模板

泛型程序设计

- 泛型程序设计（Generic Programming）是一种算法在实现时不指定具体要操作数据的类型的程序设计方法。所谓“泛型”，指的是算法只要实现一遍，就能适用于多种数据类型。泛型程序设计方法的优势在于能够减少重复代码的编写。
- 泛型程序设计的概念最早出现于 1983 年的 Ada 语言，其最成功的应用就是 C++ 的标准模板库（STL）。也可以说，泛型程序设计就是大量编写模板、使用模板的程序设计。泛型程序设计在 C++ 中的重要性和带来的好处不亚于面向对象的特性。
- 在 C++ 中，模板分为函数模板和类模板两种。

函数模板

- 所谓函数模板，实际上是建立一个通用函数，它所用到的数据的类型（包括返回值类型、形参类型、局部变量类型）可以不具体指定，而是用一个虚拟的类型来代替（实际上是用一个标识符来占位），等发生函数调用时再根据传入的实参来逆推出真正的类型。这个通用函数就称为函数模板（Function Template）。
- 在函数模板中，数据的值和类型都被参数化了，发生函数调用时编译器会根据传入的实参来推演形参的值和类型。换个角度说，函数模板除了支持值的参数化，还支持类型的参数化。
- 一但定义了函数模板，就可以将类型参数用于函数定义和函数声明了。说得直白一点，原来使用 int、float、char 等内置类型的地方，都可以用类型参数来代替。

Thanks

MyArray 封装int类型的数组

MyArray(); 初始化容量100
MyArray(int capacity);
MyArray(const MyArray& arr);

operator=
operator[]
push_back()
getCapacity()
getSiz();

属性:
int * pAddress;
int m_Capacity;
int m_Size;

利用类模板实现数组
MyArray<T>

MyArray(); 初始化容量100
MyArray(int capacity) { this->pAddress = new T[capacity]; }
MyArray(const MyArray& arr);

operator=
operator[]
push_back()
getCapacity()
getSize();

属性:
T * pAddress;
int m_Capacity;
int m_Size;

测试：分别用int数组和Person数组做测试