Project 1

****

班级：16 软件工程教务三班

姓名黄俊　　学号 16350027

姓名王继武 学号 16340213

姓名谭江华 学号 16340203

【题目要求】

利用多个栈来模拟将带有不同连续编号且无序的火车通过入栈和出栈的操作使无序的火车有序化。

题目 1:编写一个程序，实现将连续无序输入的数字串利用最少的缓冲栈实现有序化。要求程序具有如下功能：

动态添加缓冲栈

（2）如果输入的是按顺序可直接输出则不经过缓冲栈

（3）能判断怎样将数字放入栈才能实现使所需的栈最少

【数据结构与算法】

（一）证明使用贪心算法可以达到所需栈最少这个最优解

首先可以经过逻辑推理得到当一个火车的编号直接输入不能直接输出而要 push 到栈中，如果所有栈顶元素都比这个编号小那么必须要启用额外的栈。

其次如果不需要额外栈的话则需要找到究竟放在哪一个栈才是最优的可以达到让栈数最小这个目的，可以知道的是放入一定是离输入编号最近的那一个栈是最好的，这就是贪心算法的核心思想，要想证明这一点可以通过反证法证明其正确性。

最后比较核心的问题就是到底是把火车编号放入已有的栈中还是令开一个栈对结果的影响哪一个比较好，乍一看很难评判，但是通过一个假设将另开一个栈等价于已有的一个栈底元素为无限大的一个编号，这个假设无疑是合理的，因为如果将一个编号放入新栈，如果接下来的元素有大过这个元素的，就另外再开一个栈，而最优解是栈最少，相对与按照贪心算法每一步都尽量考虑所有可能性减少开新栈，且因为考虑了后来进入元素的可能性，这样做不会对结果造成不可预料的后果，这样贪心算法得到的结果就可以被证明一定是最优解。

（二）总体设计

决定利用函数式编程的方法达到分治的效果，便于维护和改错。

for (int currentTrainNum : trainNum_list)//its safe to use range for 首先 main 函数主题就是这一个 range for 读取输入的语句，之后的所有操作语句都建立在这个 for 语句的内部。putTopOfTrainNumIntoContainer(listOfBufferLine,topOfBuffer);把所有栈顶元素都存在一个 vector 中方便之后的比较

```
if
(checkWhetherTheTrainCanJustGetOut(currentTrainNum,judgeNum))//to  see
whether can output the target train number
            {
```

```
            putTopOfTrainNumIntoContainer(listOfBufferLine,
topOfBuffer);
            for     (;     canTheTopOfBufferJustOut(listOfBufferLine,
topOfBuffer,                                                    judgeNum);)
{ putTopOfTrainNumIntoContainer(listOfBufferLine, topOfBuffer); }//when
a train just leave without pass py the buffer line we need to check each
top of the bufferline whether there is anther train can just pop out
            int flag = 0;//to see whether the program has been finished
so we can pass the rest code
            vector<int> regTopOfBuffer;
            putTopOfTrainNumIntoContainer(listOfBufferLine,
regTopOfBuffer);//check if the train are all out
            if (regTopOfBuffer.empty())continue;
            for (int n : regTopOfBuffer)
            {
                if (n != -1)flag = 1;
            }
            if (flag == 0)return 0;// to end the program
            continue;
        }
```

这是用来判断这个输入数字是否是目标编号，即可以直接输出不必经过缓冲栈。如果可以则直接输出，目标编号加一，之后再判断各个栈顶元素是否可以紧跟着输出。最后判断是否全部元素都已经输出，如果是的话就直接结束程序。

```
    if
(checkWhetherThereNeedABufferLine(listOfBufferLine,topOfBuffer,current
TrainNum))
        {
            getABufferLine(listOfBufferLine,currentTrainNum);//use  a
new line and put the train number into in
        }
        else
        {

    decideWhichBufferLineSuitBest(listOfBufferLine,topOfBuffer,currentT
rainNum);
        }
```

这是在判断后发现此火车编号不能直接输出，则判断是否要另开一个新栈，如果要的话直接声明一个栈并把元素压入栈中再 pushback 到 vector 中就行了。不行的话就判断哪一个栈顶元素离现有的元素最近，并放入其中。

for (; getTheTrainOuttaHere(listOfBufferLine, judgeNum) != -1;);//pop the train out at an order

