《数据结构》上机报告

2018 年 11 月 12 日

姓名：刘思源 学号：1651390 班级：电子三班 得分：

<table>
<tr><td>实验题目</td><td colspan="2">汉诺塔</td></tr>
<tr><td>问题描述</td><td colspan="2">汉诺塔（又称河内塔）是源于印度一个古老传说的益智玩具。
大梵天创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上按照大小顺序摞着 64 片黄金圆盘。
大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。
现在有 n（n>=2）根高为 mi（mi>=1，1<=i<=n）的柱子，一开始第一根柱子有 k（0<=k<=m1）个从下到上放置的从大到小、大小依次为 K1~Kk 的圆盘，现在给与你一些操作步骤，你依次按照操作步骤给出结果。</td></tr>
<tr><td rowspan="2">基本要求</td><td colspan="2">1. (p1) 本题练习用栈实现汉诺塔的基本操作和错误判断。</td></tr>
<tr><td>已完成基本内容（序号）：</td><td></td></tr>
<tr><td rowspan="2">选做要求</td><td colspan="2"></td></tr>
<tr><td>已完成选做内容（序号）</td><td>1</td></tr>
<tr><td>数据结构设计</td><td colspan="2">

```
typedef struct {
    ElemType *base;
    ElemType  *top;
    int  stacksize;
 }SqStack;
```

</td></tr>
</table>

功能(函数)说明	1、 总体分析 我们所要完成的需求是：根据不同的输入完成汉诺塔的搭建；根据不同的输入完成操作：移动盘子。判断移动是否合法；判断塔是否为空，判断塔是否为满。打印汉诺塔上的盘子。完成退出。 2、 Main 函数完成汉诺塔搭建 设置变量 n，k。设立一个二维数组，来表示汉诺塔。先输入 n 个柱子，申请 n 个空间。再对每一个柱子上申请盘子的空间。将每一个盘子设置为 EMPTY。此后用这个值来判断柱子是否为空。申请盘子空间时多申请一个，用 0 号空间存储每个柱子的高度。最后将第一个柱子根据输入填满。

```
int main()
{
    int n, k,n1;
    cin >> n >> k;
    n1 = n;
    ElemType **Tower;
    int i = 0, j = 0;

    Tower = new ElemType* [n];
    for (i = 0; i<n; i++){
        Tower[i]=new ElemType[MAXSIZE];
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < MAXSIZE; j++) {
            Tower[i][j] = EMPTY;
        }
    }
    i = 0;
    j = 0;
    while (n1) {
        cin >> Tower[i][0];
        if (!cin.good()) {
            cin.ignore(1024, '\n');
            cin.clear();
            continue;
        }
        i++;
        n1--;
    }
    i = 0;
    j = 1;
    while (k) {
        cin >> Tower[i][j];
        if (!cin.good()) {
```

```
            cin.ignore(1024, '\n');
            cin.clear();
            continue;
        }
        j++;
        k--;
    }

    Hanoi(Tower, n);
    return 0;

}
```

3、 汉诺塔操作管理

设立输入的 str，来进行比对。如果输入的是 MOVE，就调用 MOVE 函数。如果输入的是 DISPLAY，就将输入的选择输入 HDisplay 函数。如果输入 QUIT，就使用一个循环，从第一个柱子调用 HDisplay 函数，全部打印完成后退出。

```
Status Hanoi(ElemType **Tower,int n)
{
    while (1) {
        char str[10];
        cin >> str;
        if (strcmp(str, "MOVE")==0) {
            HMove(Tower);
        }
        else if (strcmp(str, "DISPLAY")==0) {
            int choice;
            cin >> choice;
            HDisplay(Tower,choice);
        }
        else if (strcmp(str, "QUIT")==0) {
            int choice;
            for (choice = 0; choice < n; choice++) {
                HDisplay(Tower, choice+1);
            }
            return OK;
        }
        else {
            cin.ignore(1024, '\n');
            cin.clear();
            continue;
        }
    }
    return OK;
}
```

