Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б10-мм

Интеграция clang-tidy в Eclipse

СОТНИКОВ Илья Александрович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ст. преподаватель кафедры ИАС, Смирнов К. К.

Консультант:
старший программист-разработчик, ООО «СофтКом», Бабанов П. А.

Санкт-Петербург
2025

Оглавление

Введение

В современном мире использование IDE (Integrated Development Environment) с целью повышения удобства и ускорения разработки программного обеспечения стало привычной и широко использующейся практикой. IDE обрели популярность благодаря своим полезным инструментам для написания, отладки и тестирования кода. Платформа Eclipse [7], обладая открытой архитектурой и широкими возможностями для расширения функциональности, привлекает внимание программистов, стремящихся вести эффективную разработку на многих языках, в частности на C/C++. Однако, платформа Eclipse CDT (C/C++ Development Tooling) [4], ставшая стандартом для C/C++ разработки в Eclipse, имеет слишком подробную и сложную настройку, поскольку поддерживает почти все существующие компиляторы.

ООО «Софтком» разрабатывает своего рода альтернативу для CDT на базе Eclipse, использующую определенные технологии. У заказчика есть претензии к перегруженности интерфейса, наличию излишней функциональности, сложности реализации дополнительной бизнес-логики в IDE на базе CDT, поэтому продукт компании CLDT (Clang Development Tool) должен стать более компактным по сравнению с CDT, но при этом обладать широкой функциональностью. Одним из принципов является повсеместное использование пользовательского графического интерфейса в целях облегчения взаимодействия с IDE. Очередным этапом разработки стала интеграция такого инструмента, как clang-tidy [2], который способен значительно облегчить процесс поиска ошибок и нарушений стиля в коде. Этот инструмент не только помогает выявлять потенциальные проблемы, но и предлагает рекомендации по их устранению, что делает его очень полезным помощником для разработчиков. Тем не менее, при работе с ним возникают трудности из-за отсутствия интуитивно понятного пользовательского интерфейса и сложной настройки.

В рамках данной работы главной целью является расширение функциональности IDE Eclipse для языков C/C++ путем интеграции инстру-

мента clang-tidy и разработки графического пользовательского интерфейса для него.

Для решения поставленной задачи был принят поэтапный подход, включающий анализ существующих решений, разработку плагина для интеграции clang-tidy в среду Eclipse, а также тестирование его работоспособности в составе IDE. Таким образом, работа направлена на внедрение в IDE инструмента, который поможет разработчикам повысить качество кода и упростить процесс его написания и отладки.

1. Постановка задачи

Целью работы является интеграция утилиты clang-tidy в IDE для C/C++ на базе Eclipse. Для её выполнения были поставлены следующие задачи:

1. Спроектировать архитектуру плагина для интеграции clang-tidy.
2. Реализовать получение и хранение параметров запуска clang-tidy через пользовательский интерфейс.
3. Реализовать запуск clang-tidy.
4. Реализовать обработку вывода clang-tidy.

2. Обзор

2.1. Eclipse

Eclipse — это мощная открытая платформа для разработки, которая может быть использована не только как IDE для многих языков программирования, но и как фреймворк для написания своих собственных решений. Особенностью Eclipse является модульность. То есть в основе платформы лежит ядро, загружающее плагины, а вокруг него находится множество плагинов, что позволяет пользователям настраивать среду под свои нужды [5]. В рамках этой работы Eclipse будет использоваться как фреймворк для IDE.

Плагин для Eclipse может зависеть от ядра или от другого плагина и имеет следующую структуру:

- Java-классы, находятся внутри JAR файла [10].
- Ресурсы плагина (например, иконки), так же хранятся в JAR файле.
- META-INF/MANIFEST.MF — файл, описывающий версию плагина, идентификатор, зависимости.
- plugin.xml — файл, описывающий расширения и точки расширения.

