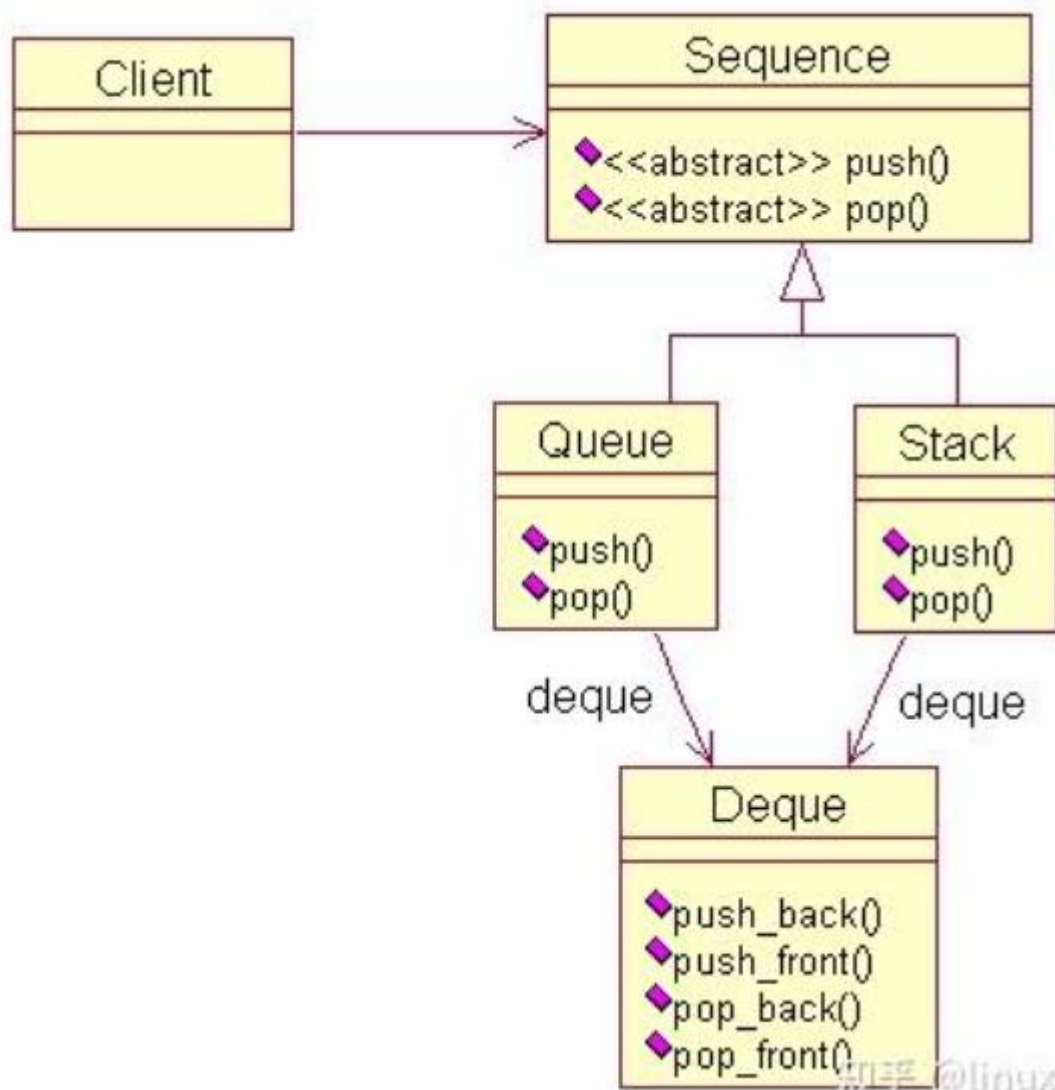


适配器设计模式

DP 上的定义：适配器模式将一个类的接口转换成客户希望的另外一个接口，使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。它包括类适配器和对象适配器，本文针对的是对象适配器。举个例子，在 STL 中就用到了适配器模式。STL 实现了一种数据结构，称为双端队列（deque），支持前后两段的插入与删除。STL 实现栈和队列时，没有从头开始定义它们，而是直接使用双端队列实现的。这里双端队列就扮演了适配器的角色。队列用到了它的后端插入，前端删除。而栈用到了它的后端插入，后端删除。假设栈和队列都是一种顺序容器，有两种操作：压入和弹出。下面给出相应的 UML 图，与 DP 上的图差不多。



