D2DS | COURSES | 2024

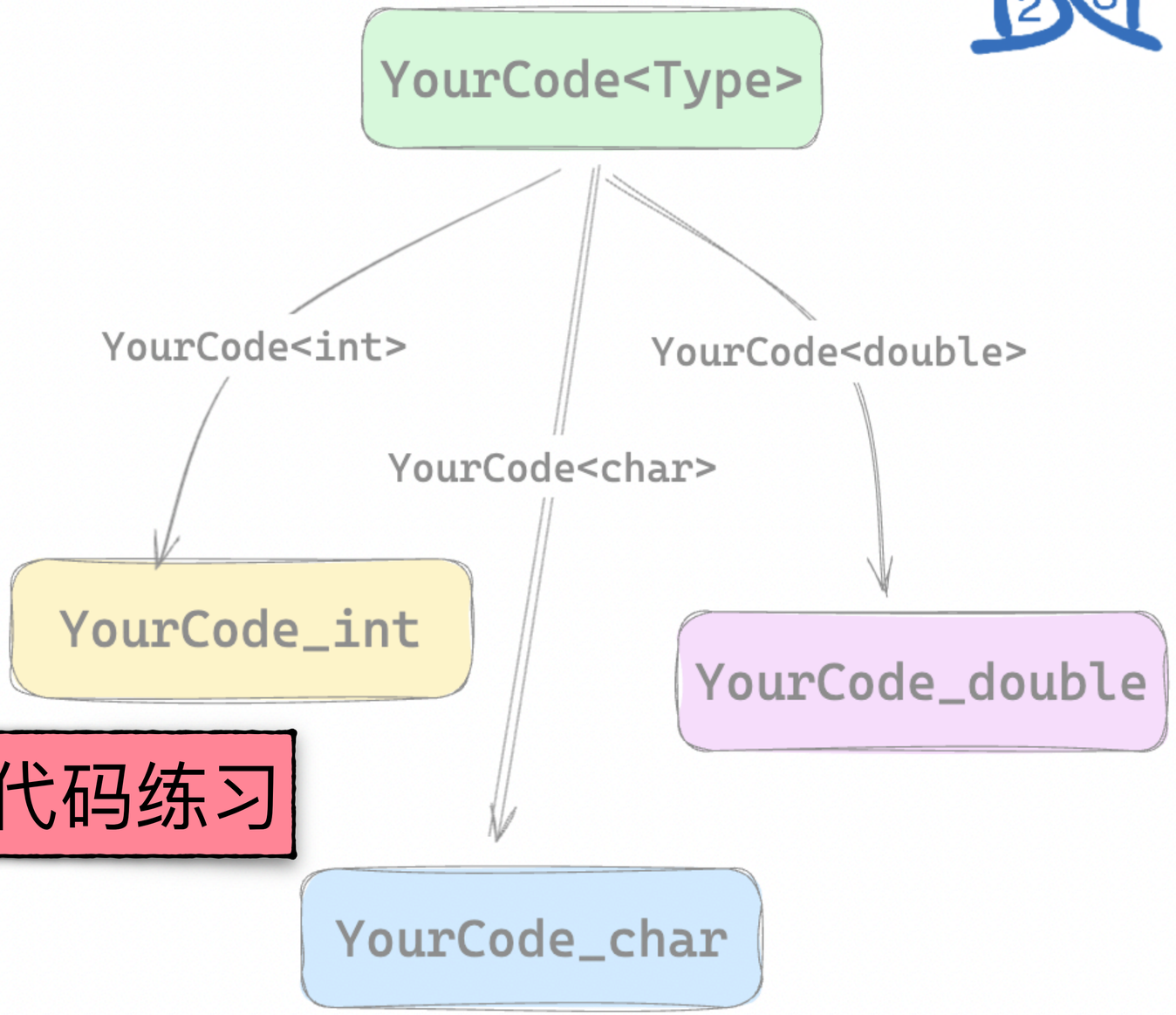
C++ Basics: template

泛型编程初识

@sunrisepeak | 2024.4

预览

- 基础概念
- max - 函数模版
- Box - 类模板
- dslings - max/Box代码练习

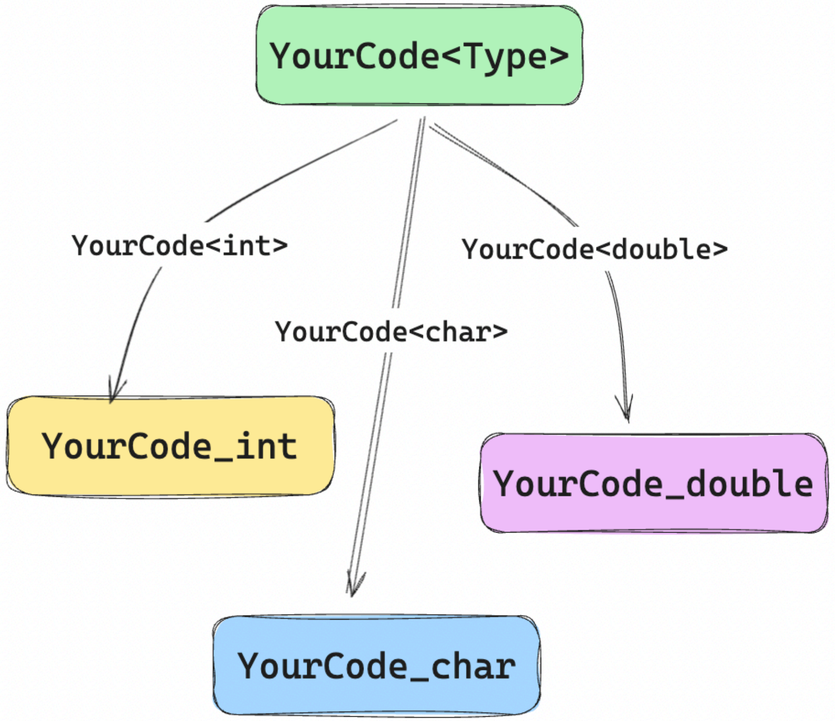


基础概念

- **范型编程**是一种**代码生成技术**，它能帮助我们节省写大量重复代码的时间。

- 实现数据结构的时候，使用**范型编程**技术可以让我们写一套代码就能应用到多种类型的效果

- 要想深度掌握**范型编程**技术不是一个简单的事情，它的难度不亚于学习一门新的语言。 在**d2ds**中我们只涉及其最基础的部分



https://sunrisepeak.github.io/d2ds-courses

函数模版 - max

```cpp
int main() {
    { // int
        int a = -1, b = 1;
        d2ds_assert_eq(d2ds::max(a, b), dstruct::max(a, b));
    }

    { // unsigned int
        unsigned int a = 4294967295, b = 1;
        d2ds_assert_eq(d2ds::max(a, b), dstruct::max(a, b));
    }

    { // double
        double a = 1.3, b = 3.1;
        d2ds_assert_eq(d2ds::max(a, b), dstruct::max(a, b));
    }

    return 0;
}
```

实现max函数

获取a和b两个变量中的最大值

(int | unsigned int | double)

函数模版 - max | 函数重载实现

通过C++的函数**重载技术(overload)**, 我们分别对 `int` | `unsigned int` | `double` 类型版本的max进行实现

```
int max(int a, int b) {
    return a > b ? a : b;
}

unsigned int max(unsigned int a, unsigned int b) {
    return a > b ? a : b;
}

double max(double a, double b) {
    return a > b ? a : b;
}
```

函数模版 - max | 函数模板实现

```
template <typename T>