Каждый плагин для Eclipse представляет собой JAR-файл со специфичным именем и находится внутри определённой директории Eclipse. Имя файла состоит из идентификатора плагина, подчёркивания и версии плагина.

2.2. Обзор используемых технологий

Eclipse IDE for RCP and RAP Developers — полный набор инструментов для разработчиков, желающих создавать плагины для Eclipse. Включает в себя Eclipse Plug-in Development Environment [8],

который предоставляет инструменты для создания, разработки, тестирования, отладки, сборки и развертывания плагинов Eclipse.

JDK (Java Development Kit) — набор инструментов для разработки программного обеспечения на языке программирования Java, включающий вспомогательные библиотеки, объединённые с Java Runtime Environment (JRE) и Java Virtual Machine (JVM).

2.3. Существующее решение

CodeChecker Eclipse Plugin [3] — плагин для среды разработки Eclipse, предназначенный для отображения результатов статического анализа кода на C и C++, выполненного с помощью clang static analyzer [1] и clang-tidy. Для его использования необходим Eclipse CDT. Имеет свой графический пользовательский интерфейс, который упрощает настройку и работу с анализаторами.

2.4. Выводы

Таким образом, уже существующий плагин не может использоваться для достижения поставленной цели, так как целью работы является разработка окружения, отдельного от CDT, в то время как CodeChecker расширяет Eclipse CDT.

3. Реализация

В данном разделе будет рассмотрена архитектура плагина и классы с методами, которые были реализованы в процессе интеграции clang-tidy в IDE на базе Eclipse.

3.1. Архитектура плагина

Плагин для интеграции clang-tidy разбивается на 5 компонентов, как показано на рисунке 1, и имеет две основные части: одна отвечает за UI, другая — за запуск анализа и обработку результатов.

- Настройка параметров запуска clang-tidy через UI
  ClangTidyPropertyPage расширяет AbstractPropertyPage. Класс отвечает за внешний вид, функциональность страницы настроек и обеспечивает получение и сохранение параметров запуска для clang-tidy.

- Запуск анализа clang-tidy и генерация маркеров
  Класс ClangTidyAnalysisHandler наследуется от AnalysisHandler. Он используется для представления команды — абстрактного действия, которое будет вызываться через меню запуска анализа. Класс обеспечивает поиск проекта, папки или файла среди выбранных пользователем объектов, создаёт объект ClangTidyAnalysisTool и запускает его.

  ClangTidyAnalysisTool реализует интерфейс IAnalysisTool и наследуется от класса Job. Класс выполняет поиск всех исходных файлов в подкаталогах проекта или директории, исключая папку сборки. Далее происходит создание консоли вывода и запуск анализа clang-tidy с помощью ClangTidyOperation.

  Компонент ClangTidyOperation реализует ICoreRunnable. Его задачами являются проверка существования базы данных команд компиляции, формирование строки параметров запуска clang-tidy

и запуск процесса анализа. Результат анализа выводится в консоль.

За обработку вывода clang-tidy отвечает ClangTidyOutputParser. Во время анализа генерируются маркеры ошибок, предупреждений или замечаний в нужном месте исходного кода.

