0. 课堂内容复习演练

课上内容回顾实验

选择题。

1. 利用模板将 strategy 模式改为静态形式

请模仿课上的讲解，将第二天实验中 FileWriter 内部所用的 strategy 模式改为静态形式，其接口变化可能如下，请实验这种形式跟 pImpl 有无冲突，如有该如何解决

Day2

```
template <typename Statictrategy>
struct CFileWriter::FileWriterImpl
{
    StaticStrategy* m_pStrategy;
};
```

2. 利用 CRTP 将 strategy 模式改为静态形式

接口形式变为：

```
template <typename Statictrategy>
struct CFileWriter::FileWriterImpl : public Statictrategy<FileWriterImpl>
{
};
```

进阶要求

在第五天实验工厂方法的基础上将生成的策略变为支持模板的静态形式。

工业级实现

LLVM:

CartesianBenchmarks

经典用法：
CGObjC

```
namespace {
/// A CRTP base class for emitting expressions of retainable object
/// pointer type in ARC.
template <typename Impl, typename Result> class ARCExprEmitter {
protected:
  CodeGenFunction &CGF;
  Impl &asImpl() { return *static_cast<Impl*>(this); }
```

v8 elements

```
// This is an example of the Curiously Recurring Template Pattern (see
// http://en.wikipedia.org/wiki/Curiously_recurring_template_pattern).  We use
// CRTP to guarantee aggressive compile time optimizations (i.e.  inlining and
// specialization of SomeElementsAccessor methods).
template <typename Subclass, typename ElementsTraitsParam>
class ElementsAccessorBase : public InternalElementsAccessor {
 public:
  ElementsAccessorBase() = default;
  ElementsAccessorBase(const ElementsAccessorBase&) = delete;
  ElementsAccessorBase& operator=(const ElementsAccessorBase&) = delete;

  using ElementsTraits = ElementsTraitsParam;
  using BackingStore = typename ElementsTraitsParam::BackingStore;

  static ElementsKind kind() { return ElementsTraits::Kind; }

  static void ValidateContents(JSObject holder, size_t length) {}

  static void ValidateImpl(JSObject holder) {
    FixedArrayBase fixed_array_base = holder.elements();
    if (!fixed_array_base.IsHeapObject()) return;
    // Arrays that have been shifted in place can't be verified.
    if (fixed_array_base.IsFreeSpaceOrFiller()) return;
    size_t length = 0;
    if (holder.IsJSArray()) {
      Object length_obj = JSArray::cast(holder).length();
      if (length_obj.IsSmi()) {
        length = Smi::ToInt(length_obj);
      }
    } else if (holder.IsJSTypedArray()) {
      length = JSTypedArray::cast(holder).length();
    } else {
      length = fixed_array_base.length();
    }
    Subclass::ValidateContents(holder, length);
  }

  void Validate(JSObject holder) final {
    DisallowGarbageCollection no_gc;
    Subclass::ValidateImpl(holder);
  }
```