

# ВОКРУГ РБПО ЗА 25 ВЕБИНАРОВ

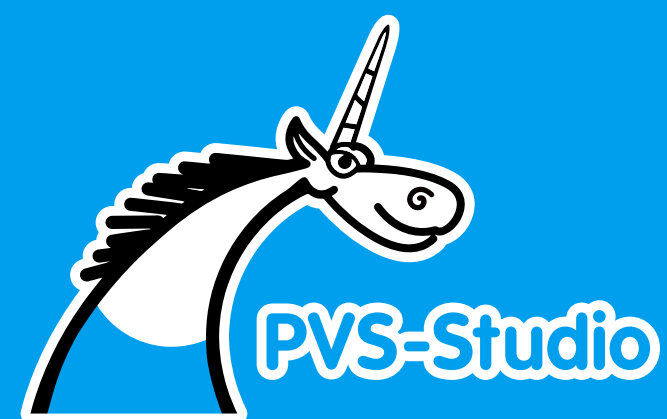
ГОСТ Р 56939-2024

## Вебинар 8. Формирование и поддержание в актуальном состоянии правил кодирования





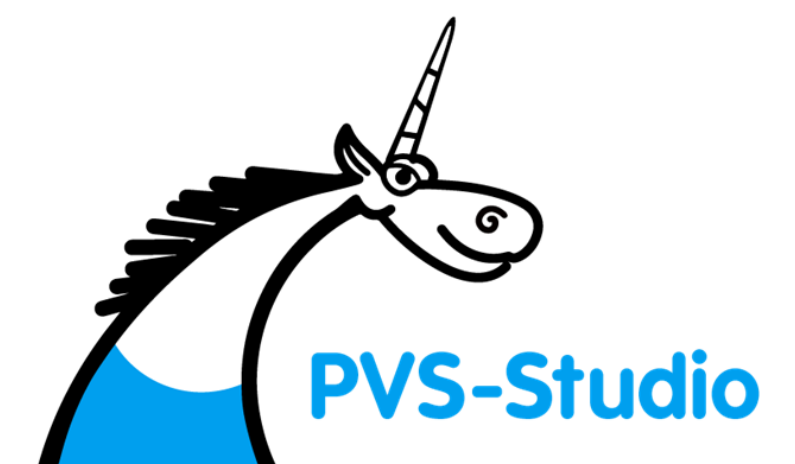
# Докладчики и гость вебинара



# Андрей Карпов

Директор по развитию бизнеса (CBDO)

- Один из основателей проекта PVS-Studio  
<https://pvs-studio.ru/>
- 18 лет в сфере качества и анализа кода
- Хабр: [@Andrey2008](#)
- Email: [karpov \(@\) viva64.com](mailto:karpov (@) viva64.com)



# Виталий Пиков

Эксперт в области ИТ, ИБ, преподаватель

- Стаж преподавательской работы более 10 лет
- Заслуженный доцент Российского нового университета, преподаватель высшей школы
- Microsoft Certifications Earned: MCT, MCPS, MCSA, MCTS
- Автор более 30 научных публикаций



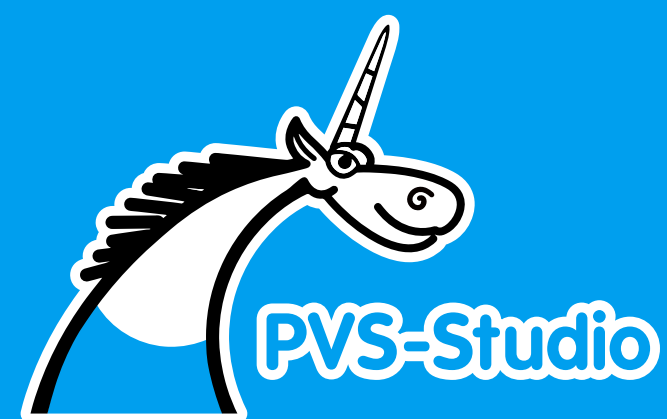
# Анна Мелехова

Senior security architect в Kaspersky OS

- 20+ лет в ИТ
- Занималась системным программированием (облака, виртуализация, операционные системы) и архитектурой
- Сейчас senior security architect в [Kaspersky OS](#)
- К.т.н., преподаёт в Университете Иннополис
- Email: [@kaspersky.com](mailto:anna.melekhova)