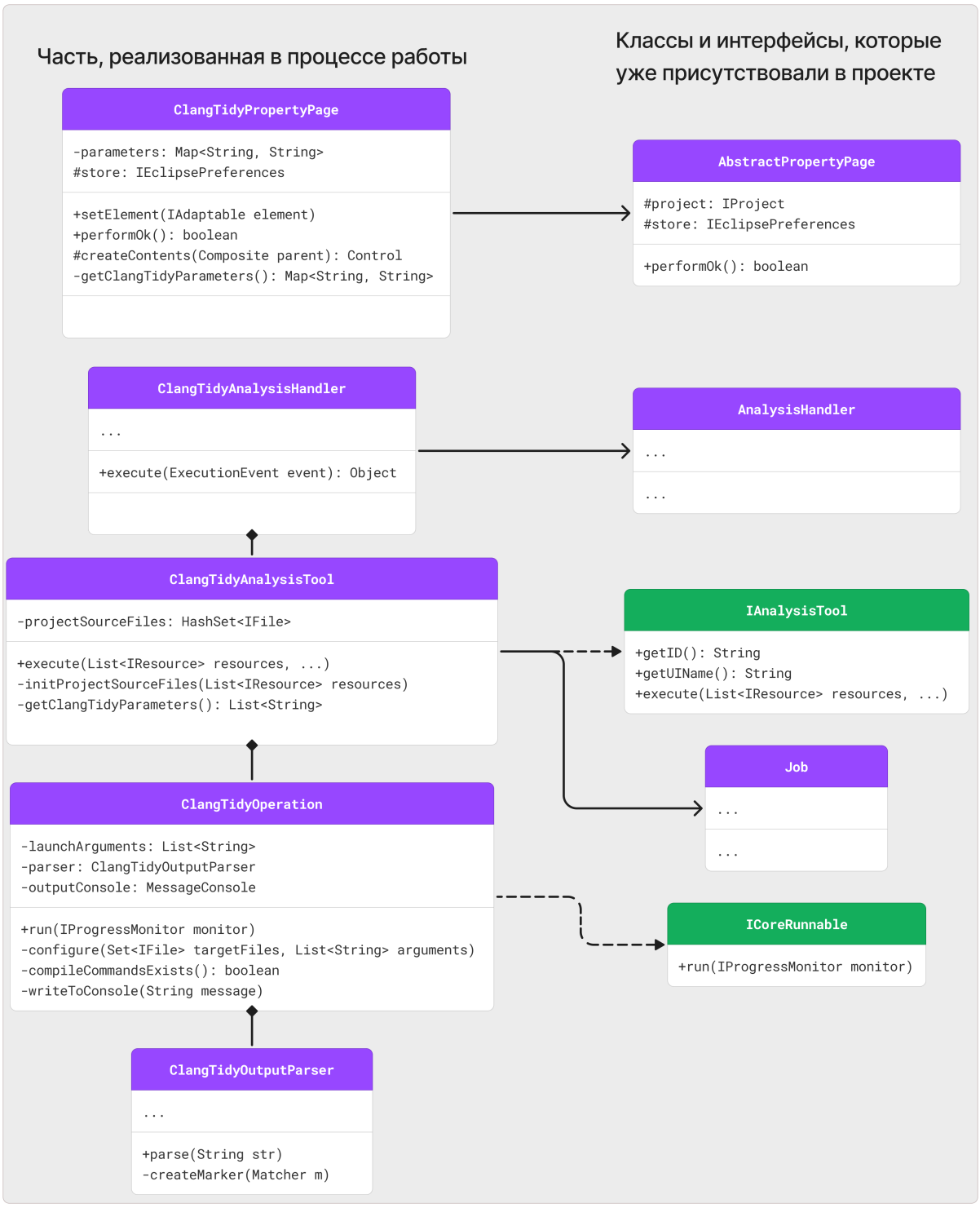

Рис. 1: Архитектура плагина.

3.2. Получение и хранение параметров запуска clang-tidy

Для того чтобы пользователь смог передать параметры для clang-tidy, был создан раздел в меню свойств проекта «Настройки clang-tidy». С целью реализации страницы настроек был написан класс ClangTidyPropertyPage. Для взаимодействия пользователя с разделом была добавлена таблица «Название параметра — Значение параметра», используя стандартные наборы инструментов SWT [6] и JFace [9] (Рис. 2).