这是输出的命令，知道所有元素都输出，不如目标编号一次次加大，不断从栈顶寻找元素输出。

```
int decideWhichBufferLineSuitBest(vector<stack<int>>&listOfBufferLine, vector<int>topOfBuffer, int currentTrainNum)
    {
        //cout << "is it all right" << endl;//testing
        if (topOfBuffer.empty())return -1;
        int temp = 10000;
        int count = 0, tempForCount=9999;
        for (int n : topOfBuffer)
        {
            if (n < currentTrainNum)
            {
                count++;
                continue;
            }
            else
            {
                if (temp > (n - currentTrainNum))
                {
                    temp = n - currentTrainNum;
                    tempForCount = count;
                    count++;
                }
            }
        }
        if (tempForCount != INT16_MAX)
        {
            listOfBufferLine[tempForCount].push(currentTrainNum);
            cout << "train No. " << currentTrainNum << " has get into
```

```
bufferline-- " << tempForCount << endl;
        }
        return 0;
```

}这是其中一个函数的具体实现，这是判断究竟应该放入那个栈比较好的函数，为了防止栈 underflow 在其中加上 maxINT 等价于栈顶无限大可以当作空

【测试数据、结果及分析】

（一）测试结果

```
please key in the total number of train.
>>> 9
key in the list of the destination of the trains with a space to  make it seperated
>>> 3 6 9 2 4 7 1 8 5
use a new bufferline--0 and put train  No. 5 in it
use a new bufferline--1 and put train  No. 8 in it
train No. 1 has just leave without go through the buffer line
train No. 7 has get into bufferline-- 1
train No. 4 has get into bufferline-- 0
train No. 2 has just leave without go through the buffer line
use a new bufferline--2 and put train  No. 9 in it
train No. 6 has get into bufferline-- 1
train No. 3 has just leave without go through the buffer line
train No. 4 has just leave the bufferline-- 0
train No. 5 has just leave the bufferline-- 0
train No. 6 has just leave the bufferline-- 1
train No. 7 has just leave the bufferline-- 1
train No. 8 has just leave the bufferline-- 1
train No. 9 has just leave the bufferline-- 2
```

```
please key in the total number of train.
>>> 20
key in the list of the destination of the trains with a space
to  make it seperated
>>> 5 2 7 8 9 10 20 13 4 3 11 16 15 18 19 14 12 17 1 6
use a new bufferline--0 and put train  No. 6 in it
train No. 1 has just leave without go through the buffer line
use a new bufferline--1 and put train  No. 17 in it
train No. 12 has get into bufferline-- 1
use a new bufferline--2 and put train  No. 14 in it
use a new bufferline--3 and put train  No. 19 in it
train No. 18 has get into bufferline-- 3
train No. 15 has get into bufferline-- 3
use a new bufferline--4 and put train  No. 16 in it
train No. 11 has get into bufferline-- 1
train No. 3 has get into bufferline-- 0
train No. 4 has get into bufferline-- 1
train No. 13 has get into bufferline-- 2
use a new bufferline--5 and put train  No. 20 in it
train No. 10 has get into bufferline-- 2
train No. 9 has get into bufferline-- 2
train No. 8 has get into bufferline-- 2
train No. 7 has get into bufferline-- 2
train No. 2 has just leave without go through the buffer line
train No. 3 has just leave the bufferline-- 0
train No. 4 has just leave the bufferline-- 1
train No. 5 has just leave without go through the buffer line
train No. 6 has just leave the bufferline-- 0
train No. 7 has just leave the bufferline-- 2
train No. 8 has just leave the bufferline-- 2
train No. 9 has just leave the bufferline-- 2
train No. 10 has just leave the bufferline-- 2
train No. 11 has just leave the bufferline-- 1
train No. 12 has just leave the bufferline-- 1
train No. 13 has just leave the bufferline-- 2
train No. 14 has just leave the bufferline-- 2
train No. 15 has just leave the bufferline-- 3
train No. 16 has just leave the bufferline-- 4
train No. 17 has just leave the bufferline-- 1
train No. 18 has just leave the bufferline-- 3
train No. 19 has just leave the bufferline-- 3
train No. 20 has just leave the bufferline-- 5
```

```
please key in the total number of train.
>>> 30
key in the list of the destination of the trains with a space to  make it seperated
>>> 5 2 7 24 8 9 26 10 23 20 13 25 27 4 3 22 11 28 16 15 29 18 19 21 30 14 12 17 1 6
use a new bufferline--0 and put train  No. 6 in it
train No. 1 has just leave without go through the buffer line
use a new bufferline--1 and put train  No. 17 in it
train No. 12 has get into bufferline-- 1
use a new bufferline--2 and put train  No. 14 in it
use a new bufferline--3 and put train  No. 30 in it
train No. 21 has get into bufferline-- 3
train No. 19 has get into bufferline-- 3
train No. 18 has get into bufferline-- 3
use a new bufferline--4 and put train  No. 29 in it
train No. 15 has get into bufferline-- 3
train No. 16 has get into bufferline-- 4
use a new bufferline--5 and put train  No. 28 in it
train No. 11 has get into bufferline-- 1
train No. 22 has get into bufferline-- 5
train No. 3 has get into bufferline-- 0
train No. 4 has get into bufferline-- 1
use a new bufferline--6 and put train  No. 27 in it
train No. 25 has get into bufferline-- 6
train No. 13 has get into bufferline-- 2
train No. 20 has get into bufferline-- 5
train No. 23 has get into bufferline-- 6
train No. 10 has get into bufferline-- 2
use a new bufferline--7 and put train  No. 26 in it
train No. 9 has get into bufferline-- 2
train No. 8 has get into bufferline-- 2
train No. 24 has get into bufferline-- 7
train No. 7 has get into bufferline-- 2
train No. 2 has just leave without go through the buffer line
train No. 3 has just leave the bufferline-- 0
train No. 4 has just leave the bufferline-- 1
train No. 5 has just leave without go through the buffer line
train No. 6 has just leave the bufferline-- 0
train No. 7 has just leave the bufferline-- 2
train No. 8 has just leave the bufferline-- 2
train No. 9 has just leave the bufferline-- 2
train No. 10 has just leave the bufferline-- 2
train No. 11 has just leave the bufferline-- 1
train No. 12 has just leave the bufferline-- 1
```