4、 汉诺塔移动

设立一个bool值来判断是否可进行合法的移动。如果输入的起始柱第一个数组元素是空，证明该柱子是空柱。输出提示语句。将flag置为false。如果目的柱柱高位置的元素不为空，证明目的柱为满。输出提示语句。将flag置为false。用两个变量loc1代表要移动盘子原来的位置，用loc2代表盘子目的要移动的位置。如果起始柱的loc1+1位置的元素为空，证明该元素是最上面的元素。可以进行移动，如果loc2位置的元素为空，证明该位置可进行移动。接下来如果flag为真，进行移动。输出该值。

```
Status HMove(ElemType **Tower)
{
    int from, to;
    bool flag = true;
    cin >> from >> to;
    from--;
    to--;
    if (Tower[from][1] == EMPTY) {
        cout << from+1 << " IS EMPTY" << endl;
        flag = false;
    }
    if (Tower[to][Tower[to][0]] != EMPTY) {
        cout << to+1 << " IS FULL" << endl;
        flag = false;
    }
    int loc1, loc2;
    for (loc1 = 1; loc1 <= MAXSIZE; loc1++) {
        if (Tower[from][loc1 + 1] == EMPTY) {
            break;
        }
    }
    for (loc2 = 1; loc2 <= MAXSIZE; loc2++) {
        if (Tower[to][loc2] == EMPTY) {
            break;
        }
    }
    if (Tower[to][1] !=EMPTY) {
        if (Tower[from][loc1] > Tower[to][loc2-1]) {
            cout << "ILLEGAL" << endl;
            return ERROR;
        }
    }
    if (flag) {
        Tower[to][loc2] = Tower[from][loc1];
        Tower[from][loc1] = EMPTY;
```

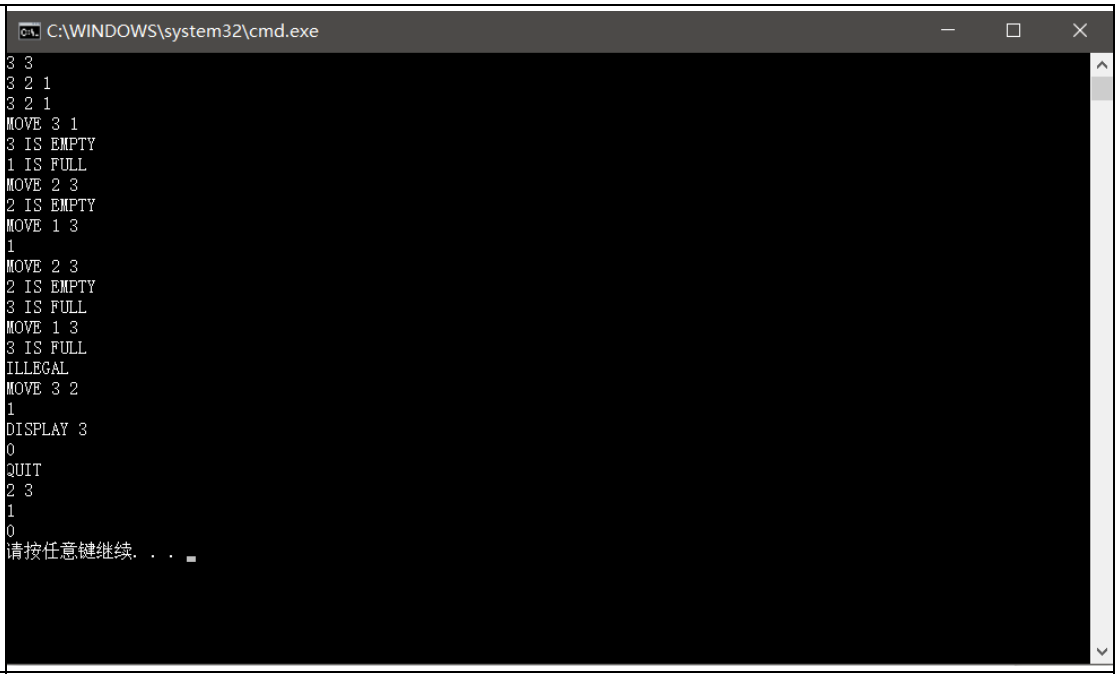
```
            cout << Tower[to][loc2] << endl;
        }
        return OK;
}
```

5、 汉诺塔打印

要打印汉诺塔时，将该数组内部所有的元素压入栈中。直到压到倒数第二个元素，再将栈中的所有元素 pop 出来打印即可。

```
Status HDisplay(ElemType **Tower,int choice)
{
    choice--;
    int i=1;
    SqStack S;
    InitStack(S);
    while (1) {
        if (Tower[choice][i] == EMPTY) {
            if (i == 1) {
                cout << 0 << endl;
                return EMPTY;
            }
            break;
        }
        Push(S, Tower[choice][i]);
        i++;
    }
    ReadAll(S);
    return OK;
}
```

开发环境	Visual studio 2017

调试分析	
心得体会	1、 使用合适的数据结构 汉诺塔本身就是一个栈的结构，在打印时直接入栈出栈打印即可。 2、 函数的多次利用 最后使用 QUIT 和 HDisplay 时，实际上要实现的功能是相似的，所以可以重复利用显示的函数多次利用，达到效率的最大化。 3、值得注意的是，本题中认为两个盘子一样大时移动是合法的。在判断时要注意。