实验2:栈和队列

确保把类定义写在stack\_queue.h中，类实现写在stack\_queue.cpp中

给出的stack\_queue.h中的函数声明不要改动，stack\_queue.h中的内容可以增但不可以删，测试文件将根据已有的函数声明调用，stack\_queue.cpp中的打印函数不要改动。

将类和函数声明写在stack\_queue.h中，类和函数的实现写在stack\_queue.cpp中，最终提交两个文件：stack\_queue.h和stack\_queue.cpp

1. 实现无表头节点的链栈类MyStack

* 栈初始化, MyStack();//创建栈
* 销毁栈,~MyStack();//清除栈内元素,销毁栈
* 入栈, bool push(const int&);//入栈,成功返回true,失败返回false
* 出栈, bool pop(int& item); //出栈,成功返回true,失败返回false,出栈元素放在item
* 返回栈顶元素bool getTop(int& item);//获取栈顶,成功返回true,失败返回false,栈顶元素放在item
* 输出栈内元素,void printStack();//按照栈顶到栈底的顺序输出栈内所有元素

输入输出示例:

MyStack ms;

int val;

ms.push(1);

ms.printStack();//1

ms.push(2);

ms.printStack();//2,1

ms.pop(val);

cout<<val<<endl;//2

ms.printStack();//1

ms.getTop(val);

cout<<val<<endl;//1

ms.printStack();//1

1. 实现顺序循环队列类MyQueue

* 队列初始化, MyQueue(const int& capacity);//创建队列,容量为capacity
* 销毁队列,~ MyQueue();//销毁队列
* 入队,bool enQueue(const int&);//入队,成功返回true,失败返回false
* 出队,bool deQueue(int& item); //出栈,成功返回true,失败返回false, 出队元素放在item
* 返回队头元素,bool getFront(int& item); //获取队首,成功返回true,失败返回false,栈顶元素放在item
* 输出队列内元素, void printQueue();//按照队头到队尾的顺序输出栈内所有元素

输入输出示例:

MyQueue mq(10);

int val;

mq.enQueue (1);

mq.printQueue();//1

mq.enQueue (2);

mq.printQueue();//1,2

mq.deQueue (val);

cout<<val<<endl;//1

mq.printQueue();//2

mq.getFront(val);

cout<<val<<endl;//2

mq.printQueue();//2

1. 基于MyStack进行扩展改进(如改变栈内元素类型、对符号进行合理数值映射)，实现中缀表达式转后缀表达式和表达式求值，支持小括号、加减乘除四则运算,一位正整数(0-9)运算,不涉及多位数、负数及小数

* bool postfixExpression(const string&, string& result); //中缀表达式转后缀表达式,转化成功返回true, 后缀表达式放在result中,表达式错误返回false, 错误字符串Expression is wrong!放在result中
* bool expressionVal(const string&, int& result);//表达式求值,表达式正确成功计算返回true,结果放在result中,表达式错误计算不能完成返回false, result中放0

输入输出示例:

string result;

int val;

postfixExpression("5\*(7-2)",result);

cout<<result<<endl; // 572-\*

postfixExpression("5\*(7-2",result);

cout<<result<<endl; // Expression is wrong!

expressionVal("5\*(7-2)", val);

cout<<val<<endl; // 25

1. 基于MyQueue，实现输出杨辉三角

* void yanghui(const int&);//输出杨辉三角

输入输出示例:

yanghui(3);

//输出

1,1

1,2,1

1,3,3,1