

```
1 //
2 //  main.cpp
3 //  DesignPattern
4 //
5 //  Created by shadot on 2019/1/8.
6 //  Copyright © 2019 shadot. All rights reserved.
7 //
8
9 #include <iostream>
10 using namespace std;
11
12 //运算的抽象基类
13 class Operation{
14 public:
15     Operation(){
16         m_numberX = 0.0;
17         m_numberY = 0.0;
18     }
19
20     void Setxy(double x, double y){
21         m_numberX = x;
22         m_numberY = y;
23     }
24
25     virtual double GetResult() = 0;
26
27 protected:
28     double m_numberX;
29     double m_numberY;
30 };
31
32 //加减乘除具体实现类
33 class OperationAdd : public Operation{
34 public:
35     OperationAdd():Operation(){}
36
37     double GetResult(){
38         return m_numberX + m_numberY;
39     }
40 };
41
42 class OperationSub : public Operation{
```

```
43 public:
44     OperationSub():Operation(){}
45
46     double GetResult(){
47         return m_numberX - m_numberY;
48     }
49 };
50
51 class OperationMul : public Operation{
52 public:
53     OperationMul():Operation(){}
54
55     double GetResult(){
56         return m_numberX * m_numberY;
57     }
58 };
59
60 class OperationDiv : public Operation{
61 public:
62     OperationDiv():Operation(){}
63
64     double GetResult(){
65         if (m_numberY == 0)
66             return 0;
67         return m_numberX / m_numberY;
68     }
69 };
70
71 //简单运算工厂类
72 class OperationFactory
73 {
74 public:
75     enum EMOperate {Add, Sub, Mul, Div};
76
77     OperationFactory(){}
78
79     Operation* CreateOperate(EMOperate operate){
80         Operation* pOper = nullptr;
81
82         switch (operate) {
83             case Add:
84                 pOper = new OperationAdd(); break;
85             case Sub:
86                 pOper = new OperationSub(); break;
87             case Mul:
88                 pOper = new OperationMul(); break;
```

```

89         case Div:
90             p0per = new OperationDiv(); break;
91         default: break;
92     }
93     return p0per;
94 }
95 };
96
97 int main(int argc, const char * argv[]) {
98
99     Operation* p0per = OperationFactory().CreateOperate(OperationFa
100     p0per->Setxy(1, 2);
101     auto result = p0per->GetResult();
102
103     return 0;
104 }
105

```

简单运算工厂负责如何去实例化对象，以后如果需要再次进行扩展，只是在基于Operation类进行派生，然后重新实现GetResult()方法，再在工厂类switch中添加分支即可