T max(T a, T b) {
    return a > b ? a : b;
}
```

标识	解释
`template`	模板标识
<>	范形参数类型列表(可以有多个参数)
`typename T`	`typename` 为类型名修饰符, 后面跟着类型名标识 `T`

可看作类型占位符 | 使用方法和正常类型类似 | 模版参数名可自定义

模版 - 编译期实例化

如当只使用 `int` 和 `double` 类型时:

```
d2ds::max(1, 2);
d2ds::max(1.1, 0.8);
```

编译器通过按需进行代码生成来减少代码量, 只会实例化出如下两个版本:

```
int max(int a, int b) {
    return a > b ? a : b;
}

double max(double a, double b) {
    return a > b ? a : b;
}
```

类模版 - Box

实现d2ds::Box用于存储**指定类型**(原生类型和自定义类型)的值

- int类型

解耦 - 数据结构代码与类型无关

```
d2ds::Box<int> box;
box.set_value(2);
d2ds_assert_eq(box.get_value(), 2);
```

- 自定义类型

```
d2ds::Box<dstruct::String> box;
box.set_value("Hello, d2ds!");
d2ds_assert(box.get_value() == dstruct::String("Hello, d2ds!"));
```

Box类模版 - 类型定义

模版标识

模版参数列表

```
template <typename T>
class Box {

};
```

类模板名

Box类模版 - 类型定义

- 使用未知类型T
  - 定义成员变量
  - 值传递 | 引用传递

```
template <typename T>
class Box {
public:
    Box() : __mVal{} { }

    T get_value() const {
        return __mVal;
    }

    void set_value(const T &val) {
        __mVal = val;
    }

private:
    T __mVal;
};
```

总结 - C++泛型编程基础

- 概念 — 编译期代码生成
- 用处 — 数据结构实现和类型解耦
- 示例 — 函数模板max | 类模板Box

dslings - max/Box代码练习

动手写函数模板max和类模板Box

https://sunrisepeak.github.io/d2ds-courses