实验课 3 模版类与比特引用

背景

首先，在本节课学习模版类后，你可以将你的 bits 类修改为一个模版类，其模版参数即为比特的个数。

对于一个比特流，能不能像数组一样对比特流中的比特执行操作呢🤔

也就是说，我们需要能够访问比特流中每个位置的值以及对每个位置的值执行修改操作。

我们可以尝试自己实现一个针对比特的引用类，并对其进行包装，从而做到获得一个位置的值和修改一个位置的值。

实验描述

概述

你需要实现一个模版类 `bit_reference` ，其模版参数为其指向的比特存储于什么类中。

其需要两个值来描述其指向的位置，分别是一个指针，用于存储指向的位置具体是什么，和一个掩码，其有且仅有一位为 1 ，用于描述其指向的是这个值的哪一位。

```
template<class origin>
class bit_reference {
  using storage_type = typename origin::storage_type;
  using storage_pointer = typename origin::storage_pointer;
  storage_pointer seg;
  storage_type mask;
};
```

基础功能

- 比特引用应当具有一些构造函数。思考并实现它的不同类型的构造函数，并且想想每种构造函数应当是 `private` 还是 `public` 的。
- 显然，引用类需要和 `bool` 类型存在隐式转换的方法。这是因为任何时候都可以将比特引用当一个比特使用。

  ```
  operator bool() const { /* ... */ }
  ```

- 作为比特引用，它需要一个取反操作符。

  ```
  bool operator ~() const { /* ... */ }
  ```

- 比特引用的 `=` 操作符应当有两种实现，分别是等于一个 `bool` 类型和等于一个 `bit_reference` 。

  尝试思考：

  - 等于的返回值应当是什么？
  - 这里可以让 `=` 针对 `bit_reference` 类型默认生成为复制操作吗？
- 为比特引用实现一个 `flip` 操作，用于翻转其指向的位。

现在你已经实现了一个基础的 `bit_reference` 类，是时候把它应用于你的 `bits` 类中了。

- 在你的 `bits` 类中实现一个 `make_ref( size_t pos )` 用于生成一个比特位的引用，返回一个 `bit_reference` 类型。

通过你实现的 `make_ref` 操作，实现以下几种基本操作：

- `operator []( size_t p )` 用于返回一个位的比特引用。
- `flip( size_t p )` 用于翻转一个位置的值。
- `set( size_t p )` 用于将一个位的值设为 1 。
- `reset( size_t p )` 用于将一个位的值设为 0 。

尝试利用 `bit_reference` 重写 `to_string` 函数。**注意**，`to_string` 函数不应当去掉 `const` 属性。

如果你遇到了问题，先尝试阅读实验指北或上网查询资料，如果还是无法解决可以询问助教。

实验指北和提示

- 思考 `bit_reference` 类、`bits` 类之间的友元关系，并用 `friend` 关键字定义。
- 思考上一次课提到的 `const` 相关的问题。有时你会想要在常成员函数中使用引用，该怎么处理？

  提示：再实现一个 `bit_const_reference` 用于处理所有常量的情况，并针对它实现 `bit_const_reference make_ref( size_t p ) const` 。它在 `bit_reference` 的基础上应当减少一些功能。