（二）结果分析

可以看出第一个 test9 个数据只用了三个栈，与利用穷举所得最少栈数相同。

第二个 test20 个数据用了 6 个栈。第三个 test30 个数据用了 8 个栈。

【分工、贡献%、自我评分】

王继武 程序 demo 代码 讨论 40%

黄俊 最终程序代码 报告 40%

谭江华 讨论 20%

【项目总结】

通过这次 project，熟悉栈的用法，深刻体会了编程在模拟方面的用途之大，并且知道了模拟中需要考虑很多实际因素这样导致了不一定有最优的解决办法，只能说这种方法在某种特定条件下是比较好的。

【程序清单】

```
#include "stdafx.h"
#include<iostream>
#include<stack>
#include<vector>
using namespace std;
int numOfBufferline = 0;
bool checkWhetherTheTrainCanJustGetOut(int currentTrainNum,int &judgeNum);
void putTopOfTrainNumIntoContainer(vector<stack<int>> listOfBufferLine, vector<int>
&topOfBuffer);
bool checkWhetherThereNeedABufferLine(vector<stack<int>> listOfBufferLine,vector<int>
topOfBuffer,int currentTrainNum);
bool
canTheTopOfBufferJustOut(vector<stack<int>>&listOfBufferLine,vector<int>topOfBuffer,int
&judgeNum);
void getABufferLine(vector<stack<int>>&listOfBufferLine, int currentTrainNum);
int decideWhichBufferLineSuitBest(vector<stack<int>>&listOfBufferLine,
vector<int>topOfBuffer,int currentTrainNum);
int getTheTrainOuttaHere(vector<stack<int>>&listOfBufferLine, int &judgeNum);
int main()
{
    int totalNumOfTrain;
    cout << "please key in the total number of train." << endl;
    cout << ">>>" << ends;
    cin >> totalNumOfTrain;
    stack<int>regForTrainNum;
    cout <<"key in the list of the destination of the trains with a space to  make it seperated"
<< endl;
    cout << ">>>" << ends;
    for (int i = 0; i < totalNumOfTrain; i++)
    {
        int reg;
        cin >> reg;
        regForTrainNum.push( reg);
    }
    vector<int> trainNum_list;
    for (; !regForTrainNum.empty();)
    {
        trainNum_list.push_back(regForTrainNum.top());
        regForTrainNum.pop();
    }//upper from here are all receive the input of train number
    vector<stack<int>> listOfBufferLine;
    vector<int> topOfBuffer;
```

```
    int judgeNum = 1;
    for (int currentTrainNum : trainNum_list)//its safe to use range for
    {
        putTopOfTrainNumIntoContainer(listOfBufferLine,topOfBuffer);
        if (checkWhetherTheTrainCanJustGetOut(currentTrainNum,judgeNum))//to see whether
can output the target train number
        {
            putTopOfTrainNumIntoContainer(listOfBufferLine, topOfBuffer);
            for (; canTheTopOfBufferJustOut(listOfBufferLine, topOfBuffer, judgeNum);)
{ putTopOfTrainNumIntoContainer(listOfBufferLine, topOfBuffer); }//when a train just leave
without pass py the buffer line we need to check each top of the bufferline whether there is
anther train can just pop out
            int flag = 0;//to see whether the program has been finished so we can pass the
rest code
            vector<int> regTopOfBuffer;
            putTopOfTrainNumIntoContainer(listOfBufferLine, regTopOfBuffer);//check if
the train are all out
            if (regTopOfBuffer.empty())continue;
            for (int n : regTopOfBuffer)
            {
                if (n != -1)flag = 1;
            }
            if (flag == 0)return 0;// to end the program
            continue;
        }
        if
(checkWhetherThereNeedABufferLine(listOfBufferLine,topOfBuffer,currentTrainNum))
        {
            getABufferLine(listOfBufferLine,currentTrainNum);//use a new line and put the
train number into in
        }
        else
        {
            decideWhichBufferLineSuitBest(listOfBufferLine,topOfBuffer,currentTrainNum);
