AlexOne

六、适配器模式

目录

1 把狼伪装成羊

```
#include <iostream>
using namespace std;

// 将一个类的接口变成客户端所期待的另一种接口，从而使原本因接口不匹配而无法在一起工作的两个类能够在一起工作
// 本程序就是将一头会嚎叫的狼伪装成一头会咩咩叫的狼

class Goat{
public:
    virtual void miemie()=0;
};

class NormalGoat:public Goat{
public:
    void miemie(){
        cout<<"我是真正的羊"<<endl;
    }
};

class Wolf{
public:
    void howl(){
        cout<<"我是真正的狼"<<endl;
    }
};

class AdapterGoat:public Goat{
public:
    AdapterGoat(Wolf *w):_w(w){}
    void miemie(){
        _w->howl();
    }
protected:
    Wolf * _w;
};

int main(){
    NormalGoat ng;
    ng.miemie();

    Wolf w;
```

```
    w.howl();

    AdapterGoat ag(&w);     //把狼伪装成羊
    ag.miemie();
    return 0;
}
```

2 双端队列

```
#include <iostream>
#include <queue>
#include <stack>
#include <deque>

using namespace std;
// 双端队列经过适配后变成栈和普通队列
class MyDeque{
public:
    void push_back(int x){
        cout<<"void push_back(int x) "<<endl;
    }
    void push_front(int y){
        cout<<"void push_front(int y)"<<endl;
    }
    int pop_back(){
        cout<<"int pop_back()"<<endl;
        return 0;
    }
    int pop_front(){
        cout<<"int pop_front()"<<endl;
        return 0;
    }
};

class MySequence{
public:
    virtual int pop() = 0;
    virtual void push(int x) = 0;
};

class MyStack: public MySequence{
public:
    int pop(){
        return md.pop_back();
    }
    void push(int x){
        md.push_back(x);
    }
private:
    MyDeque md;
};

class MyQueue: public MySequence{
public:
    int pop(){
        return md.pop_front();
    }
    void push(int x){
```

```
        md.push_back(x);
    }
private:
    MyDeque md;
};

int main(){
    MyStack ms;
    ms.push(10);
    ms.pop();
    MyQueue mq;
    mq.push(10);
    mq.pop();
    return 0;
}
```