9. Не пессимизируйте преждевременно

Резюме

То, что просто для вас, — просто и для кода. При прочих равных условиях, в особенности — сложности и удобочитаемости кода, ряд эффективных шаблонов проектирования и идиом кодирования должны естественным образом "стекать с кончиков ваших пальцев" и быть не сложнее в написании, чем их пессимизированные альтернативы. Это не преждевременная оптимизация, а избежание излишней пессимизации.

Обсуждение

Избежание преждевременной оптимизации не влечет за собой снижения эффективности. Под преждевременной пессимизацией мы подразумеваем написание таких неоправданных потенциально неэффективных вещей, как перечисленные ниже.

- Передача параметров по значению там, где применима передача параметров по ссылке (рекомендацня 25).
- Использование постфиксной версии ++ там, где с тем же успехом можно воспользоваться префиксной версией (рекомендация 28).
- Использование присваивания в конструкторах вместо списка инициализации (рекомендация 48).

Не является преждевременной оптимизацией снижение количества фиктивных временных копий объектов, в особенности во внутренних циклах, если это не приводит к усложнению кода. В рекомендации 18 поощряется максимально локальное объявление переменных, но там же приведено и описание исключения — возможный вынос переменных из цикла. В большинстве случаев такое действие не усложняет понимание предназначения кода, более того, может помочь пояснить, что именно делается в цикле и какие вычисления являются его инварнантами. Конечно же, предпочтительно использовать готовые алгоритмы вместо написания явных циклов (рекомендация 84).

Два важных способа усовершенствования программы, которые делают ее одновременно и яснее, и эффективнее — это использование абстракций (см. рекомендации 11 и 36) и библиотек (рекомендация 84). Например, использование `vector`, `list`, `map`, `find`, `sort` и других возможностей стандартной библиотеки, стандартизированных и реализованных экспертами мирового класса, не только делают ваш код яснее и легче понимаемым, но зачастую и более быстрым.

Избежание преждевременной пессимизации становится особенно важным, когда вы пишете библиотеку. При этом вы обычно не знаете контекста использования вашей библиотеки, а поэтому должны суметь сбалансировать эффективность и возможность повторного использования. Не забывайте уроков рекомендации 7 — следует куда больше заботиться о масштабируемости, чем о выигрыше пары тактов процессора.

Ссылки

[Keffer95] pp.12-13 • [Stroustrup00] §6 introduction • [Sutter00] §6