УДК 536.24

ПРОГРАМА PVS-STUDIO ДЛЯ ПОШУКУ ПОМИЛОК У ПРОГРАМНОМУ КОДІ НА МОВІ С++

Біляєва В. А.
Харківський національний автомобільно-дорожній університет

Статический анализ кода это процесс выявления ошибок и недочетов в исходном коде программ. Статический анализ можно рассматривать как автоматизированный процесс обзора кода. Остановимся на обзоре кода чуть подробнее.

Обзор кода (code review) – один из самых старых и надежных методов выявления дефектов. Он заключается в совместном внимательном чтении исходного кода и высказывании рекомендаций по его улучшению. В процессе чтения кода выявляются ошибки или участки кода, которые могут стать ошибочными в будущем. Также считается, что автор кода во время обзора не должен давать объяснений, как работает та или иная часть программы. Алгоритм работы должен быть понятен непосредственно из текста программы и комментариев. Если это условие не выполняется, то код должен быть доработан.

Как правило, обзор кода хорошо работает, так как программисты намного легче замечают ошибки в чужом коде. Более подробно с методикой обзора кода можно познакомиться в книге Стива Макконнелла "Совершенный код" (Steve McConnell, "Code Complete") [1].

Единственный существенный недостаток методологии совместного обзора кода – это крайне высокая цена. Необходимо регулярно собирать нескольких программистов для обзора нового кода или повторного обзора кода после внесения рекомендаций. При этом программисты должны регулярно делать перерывы для отдыха. Если пытаться просматривать сразу большие фрагменты кода, то внимание быстро притупляется и польза от обзора кода быстро сходит на нет.

Получается, что с одной стороны хочется регулярно осуществлять обзор кода, а с другой - это слишком дорого. Компромиссным

решением являются инструменты статического анализа кода. Они без устали обрабатывают исходные тексты программ и выдают программисту рекомендации обратить повышенное внимание на определенные участки кода.

Конечно, программа не заменит полноценного обзора кода, выполняемого коллективом программистов. Однако соотношение польза/цена делает использование статического анализа весьма полезной практикой, применяемой многими компаниями.

Задачи, решаемые программами статического анализа кода можно разделить на 3 категории:

- Выявление ошибок в программах. Подробнее про это будет рассказано ниже.
- Рекомендации по оформлению кода. Некоторые статические анализаторы позволяют проверять, соответствует ли исходный код, принятому в компании стандарту оформления кода. Имеется в виду контроль количества отступов в различных конструкциях, использование пробелов/символов табуляции и так далее.
- Подсчет метрик. Метрика программного обеспечения - это мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций. Существует большое количество разнообразных метрик, которые можно подсчитать, используя те ли иные инструменты.

Есть и другие способы использования инструментов статического анализа кода. Например, статический анализ можно использовать как метод контроля и обучения новых сотрудников, еще недостаточно знакомых с правилами программирования в компании.

Стати́ческий ана́лиз ко́да (англ. *static code analysis*) - анализ программного обеспечения, производимый (в отличие от динамического анализа) без реального выполнения исследуемых программ. В большинстве случаев анализ производится над какой-либо версией исходного кода, хотя иногда анализу подвергается определенный вид объектного кода, например P-код или код на MSIL. Термин обычно применяют к анализу, производимому специальным программным обеспечением (ПО), тогда как ручной анализ называют «program understanding», «program comprehension» (*пониманием* или *постижением* программы).

В зависимости от используемого инструмента глубина анализа может варьироваться от определения поведения отдельных операторов до анализа, включающего весь имеющийся исходный код. Способы использования полученной в ходе анализа информации также различны - от выявления мест, возможно содержащих ошибки (утилиты типа Lint), до формальных методов, позволяющих математически доказать какие-либо свойства программы (например, соответствие поведения спецификации).

Некоторые люди считают программные метрики и обратное проектирование формами статического анализа. Получение метрик (англ. *software quality objectives*) и статический анализ часто совмещаются, особенно при создании встраиваемых систем.

В последнее время статический анализ всё больше используется в верификации свойств ПО, используемого в компьютерных системах высокой надёжности, особенно критичных для жизни (англ. safety-critical). Также применяется для поиска кода, потенциально содержащего уязвимости (иногда это применение называется *Static Application Security Testing*, SAST).

