**姓名: 张景耀 学号: 3200602123 班级: 计算机2005**

### 实验内容1:

（1）对二叉排序树的进行操作，要求实现一个简单的字符界面，根据用户选择完成相应处理，并输出处理结果。功能包括：

（a）建立一棵二叉排序树：对从键盘输入的顺序任意的若干个正整数建立一颗二叉排序树，以-1作为结束。例如：输入 39 11 68 46 75 23 71 8 86 34 -1

（b）中序遍历，输出遍历结果。

（c）查找：输入一个关键字，进行查找。

（d）插入：输入一个关键字，进行插入。

（e）删除：输入一个关键字，进行删除。

（f）编写递归算法，从大到小输出二叉排序树中所有关键字不小于x的数据元素。

（2）对从键盘输入的顺序任意的10个正整数，分别通过快速排序、希尔排序、堆排序、二路归并排序使之成为有序的序列。要求输出每一趟排序的结果。

#### 源代码

#include<bits/stdc++.h>

using namespace std;

struct Node

{

    int val;

    Node\* l, \*r;

};

class BiSearchTree

{

public:

    Node\* root;

    BiSearchTree(){}

    BiSearchTree(vector<int> arr)

    {

        this->root = new Node({arr[0], NULL, NULL});

        for(int i = 1; i < arr.size() - 1; i++)

            insert(root, arr[i]);

    }

    void printTree()

    {

        cout << "Tree: " << endl;

        dfs(this->root);

        puts("");

    }

    void dfs(Node\* root)

    {

        if(!root)

            return;

        dfs(root->l);

        cout << root->val << " ";

        dfs(root->r);

    }

    Node\* insert(Node\* root, int val)

    {

        if(!root)

        {

            root = new Node({val, NULL, NULL});

            return root;

        }

        if(root->val > val)

            root->l = insert(root->l, val);

        else

            root->r = insert(root->r, val);

    }

    Node\* find(Node\* root, int val)

    {

        if(!root)

            return NULL;

        if(root->val == val)

            return root;

        if(root->val < val)

            return find(root->r, val);

        else

            return find(root->l, val);

    }

    Node\* getParent(Node\* root, Node\* p)

    {

        if(root->l == p or root->r == p)

            return root;

        if(p->val > root->val)

            return getParent(root->r, p);

        else

            return getParent(root->l, p);

    }

    Node\* getMin(Node\* root)

    {

        if(root->r)

            return getMin(root->l);

        return root;

    }

    void deleteElem(Node\* root, int val)

    {

        auto p = find(root, val);

        auto parent = getParent(this->root, p);

        int cnt = !p->l + !p->r;

        if(cnt == 2)

            parent->l == p ? parent->l = NULL : parent->r = NULL;

        else if(cnt == 1)

        {

            if(parent->l == p)

                parent->l = (p->l ? p->l : p->r);

            else

                parent->r = (p->l ? p->l : p->r);

        }

        else

        {

            auto t = getMin(p->r);

            auto v = t->val;

            deleteElem(p->r, v);

            p->val = v;

        }

    }

};

void printUI()

{

    printf("1. 建立一棵二叉排序树\n");

    printf("2. 中序遍历，输出遍历结果\n");

    printf("3. 查找：输入一个关键字，进行查找\n");

    printf("4. 插入：输入一个关键字，进行插入\n");

    printf("5. 删除：输入一个关键字，进行删除\n");

    printf("6. 从大到小输出二叉排序树中所有关键字不小于x的数据元素\n");

}

int main(int argc, char const \*argv[])

{

    printUI();

    int choice, t = 0, val;

    bool flag = true;

    BiSearchTree tree;

    vector<int> arr{39, 11, 68, 46, 75, 23, 71, 8, 86, 34, -1};

    while(flag)

    {

        cout << "\n请输入选择: " << endl;

        cin >> choice;

        switch(choice)

        {

        case 1:

            arr.clear();

            t = 0;

            while(t != -1)

            cin >> t, arr.push\_back(t);

            tree = BiSearchTree(arr);

            break;

        case 2:

            tree.printTree();

            break;

        case 3:

            cout << "请输入要查找的元素:" << endl;

            cin >> val;

            if(tree.find(tree.root, val))

                cout << "找到!!!" << endl;

            else

                cout << "未找到!!!" << endl;

            break;

        case 4:

            cout << "请输入要插入的元素:" << endl;

            cin >> val;

            tree.insert(tree.root, val);

            tree.printTree();

            break;

        case 5:

            cout << "请输入要删除的元素:" << endl;

            cin >> val;

            tree.deleteElem(tree.root, val);

            tree.printTree();

            break;

        case 6:

            cout << "请输入x:" << endl;

            cin >> val;

            cout << val << " ";

            tree.dfs(tree.find(tree.root, val)->r);

            break;

        default:

            flag = false;

            break;