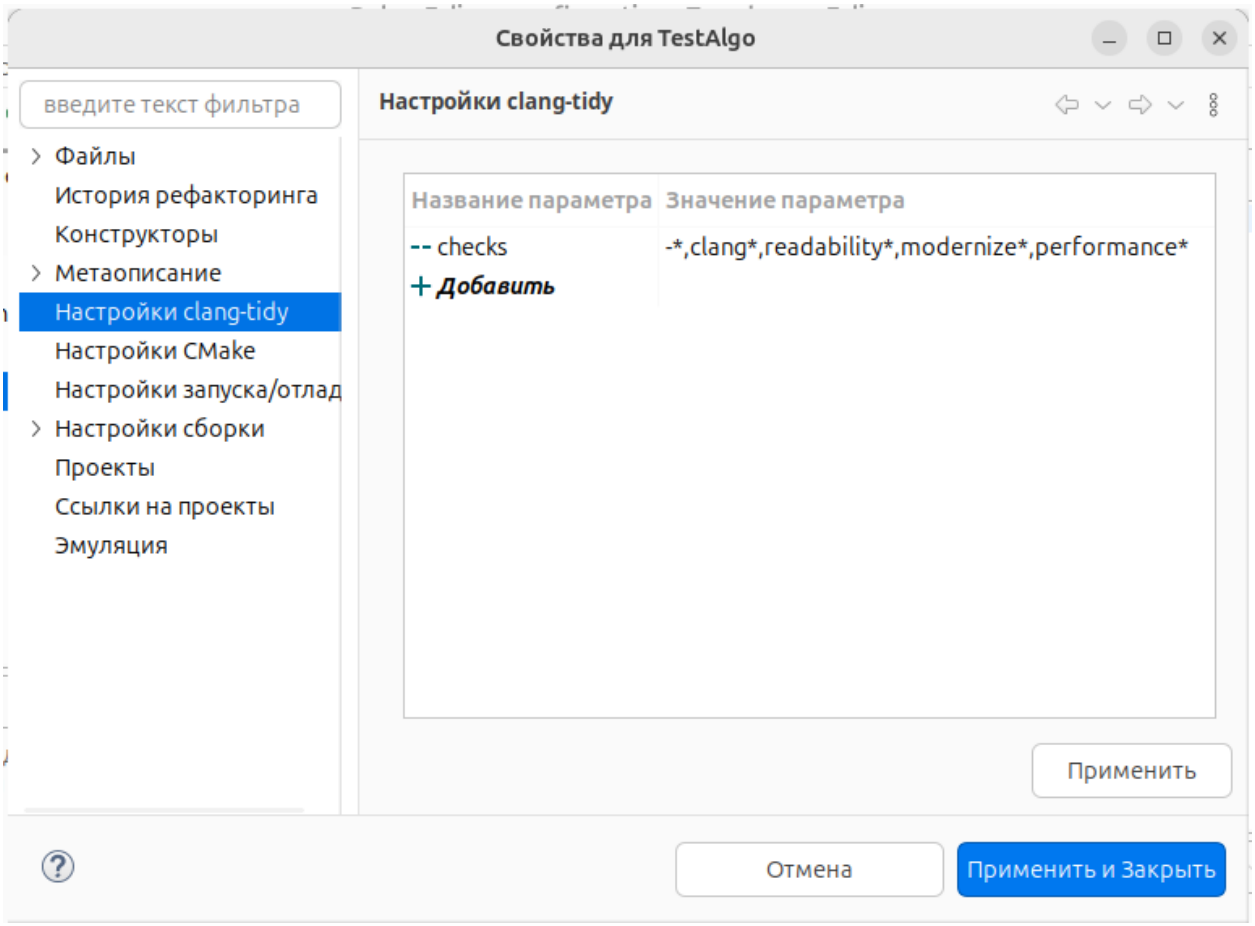

Рис. 2: Окно настроек clang-tidy.