根据 UML 图给出的实现如下:

```
//双端队列
class Deque
{
public:
    void push_back(int x) { cout<<"Deque push_back"<<endl; }
    void push_front(int x) { cout<<"Deque push_front"<<endl; }
    void pop_back() { cout<<"Deque pop_back"<<endl; }
    void pop_front() { cout<<"Deque pop_front"<<endl; }
};

//顺序容器
class Sequence
{
public:
    virtual void push(int x) = 0;
    virtual void pop() = 0;
};

//栈
class Stack: public Sequence
{
public:
    void push(int x) { deque.push_back(x); }
    void pop() { deque.pop_back(); }
private:
    Deque deque; //双端队列
};

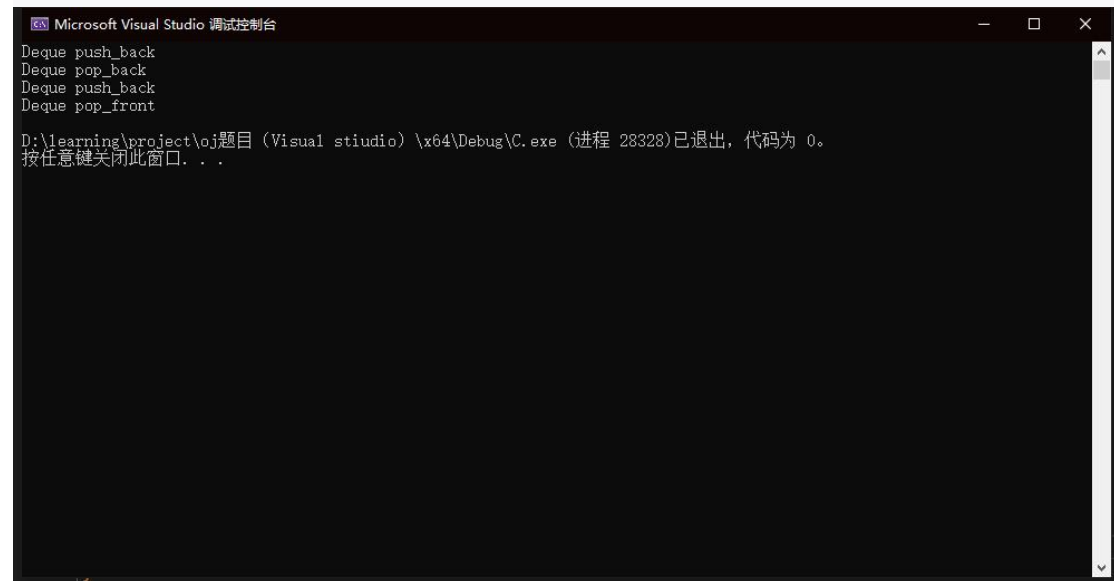
//队列
class Queue: public Sequence
{
public:
    void push(int x) { deque.push_back(x); }
    void pop() { deque.pop_front(); }
private:
    Deque deque; //双端队列
};
```

使用方式如下:

```
int main()
{
    Sequence *s1 = new Stack();
    Sequence *s2 = new Queue();
    s1->push(1); s1->pop();
    s2->push(1); s2->pop();
    delete s1; delete s2;
    return 0;
}
```

```
}
```

代码运行结果如下：



```
Microsoft Visual Studio 调试控制台  
Deque push_back  
Deque pop_back  
Deque push_back  
Deque pop_front  
D:\learning\project\oj题目 (Visual stiudio) \x64\Debug\C.exe (进程 28328)已退出，代码为 0。  
按任意键关闭此窗口. . .
```