模板特化分为全特化和偏特化

全特化是针对某一具体类型而做出特化

```
template<typename T>
struct A {

    void foo() { std::cout << "321\n"; }

    void foo1() { std::cout << "111\n"; }
};

template<>
struct A<int> {
    void foo() { std::cout << "123\n"; }
}; // 当T为int时做出特化
```

偏特化保留了一部分任意类型的性质

```
template<typename T1, typename T2>
struct B {
    void foo1() { std::cout << "T1 T2 foo1\n"; }

    void foo2() { std::cout << "T1 T2 foo2\n"; }

};

template<typename T>
struct B<T, int> {
    void foo1() { std::cout << "T int\n"; } // 当T2为int时做出特化,注意此时T1类型并没有固定
};
```

函数签名偏特化

```
template<typename T>
struct C {
    void foo() {
        std::cout << "C\n";
    }
};

template<typename T, typename...Args>
struct C<T(Args...)> { // 把一整个T(Args...)看作是一个函数类型
    void foo() {
        std::cout << "T(Args...)\n";
    }
};
```

Demo

```
int main() {
    struct A<int> a;
    a.foo(); // 123
    struct A<double> b;
    b.foo(); // 321
    b.foo1(); // 111

    struct B<double, int> c;
    c.foo1(); // T int
    B<double, double> d;
    d.foo1(); // T1 T2 foo1

    C<double> e;
    e.foo(); // C
    C<void(void)>f;
    f.foo(); // T(Args...)
}
```