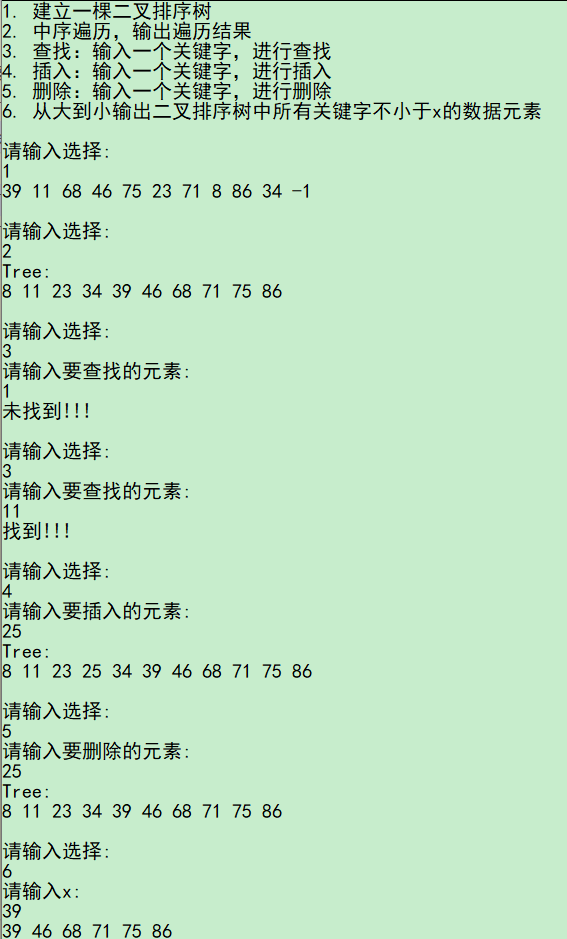
        }

    }

    cout << "欢迎使用!!!" << endl;

}

#### 测试结果



### 实验内容2:

### 对从键盘输入的顺序任意的10个正整数，分别通过快速排序、希尔排序、堆排序、二路归并排序使之成为有序的序列。要求输出每一趟排序的结果

#### 源代码

#include<bits/stdc++.h>

using namespace std;

void quick\_sort(vector<int>& arr, int left, int right)

{

    if (left >= right)

        return;

    int i = left - 1, j = right + 1, flag = arr[left + right >> 1];

    while (i < j)

    {

        while (arr[++i] < flag);

        while (arr[--j] > flag);

        if (i < j)

            swap(arr[i], arr[j]);

    }

    quick\_sort(arr, left, j);

    quick\_sort(arr, j + 1, right);

}

void sell\_sort(vector<int>& arr)

{

    int j;

    for (int gap = arr.size() >> 1; gap >  0; gap >>= 1)

        for (int i = gap; i < arr.size(); i++)

        {

            int tmp = arr[i];

            for (j = i; j >= gap and tmp < arr[j - gap]; j -= gap)

                arr[j] = arr[j - gap];

            arr[j] = tmp;

        }

}

void merge\_sort(vector<int>& arr, int left, int right, vector<int>& temp)

{

    if(left >= right)

        return;

    int mid = left + right >> 1;

    merge\_sort(arr, left, mid, temp), merge\_sort(arr, mid + 1, right, temp);

    int i = left, j = mid + 1, k = 0;

    while(i <= mid and j <= right)

        if (arr[i] <= arr[j])

            temp[k++] = arr[i++];

        else

            temp[k++] = arr[j++];

    while(i <= mid)

        temp[k++] = arr[i++];

    while(j <= right)

        temp[k++] = arr[j++];

    for(int i = left; i <= right; i++)

        arr[i] = temp[i - left];

}

//堆排序

const int maxn = 1e+5 + 10;

int\* h;

int n;

void down(int u, int n)

{

    int t = u;

    if(u \* 2 <= n and h[2 \* u] < h[t])

        t = u \* 2;

    if(u \* 2 + 1 <= n and h[2 \* u + 1] < h[t])

        t = u \* 2 + 1;

    if(u != t)

    {

        swap(h[u], h[t]);

        down(t, n);

    }

}

int top()

{

    return h[1];

}

void del\_top(int& n)

{

    h[1] = h[n];

    n--;

    down(1, n);

}

int main()

{

    vector<int> v{5, 99, 55, 262, 66, 262, 8948, 22, 11, 22, 56, 47, 52};

    cout << "原始数据: " << endl;

    for(const auto& val : v)

        cout << val << " ";

    puts("");

    cout << "快速排序" << endl;

    vector<int> arr = v;

    quick\_sort(arr, 0, arr.size() - 1);

    for(const auto& val : arr)

        cout << val << " ";

    puts("");

    cout << "希尔排序" << endl;

    arr = v;

    sell\_sort(arr);

    for(const auto& val : arr)

        cout << val << " ";

    puts("");

    cout << "归并排序" << endl;

    arr = v;

    vector<int> t(arr.size());

    merge\_sort(arr, 0, arr.size() - 1, t);

    for(const auto& val : arr)

        cout << val << " ";

    puts("");

    cout << "堆排序" << endl;

    int a[] = {5, 99, 55, 262, 66, 262, 8948, 22, 11, 22, 56, 47, 52};

    h = a;

    n = 13;

    //建堆(O(n))

    for(int i = n / 2; i; i--)

        down(i, n);

    for(int i = 0; i < 13; i++)

    {

        cout << top() << " ";

        del\_top(n);

    }

}

#### 测试结果

