PyObject

PyObject

刷新

在Python中，类型是动态绑定的。而在C++中所有实例在使用前都需要先声明类型。 那么我们可以在C++中实现类似Python的灵活性吗?

我们尝试构建这样一个类 `PyObject` 。它可以存储任意类型的数据，并且可以在运行时动态的改变数据的类型。在本题中，仅要求支持基本类型 `int` ， `double` , `char` 和自定义类型 `Test` （见下发文件 `test.h` ）

一个很有用的知识是类型转换操作符的重载。如下例中所示。

```
class A {
public:
    operator int() const { return 1; }
};
```

如果这个时候我们有一个 `A` 的实例 `a` ，那么我们可以这样使用它：

```
int b = a;
```

这样的操作会调用 `A` 中的 `operator int()` 方法，将 `a` 转换为 `int` 类型。 这可以帮中我们在实现 `PyObject` 的时候可以实现十分灵活的类型转换。

任务一

对于基本类型 `char` 、 `int` 和 `double` ， `PyObject` 需要实现动态类型绑定，且可以随时获取其值。同时在获得赋值的时候应当输出 `PyObject got a value` 。具体来讲，以下代码应当可以正常运行：

```
  PyObject p;
  p = 1;
  std::cout << (int) p << std::endl;
  p = 'c';
  std::cout << (char) p << std::endl;
  char c = p;
  std::cout << c << std::endl;
  p = 1.0;
  std::cout << (double) p << std::endl;
  int u = 2;
  p = u;
  std::cout << (int) p << std::endl;
```

任务二

我们不应当满足于基本的内建类型，我们还希望它更强大一点，可以存储复杂的类型的数据，在本题中，我们要求 `PyObject` 可以存储 `Test` 类型的数据。

注意：若您不准备完成任务四，那么在 `PyObject` 获得 `Test` 类型的赋值时，需要输出 `PyObject got a value\nBorrowing` ，而不仅仅是 `PyObject got a value`

任务三

对于基本类型（此处仅要求 `int` ， `double` , `char` ）和复杂类型，我们希望 `PyObject` 可以有不同的响应方式。

对于基本类型，我们希望 `PyObject` 在任何情况下复制原值，并且从此不受原值的干扰。 而对于复杂类型 `Test` ，我们希望避免不必要的复制，所以此时 `PyObject` 仅保存其引用。

注意：若您不准备完成任务四，那么在 `PyObject` 获得 `Test` 类型的赋值时，需要输出 `PyObject got a value\nBorrowing` ，而不仅仅是 `PyObject got a value`

任务四

更进一步地，我们希望 `PyObject` 对值负责。我们这里引入**所有权**的概念。

对于基本类型，我们希望 `PyObject` 在**任何情况**下复制原值，并且前后互相不干扰；所有权完全独立。

对于复杂类型 `Test` ，如果赋值方式是：

1. 左值： `PyObject` 不获得所有权，仅保存引用。
2. 右值： `PyObject` 获得所有权，对其**负责**。
3. `PyObject`：如果对方拥有所有权，所有权转移，旧 `PyObject` 丧失所有权，新 `PyObject` 获得所有权；如果对方是引用，则新 `PyObject` 获得对原对象的引用。
4. `PyObject *`：不获得所有权，保存指向原对象的引用。

对于 `PyObject` 向别人赋值：

1. `Test`： 复制，所有权不改变。
2. `Test &`： 返回一份引用，所有权不改变。

所谓负责，指的是 `PyObject` 需要确保在自己生命周期内且被赋其他值以前，该值一定可以访问。 而在 `PyObject` 的生命周期结束或者被赋其他值以后，该值得到了析构。

除此之外， `PyObject` 在获得引用后需要输出 `Borrowing` ，在获得所有权后输出 `Owning` 。

样例输入

见下发的 `main.cpp` 文件。

样例输出

见下发文件

文件下载

点击这里 (/staticdata/1970.6ManNyLoD5BzTeWi.pub/elnM0Wl3FLCTRxXV.download.zip/download.zip)下载，包括 `main.cpp` 、 `Makefile` 、 `test.h` 以及四个样例输出。请注意评测时， `main.cpp` 和测试点会有所不同，但考察的内容类似，因此，如果你正确实现并使用下发的 `main.cpp` 通过了下发的测试点，你也应该能通过评测时的测试点。

提交格式

- **你仅需要提交PyObject.h**。main.cpp、test.h和Makefile由题目提供。

语言和编译选项

#	名称	编译器	额外参数	代码长度限制
0	oop_custom	`make`		1048576 B

递交历史

#	状态	时间
表中没有数据		

递交答案

语言和编译选项 oop_custom

```
1
```

提交

文件请拖入编辑器中，或 上传文件