УДК 004.054

Автоматический анализ стиля написания программ

Д .А. Подлесных, А. В. Третьяков, Н. Д. Уваров
Московский физико-технический институт (государственный университет)

Для эффективной разработки программного обеспечения, особенно в коллективе авторов, написанная на языке высокого уровня программа должна быть не только правильной и быстрой, но и удобной для чтения и понимания. Как правило, для этого используются правила расстановки отступов, скобок, переводов строки и именования переменных. Соответствие имён переменных их смыслу проверить сложно, но проверку формальных правил можно автоматизировать. Это сэкономит время высокооплачиваемых программистов, производящих Code Review, за счёт отсутствия фрагментов кода, не удовлетворяющих формальным критериям. Такая утилита может не только проверять соответствие стилю кода, принятому в организации, но и автоматически приводить программу к нему. Её можно интегрировть в систему контроля версий или автоматического тестирования.

В данной работе рассматривается три подхода к анализу стиля.

Первый (простейший) подход к адаптивной проверке стиля форматирования состоит в запуске нескольких проверяющих программ (style checker-ов) и поиске наилучшего совпадения. В качестве проверяющей стиль программы можно использовать посимвольное сравнение (diff) с результатом работы утилиты-форматировщика на исходном тексте: если исходник в точности соответствует стилю форматировщика, он не будет изменён. Наиболее известным примером такой утилиты является indent.

При этом неподдерживаемыми остаются всевозможные смешанные стили, которые, тем не менее, возможно получить на выходе форматировщика если задать подходящий набор опций командной строки. Перебор всех возможных наборов опций уже не представляется возможным, поскольку число наборов растёт экспоненциально с ростом числа опций. Существует второй подход:подбирать лучший (для данной проверяемой программы) набор постепенно, по одной опции, начиная с наиболее влиятельных. На каждом шаге из возможных значений очередной опций выбирается наилучшее всмысле числа расхождений между отформатированной программой и исходной. Описанный подход был реализован с использование форматировщика C++ кода AStyle и C кода GNU indent, и в первом приближении он позволяет достаточно легко (в смысле сложности проверяющей программы) поддержать большое количество различных стилей. Если программа-форматировщик не поддерживает какой-либо стиль оформления, или же не поддерживает какие-то

синтаксические конструкции, нужно дополнительно выделить их из исходного текста и обработать отдельно.

При наличии кода, обрабатывающего фрагменты программы, фактически, разбором случаев, наличие дополнительного модуля, перебирающего стили, кажется избыточным. Поэтому второй подход к анализу стиля, рассматриваемый в данной работе, фактически совмещает в себе алгоритмы анализатора стиля и форматировщика.

Третий подход реализует программа "Formatter" / "Code Style Checker", разработанная ассистентом кафедры информатики и вычислительной математики МФТИ А. В. Третьяковым, анализирует данный ей на вход файл исходной программы (например, студенческой домашней задачи), проводит лексический разбор, вычисляет начало и конец условных операторов, циклов и других конструкций языка Си и на основе этого вычисляет правильное количество отступов, необходимых перед каждой строчкой кода, а также определяет, где должны быть дополнительные пробелы (например, после запятых, до и после знаков бинарных операций), в соответствии с правилами оформления программ (Coding Style Guide), который можно посмотреть здесь: http://kpm8.mipt.ru:8208/CodeStyle.html. В результате анализа, в зависимостиот указанного режима работы, выводится либо список мест, в которых предположительно допущены стилистические ошибки, либо отформатированный вариант программы. Style Checker соблюдает определённое соглашение о формате вывода и коде возврата, что позволяет использовать его в автоматизированной системе проверки решений Ejudge. Этот сценарий использования был успешно опробован на семинарских занятиях и контрольных работах на кафедре информатики МФТИ, в 2014/2015 и 2015/2016 учебных годах. Если студент отправляет в систему программу с плохим оформлением (количество стилевых ошибок, вернее сумма штрафных баллов, зависящих ещё и от критичности ошибок, превышает порог), то она даже не компилируется (хотя тут возможны варианты) и не запускается на исполнение, а сразу отклоняется. Пользователь в протоколе может посмотреть вывод Style Checker'а и быстро понять свои ошибки оформления, точно так же, как и ошибки компиляции и ошибки выполнения. Style Checker автоматически определяет выбранный стиль по первому факту использования знака табуляции либо группы пробелов, а также открывающей фигурной скобки в начале программы и в дальнейшем оценивает остаток программы в соответствии с этим стилем. Поддерживаются несколько распространённых стилей оформления.