C++ Template Introduction

黄昱珲

C

- 如何实现一个矩阵库？
- 为 `float` 和 `double` 分别写一套代码？
- 当然使用宏也是可以的

Templates!

- Reducing the repetition of code
- (Generalizing your code)
- Type safe
- Static polymorphism (at no cost)
- Compile time calculations

Disadvantages

1. IDE cannot give suggestion which would harm coding efficiency
2. Slow compilation
3. Cryptic compilation error messages
4. Concepts in C++20 have solved 1 and 3
5. "Modules" solves 2

Partial specialization

```
template <typename T>
struct storage;

template <>
struct storage<int> {}; // optimization for primitive types
```

Dyanmic Polymorphism

- virtual classes
- virtual methods

```
struct Shape {
    virtual void draw() = 0;
};

struct Triangle : public Shape {
    virtual void draw() override {
        // something
    }
};

void draw(const ShapeT &shape) { shape.draw(); }

// what if non-virtual methods?
```

Static Polymorphism

```
struct Shape {
        void draw() { /* Shape */ }
};

struct Triangle : public Shape {
        void draw() { /* something */ }
};

struct Rectangle : public Shape {
        void draw() { /* something */ }
};

template <typename ShapeT>
void draw(const ShapeT &shape) { shape.draw(); }
```

SFINAE

operator""

```
distance_t operator""_km(long double d) { return distance_t(d *
distance_t operator""_m(long double d) { return distance_t(d);
distance_t operator""_dm(long double d) { return distance_t(d /
distance_t operator""_cm(long double d) { return distance_t(d /

cout << (10_km + 1_m + 5_dm + 4_cm);

std::complex<double> c = 1 + 5i; // C++14
```