单例模式

- **懒汉模式**

```
//第一次使用的时候创建对象，C++11 保证static Singleton _eton 线程安全
class Singleton {
    private:
    Singleton():data_(99){
        std::cout << "单例对象已经创建"<< std::endl;
    }
    Singleton(const Singleton& t) = delete;
    ~Singleton(){}

    private:
    int data_;

    public:
    static Singleton& getInstance()
    {
        static Singleton _eton;
        return _eton;
    }
    int getData()
    {
        return data_;
    }
};
int main()
{
    std::cout << Singleton::getInstance().getData() << std::endl;
    return 0;
}
```

工厂模式

- 简单工厂模式

```cpp
#include <iostream>
#include<memory>

class Fruit
{
public:
    virtual void  name() = 0;
};
class Apple : public Fruit
{
public:
    void name() override
    {
        std::cout << "i am an apple" << std::endl;
    }
};
class Banana : public Fruit
{
    void name() override
    {
        std::cout << "i am an banana" << std::endl;
    }
};
class FruitFactory
{
public:
    static std::shared_ptr<Fruit> create(const std::string &name)
    {
        if (name == "apple")
        {
            return std::make_shared<Apple>();
        }
        else
        {
            return std::make_shared<Banana>();
        }
    }
};
```

```cpp
int main()
{
    std::shared_ptr<Fruit> fruit = FruitFactory::create("apple");
    fruit->name();
    fruit  = FruitFactory::create("banana");
    fruit->name();

    return 0;
}
```