

Problem A. Build the world

Nahida 正在学习高级程序设计，她十分喜欢通过类型间的组合与继承来构建优美的设计模式，但她始终算不准类的大小，以及函数的继承与覆盖关系。

现在给定一些类和它们之间的关系，你能帮助 Nahida 算出类的大小，以及调用正确的函数吗？

Descriptions

```
class Class{
public:
    void add_inherit_class(const Class*);
    void add_virtual_inherit_class(const Class*);
    void add_member_object(const Class*);

    virtual int get_class_size() const;

    void add_member_function(const std::string& func_name, int (*func_ptr)());
    void add_virtual_member_function(const std::string& func_name, int (*func_ptr)());

    int call_member_function(const std::string& func_name);
};
```

类 (Class) 的基础定义如上。目标编译平台为 x64 (每个指针大小为 8 bytes)。每个 Class 可以进行如下操作：

- 1) add_inherit_class: 公有继承一个其他的 Class。
- 2) add_virtual_inherit_class: 同 1)，但是虚继承。
- 3) add_member_object: 添加一个成员变量，变量类型为传入的 Class
- 4) add_member_function: 添加一个成员函数，给出了函数名及函数指针。
- 5) add_virtual_member_function: 同 4)，但是虚函数。

除了上述操作用于构建类成员及继承关系外，还有以下两个操作用于查询：

- 6) get_class_size: 返回该 Class 实例化对象的大小，单位为 bytes。

对于不可细分的基本类型，ClassWithFixedSize 类简单地重载了此函数并返回固定大小（可参考 ClassWithFixedSize.h，用法见样例#4），且仅用于构建测试用例中的成员变量，不会被其他类继承。这里仅仅为了解释原理，ClassWithFixedSize 类不会过多影响你的实现。

7) call_member_function: 给定一个函数名，实现该类的一个实例调用这个函数的情况，并返回这个函数的返回值（参考样例#9）。

简单起见，我们假设所有的成员函数都直接返回一个固定的 int，不会访问类的成员变量，这样我们就可以方便地传递函数指针。测例中的 4) 和 5) 操作都为以下形式，也可参考样例#9：

```
class1.add_[virtual_]member_function("foobar", []{ return 123; // return some const value });
```

Implement & Submitting

我们致力于减小代码框架的约束。只要不影响 Class.h 中的 7 个测试接口声明（如上文所示，修改会导致链接失败），你可以在 Class.h 和 Class.cpp 中随意进行任何实现，包括添加 Class 类成员等。我们允许使用你掌握的任何标准库，但请注意编译将使用 -std=c++0x。

压缩包中提供了完整的测试框架，除了 main.cpp 中的测例不完整外，其它代码与评测时均相同。提交时，请将 Class.h 和 Class.cpp 两个文件打包为 zip 提交。

Grading

我们提供了 25 组测试数据，每组数据 5 分，你只需要通过 20 组即可获得满分。



Figure 1: Nahida is learning C++

Testcases

我们保证所有 testcases 都是合法的（可以正常通过 MSVC、g++和 clang++编译^[1]），样例的答案符合 MSVC、g++和 clang++的现行标准。

有些编译器为了更好的性能，会对成员变量做内存对齐（比如同时存在 char 和 int 成员时，char 也会被调整为 int 的大小），但本题**不考虑这种行为**。如果有本地验证需求，需要在你用于测试的源代码文件最前面加上“#pragma pack(1)”（不含引号）来禁用对齐行为。

为了更好地帮助你调试，我们明确给出每个测试样例的操作种类。你可以参考 Table 1 推断出错的位置并规划你的得分策略。其中 testcase #2 #4 #9 和#18 已在 main.cpp 中作为样例给出。

（仅对于测例#21~#25）为避免争议，测例中不含有一个类同时虚继承多个类的情况；虚继承和虚函数不会同时存在；如果类 A 虚继承了其他类，那么类 A 及其所有派生类都不会被虚继承。

Tips

1. 接口中的方法是否应该分别实现？类似的实现是否可以共用？具体的区别是什么？
2. 如果你对 Class size 的计算结果有疑问，你可以在本地构造相同的类和继承关系，并使用编译器自带的 sizeof() 函数验证你的答案。
3. 多次提交 OJ 评测不会降低你的分数，建议实现功能后快速检验一下对应的测试点是否通过。**如果你不是很有把握，建议先只考虑前 18 个 testcases 所需的功能。**

Testcase	add inherit class	add_virtual inherit class	add member object	add member function	add_virtual member function	get class size	call member function	comments
#1						✓		empty class
#2	✓					✓		Example testcase
#3			✓			✓		
#4	✓		✓			✓		Example testcase
#5				✓		✓		
#6	✓			✓		✓		
#7			✓	✓		✓		
#8	✓		✓	✓		✓		
#9				✓			✓	Example testcase
#10	✓			✓			✓	
#11				✓		✓	✓	
#12	✓			✓		✓	✓	
#13			✓	✓		✓	✓	
#14	✓		✓	✓		✓	✓	
#15	✓		✓			✓		Multiple inheritance
#16	✓			✓			✓	Multiple inheritance
#17	✓		✓	✓		✓	✓	Multiple inheritance
#18	✓		✓	✓		✓	✓	Example testcase
#19					✓	✓		Simple virtual function
#20	✓				✓	✓	✓	Simple virtual function
#21		✓				✓		Simple virtual inherit
#22	✓	✓	✓			✓		Diamond inherit
#23	✓		✓	✓	✓	✓	✓	
#24	✓		✓	✓	✓	✓	✓	
#25	✓	✓	✓	✓		✓	✓	

Table 1: Summary of testcases

[1] 也就是说，不需要考虑会被编译器拒绝的行为，例如调用未定义的函数，或多继承中的名称冲突。