        }
    }
    for (; getTheTrainOuttaHere(listOfBufferLine, judgeNum) != -1;);//pop the train out at
an order
    int a;
    cin >> a;
    return 0;
}

bool checkWhetherTheTrainCanJustGetOut(int currentTrainNum,int &judgeNum)
```

```
{
    if (currentTrainNum == judgeNum)
    {
        cout << "train No. " << currentTrainNum << " has just leave without go through the buffer line" << endl;
        judgeNum++;
        return true;
    }
    return false;
}
void putTopOfTrainNumIntoContainer(vector<stack<int>> listOfBufferLine, vector<int> &topOfBuffer)
{
    topOfBuffer.clear();
    if (!listOfBufferLine.empty())
    {
        for (auto it = listOfBufferLine.begin(); it != listOfBufferLine.end(); it++)
        {
            topOfBuffer.push_back((*it).top());
        }
    }
}
bool canTheTopOfBufferJustOut(vector<stack<int>>&listOfBufferLine,vector<int>topOfBuffer,int &judgeNum)
{

    int count = 0;
    if (topOfBuffer.empty())return false;
    for (int topNum : topOfBuffer)
    {
        //cout << "is it all right in the can function" << topNum<< endl;
        if (topNum == judgeNum)
        {
            cout << "train No. " << topNum << " has just leave the bufferline-- " << count << endl;
            listOfBufferLine[count].pop();
            judgeNum++;
            if (listOfBufferLine[count].empty())listOfBufferLine[count].push(INT16_MAX);
            return true;
        }
        count++;
    }
    return false;
```

```
}
bool checkWhetherThereNeedABufferLine(vector<stack<int>> listOfBufferLine, vector<int>
topOfBuffer, int currentTrainNum)
{
    for (int n : topOfBuffer)
    {
        if (currentTrainNum < n)return false;
    }
    return true;
}
void getABufferLine(vector<stack<int>>&listOfBufferLine, int currentTrainNum)
{
    stack<int> newBufferLine;
    newBufferLine.push(currentTrainNum);
    listOfBufferLine.push_back(newBufferLine);
    cout << "use a new bufferline--"<<numOfBufferline<< " and put train  No. " <<
currentTrainNum << " in it" << endl;
    numOfBufferline++;
}
int decideWhichBufferLineSuitBest(vector<stack<int>>&listOfBufferLine,
vector<int>topOfBuffer, int currentTrainNum)
{
    //cout << "is it all right" << endl;//testing
    if (topOfBuffer.empty())return -1;
    int temp = 10000;
    int count = 0, tempForCount=9999;
    for (int n : topOfBuffer)
    {
        if (n < currentTrainNum)
        {
            count++;
            continue;
        }
        else
        {
            if (temp > (n - currentTrainNum))
            {
                temp = n - currentTrainNum;
                tempForCount = count;
                count++;
            }
        }
    }
    if (tempForCount != INT16_MAX)
```

```
    {
        listOfBufferLine[tempForCount].push(currentTrainNum);
        cout << "train No. " << currentTrainNum << " has get into bufferline-- " << tempForCount
<< endl;
    }
    return 0;
}
int getTheTrainOuttaHere(vector<stack<int>>&listOfBufferLine, int &judgeNum)
{
    int count = 0;
    for (auto s : listOfBufferLine)
    {
        if (s.top() == judgeNum)
        {
            s.pop();
            if (s.empty())s.push(-1);
            cout << "train No. " << judgeNum << " has just leave the bufferline-- " << count
<< endl;
            judgeNum++;
            return 0;
        }
        count++;
    }
    return -1;
}
```