    system("PAUSE");
}

```

```

-1 41 30 35 20 10 15
IN ORDER 10
15
20
30
35
41
请按任意键继续. . .

```