Для того чтобы пользователь видел, какие параметры сохранены в текущий момент и мог добавлять или изменять их, было создано поле parameters, представляющее собой словарь, где ключом и значением являются строки. Взаимодействие с таблицей происходит без от-

дельных кнопок. Элементы пользовательского интерфейса, такие как специальная ячейка таблицы «Добавить» и маркеры для ячеек параметров, были добавлены, используя уже имеющиеся в проекте классы AddElementContentProvider и AddElementLabelProvider. В столбец «Значение параметра» записываются значения из словаря parameters. Добавление нового параметра реализовано следующим образом: пользователь вводит на место текста «Добавить» новый параметр, в таблице срабатывает обработчик редактирования, который распознает нажатие на клетку «Добавить» и создаёт в словаре новую запись с введённым ключом. Далее ввод автоматически переключится на столбец значений. В процессе разработки такого поведения возникли некоторые трудности. В частности, обработчик редактирования столбца значений никогда не вызывался. Для решения данной задачи были произведены попытки применения нескольких подходов, изменяющих поведение таблицы или типов значений словаря parameters, но они не увенчались успехом. Решением стало написание метода createValueText, который создаёт поверх таблицы в нужном месте текстовое поле и выполняет необходимую логику. Редактирование параметров и их значений обеспечивается за счёт EditingSupport для каждого столбца. Для удаления параметра необходимо стереть его имя, что вызовет обработчик редактирования, который увидит, что новое значение пустое, и выполнит удаление из parameters.

Хранение параметров можно было реализовать двумя способами. Один из них — записать параметры запуска в конфиг файл .clang-tidy. Этот подход имеет ряд недостатков. Так как необходимо хранить и отображать настройки в UI, придется делать свой сериализатор-десериализатор для этого файла, помимо этого решать проблемы синхронного доступа к файлу, безопасного чтения-записи и другие. Таким образом, для хранения параметров был выбран другой подход, подразумевающий реализацию с помощью Eclipse Preferences. Фреймворк настроек позволяет работать с параметрами, не задумываясь о всех вышеописанных вопросах. Eclipse Preferences содержит в себе дерево с узлами, что упрощает его использование для хранения индивидуальных

настроек для разных частей проекта, создавая новые узлы для них. Параметры clang-tidy хранятся отдельно от всех остальных настроек проекта в соответствующем разделе. Кроме того, опции запуска clang-tidy задаются индивидуально для каждого проекта.

Метод getClangTidyParameters считывает значения из переменной store, узла настроек с опциями clang-tidy, и записывает их в поле parameters. Пользователь может оставить некоторые параметры с пустым значением, поэтому был написан метод валидации validatePage. После каждого взаимодействия пользователя с таблицей вызывается метод валидации, который в случае отсутствия необходимых значений оповещает пользователя и запрещает сохранять настройки. При нажатии кнопки «Применить» вызывается метод performOk, в котором значения из parameters записываются в store.

3.3. Запуск clang-tidy

ClangTidyAnalysisHandler используется для представления команды — абстрактного действия, которое будет вызываться через меню запуска анализа. В файле plugin.xml в точке расширения org.eclipse.ui.commands была добавлена соответствующая команда. В единственном методе execute происходит получение selection пользователя, создание списка resources и попытка найти в selection подходящие ресурсы. Реализация позволяет выбрать несколько проектов для анализа. Если resources не пуст, создается и запускается ClangTidyAnalysisTool.

Как говорилось ранее, ClangTidyAnalysisTool реализует интерфейс IAnalysisTool. В файле plugin.xml в точке расширения su.softcom.cldt.core.analyzisTool был добавлен соответствующий analyzisTool. Использование интерфейса необходимо для того, чтобы в уже имеющийся до этой работы класс AnalysisProvider в поле Set<IAnalysisTool> availableTools попало реализованное средство анализа, что позволяет кнопке запуска анализа clang-tidy появиться в UI. В методе execute реализовано выполнение подготовки к началу

операции анализа и его запуск, но при большом количестве включенных проверок clang-tidy стало ясно, что анализ блокирует основной поток программы. Для решения задачи асинхронного запуска анализа ClangTidyAnalysisTool стал наследоваться от класса Job из одноимённого фреймворка. Фреймворк Job в Eclipse позволяет разработчикам создавать и управлять асинхронными задачами, не блокируя основной поток пользовательского интерфейса. В методах класса произошли необходимые изменения. По итогам запуск операции перестал блокировать основной поток и у класса появилась функциональность, позволяющая работать с операцией анализа, как с асинхронной задачей. В методе initProjectSourceFiles происходит рекурсивный поиск всех исходных файлов проекта или директории. Для подготовки аргументов, предназначенных объекту класса ClangTidyOperation, написаны методы getClangTidyParameters и findConsole. Полученные с помощью них и другие необходимые аргументы передаются в конструктор ClangTidyOperation.

