Оператор приведения к типу

После того, как мы сделали внутренний класс `IndexProxy`, у которого есть `operator=`, казалось бы, всё должно быть хорошо. Однако ваш код вряд ли будет компилироваться, например, при выполнении такого теста:

```
Polynomial<int> poly;
// Здесь не компилируется, потому что мы сравниваем IndexProxy и int
ASSERT_EQUAL(poly[0], 1);
// Тоже не компилируется — мы присваиваем IndexProxy в int
int x = poly[1];
```

Нам нужно сделать так, чтобы код выше работал. Для успешной компиляции и работы ассерта можно перегрузить `operator==` для `IndexProxy` и `T` (`T` -- тип, с которым мы инстанцируем шаблон `Polynomial`). Кроме того, нам придётся перегрузить `operator<<` для IndexProxy, чтобы он умел выводиться в поток.

Это будет работать, однако команда `int x = poly[1];` все равно не будет компилироваться. Универсальным решением здесь будет добавить в класс `IndexProxy` оператор приведения к типу `T`. Это довольно экзотическая возможность C++, о которой мы не рассказывали в лекциях. Поэтому мы просто приведём его реализацию:

```
template<typename T>
class Polynomial {
private:
  ...
  class IndexProxy {
  public:
    IndexProxy(Polynomial& poly, size_t degree);

    operator T() const {
      // Вызываем константную версию Polynomial::operator[]
      return std::as_const(poly_)[degree_];
    }

  private:
    Polynomial& poly_;
    size_t degree_;
  };
  ...
};
```

›