Статический анализ постоянно применяется для критического ПО в следующих областях:

- ПО для медицинских устройств;
- ПО для атомных станций и систем защиты реактора (Reactor Protection Systems);
- ПО для авиации (в комбинации с динамическим анализом);
- ПО на автомобильном или железнодорожном транспорте.

Принципы статического анализа

Большинство компиляторов (например, GNU C Compiler) выводят на экран «предупреждения» (англ. *warnings*) – сообщения о том, что код, будучи синтаксически правильным, скорее всего, содержит ошибку. Например:

```
int x;
int y = x+2;    // Переменная x не инициализирована!
```

Это простейший статический анализ. У компилятора есть много других немаловажных характеристик — в первую очередь скорость работы и качество машинного кода, поэтому компиляторы проверяют

код лишь на очевидные ошибки. Статические анализаторы предназначены для более детального исследования кода.

Достоинства и недостатки статического анализа

Главное преимущество статического анализ состоит в возможности существенной снижении стоимости устранения дефектов в программе. Чем раньше ошибка выявлена, тем меньше стоимость ее исправления. Инструменты статического анализа позволяют выявить большое количество ошибок этапа конструирования, что существенно снижает стоимость разработки всего проекта. Например, статический анализатор кода PVS-Studio может запускаться в фоновом режиме сразу после компиляции и в случае нахождения потенциальной ошибки уведомит программиста (см. режим инкрементального анализа).

Другие преимущества статического анализа кода:

1. Полное покрытие кода. Статические анализаторы проверяют даже те фрагменты кода, которые получают управление крайне редко. Такие участки кода, как правило, не удается протестировать другими методами. Это позволяет находить дефекты в обработчиках редких ситуаций, в обработчиках ошибок или в системе логирования.

2. Статический анализ не зависит от используемого компилятора и среды, в которой будет выполняться скомпилированная программа. Это позволяет находить скрытые ошибки, которые могут проявить себя только через несколько лет. Например, это ошибки неопределенного поведения.

Такие ошибки могут проявить себя при смене версии компилятора или при использовании других ключей для оптимизации кода. Другой интересный пример скрытых ошибок приводится в статье "Перезаписывать память - зачем?".

3. Можно легко и быстро обнаруживать опечатки и последствия использования Copy-Paste. Как правило, нахождение этих ошибок другими способами является кране неэффективной тратой времени и усилий. Обидно после часа отладки обнаружить, что ошибка заключается в выражении вида "strcmp(A, A)". Обсуждая типовые ошибки, про такие ляпы, как правило, не вспоминают. Но на практике на их выявление тратится существенное время.

Недостатки статического анализа кода:

1. Статический анализ, как правило, слаб в диагностике утечек памяти и параллельных ошибок. Чтобы выявлять подобные ошибки, фактически необходимо виртуально выполнить часть программы. Это крайне сложно реализовать. Также подобные алгоритмы требуют очень много памяти и процессорного времени. Как правило, статические анализаторы ограничиваются диагностикой простых случаев. Более эффективным способом выявления утечек памяти и параллельных ошибок является использование инструментов динамического анализа.

2. Программа статического анализа предупреждает о подозрительных местах. Это значит, что на самом деле код, может быть совершенно корректен. Это называется ложно-позитивными срабатываниями. Понять, указывает анализатор на ошибку или выдал ложное срабатывание, может только программист. Необходимость просматривать ложные срабатывания отнимает рабочее время и ослабляет внимание к тем участкам кода, где в действительности содержатся ошибки.

Ошибки, обнаруживаемые статическими анализаторами весьма разнообразны. Вот, например, список диагностик, которые реализованы в инструменте PVS-Studio. Некоторые анализаторы специализируются на определенной области или типах дефектов. Другие, поддерживают определенные стандарты кодирование, например MISRA-C:1998, MISRA-C:2004, Sutter-Alexandrescu Rules, Meyers-Klaus Rules и так далее.

Область статического анализа активно развивается, появляются новые диагностические правила и стандарты, некоторые правила устаревают. Поэтому нет смысла пытаться сравнить анализаторы, на основании списков обнаруживаемых дефектов. Единственный способ сравнить инструменты, это проверить с их помощью набор проектов и посчитать найденных ими количество настоящих ошибок.

Література

1. Макконнелл "Совершенный Код", 2005.
2. https://www.viva64.com/ru/t/0046/.
3. https://www.viva64.com/ru/pvs-studio/ - сайт PVS-Studio.