Интерфейс для класса ClangTidyOperation был взят также из фреймворка Job. Реализованный классом ClangTidyOperation интерфейс ICoreRunnable предназначен для создания задач, которые могут выполняться асинхронно, обрабатывать прогресс выполнения и возможные ошибки. Внутри класса был написан метод compileCommandsExists, выполняющий проверку существования базы данных команд компиляции. Если её не существует в данном проекте, запускается сборка конфигурации проекта. Метод configure формирует список строк launchArguments параметров запуска clang-tidy. Метод run запускает процесс анализа с использованием launchArguments. Результат анализа сразу выводится в консоль (Рис. 3) с помощью метода writeToConsole.

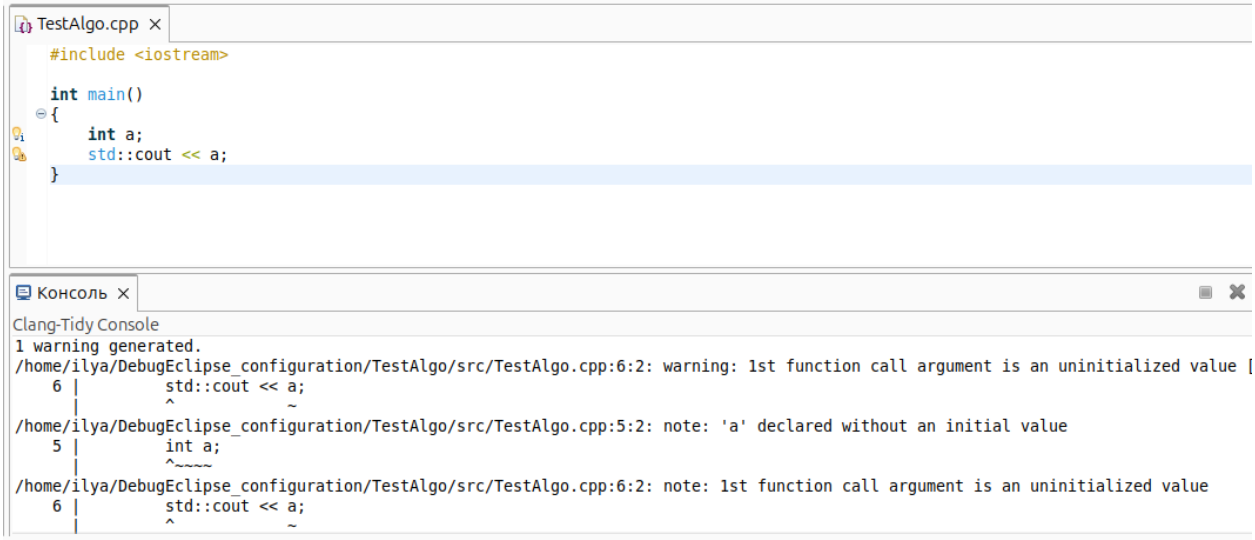

Рис. 3: Пример вывода clang-tidy в консоль.

3.4. Обработка вывода clang-tidy

Для обработки вывода был реализован класс ClangTidyOutputParser. Метод parse получает строку вывода clang-tidy и проверяет её на соответствие регулярному выражению. Регулярное выражение имеет несколько групп захвата, оно считывает путь к файлу, номер строки и номер символа в строке, уровень важности, текст сообщения и раздел предупреждения. В случае совпадения строка разделяется на эти группы, из которых далее используются: путь к файлу, номер строки, степень критичности, сообщение. По этим данным генерируется маркер ошибки, предупреждения или замечания на нужных строках в файлах исходного кода (Рис. 4).

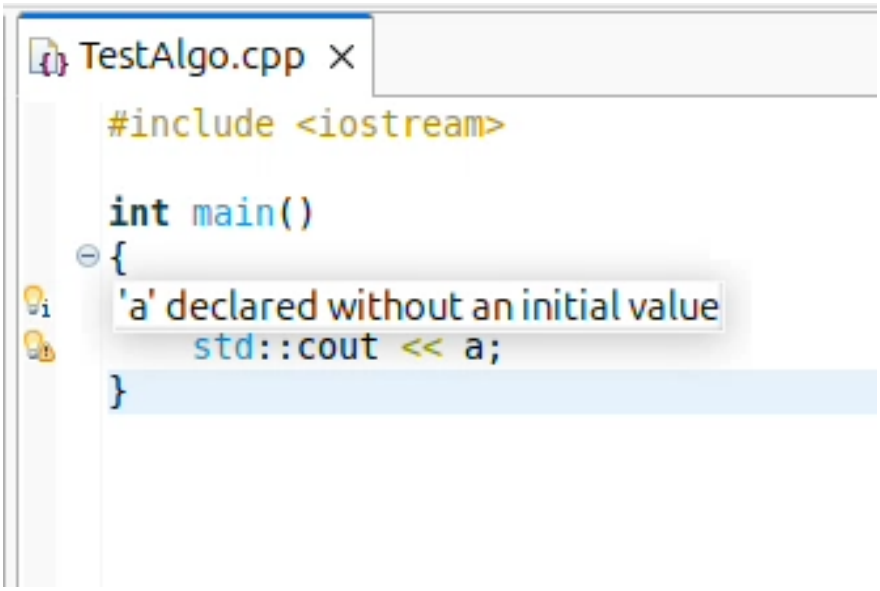

Рис. 4: Пример маркера, сгенерированного с помощью clang-tidy.

4. Апробация

Апробация была проведена с консультантом и работником компании. Целью являлась оценка качества кода, проверка работоспособности плагина и соответствия реализованной функциональности ожиданиям. Были созданы 3 запроса на слияние, отдельно для страницы настроек, реализации запуска и парсинга вывода. В ходе ревью была проведена работа по исправлению и доработке плагина. Результаты работы частично добавлены в главную ветку проекта.

Заключение

В ходе выполнения данной работы были достигнуты все поставленные цели.

- Спроектирована архитектура плагина для интеграции clang-tidy.
- Реализовано получение и хранение параметров запуска clang-tidy через пользовательский интерфейс.
- Реализован запуск clang-tidy.
- Реализована обработка вывода clang-tidy.

Код работы является закрытым.

Список литературы

[1] Clang static analyzer официальный сайт. — URL: https://clang-analyzer.llvm.org/.

[2] Clang-tidy документация. — URL: https://clang.llvm.org/extra/clang-tidy/.

[3] CodeChecker Eclipse plugin. — URL: https://github.com/Ericsson/CodeCheckerEclipsePlugin.

[4] Eclipse CDT официальный сайт. — URL: https://projects.eclipse.org/projects/tools.cd.

[5] Eric Clayberg Dan Rubel. Eclipse Plug-ins, 3-е издание. — Addison-Wesley, 2008.

[6] SWT Eclipse официальный сайт. — URL: https://www.eclipse.org/swt/.

[7] Официальный сайт Eclipse IDE. — URL: https://eclipseide.org/.

[8] Официальный сайт Eclipse PDE. — URL: https://eclipse.dev/pde/.

[9] Репозиторий JFace Eclipse. — URL: https://github.com/eclipse-platform/eclipse.platform.ui/blob/master/docs/JFace.md.

[10] Спецификация JAR файла. — URL: https://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html.