# О цикле вебинаров и других мероприятиях

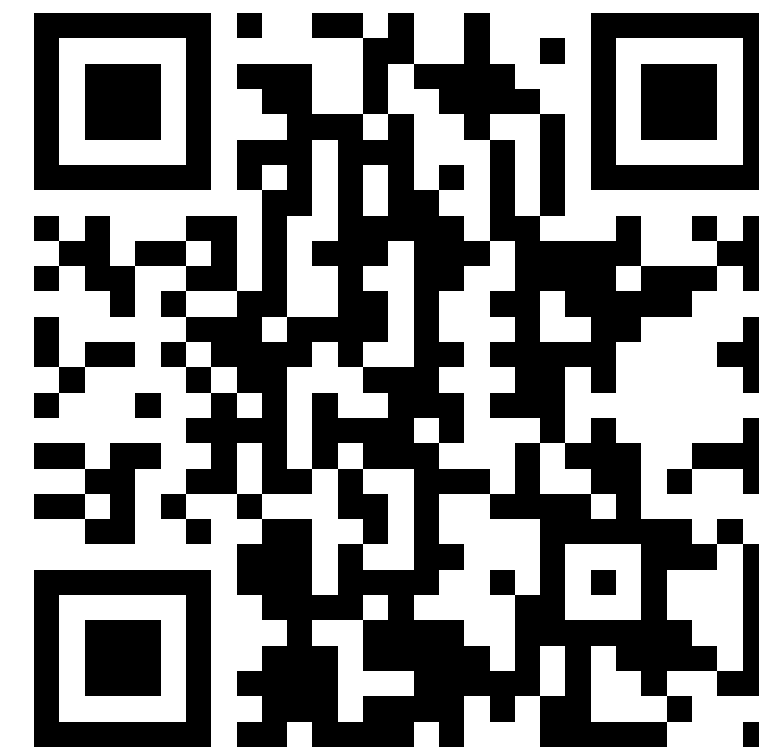




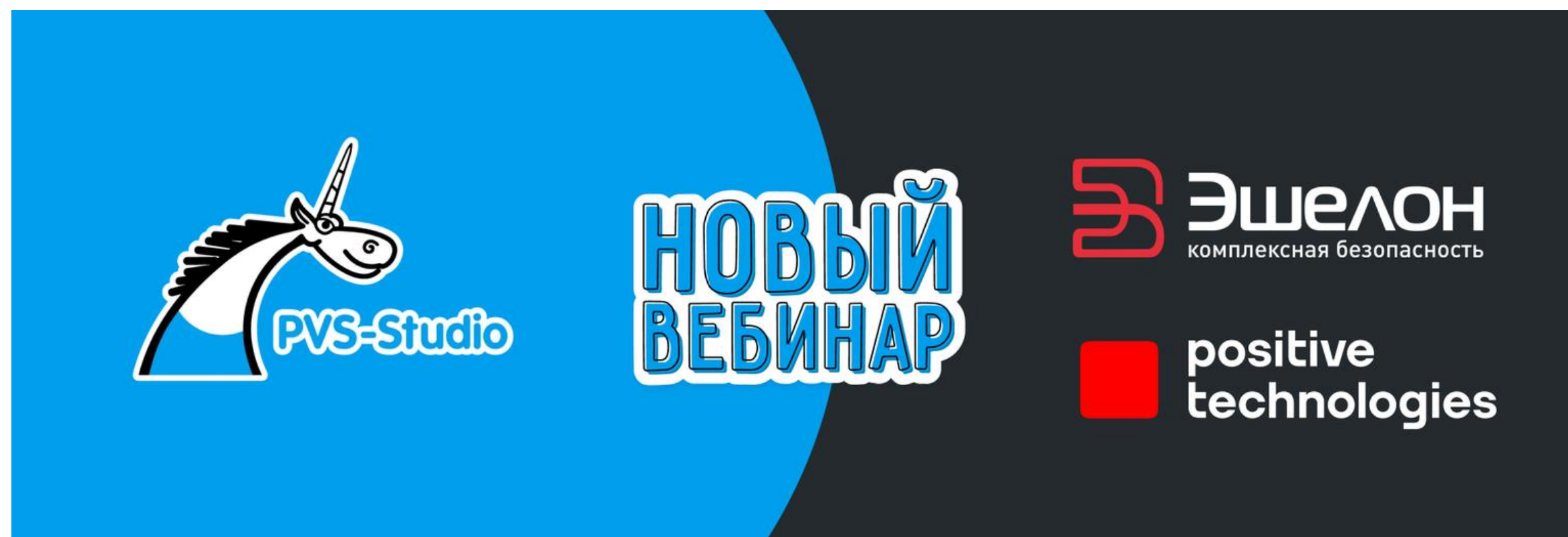
# Вокруг РБПО за 25 вебинаров

7

- Организует УЦ МАСКОМ и ООО «ПВС» (PVS-Studio)
- 25 вебинаров, т.к. ГОСТ Р 56939-2024 описывает 25 процессов для реализации разработки безопасного ПО
- На самом деле, больше за счёт бонусных вебинаров 😊
- Мы открыты к сотрудничеству по разбору тем, пишите нам
- Записи вебинаров:  
[pvs-studio.ru/ru/webinar/rbpo/](https://pvs-studio.ru/ru/webinar/rbpo/)



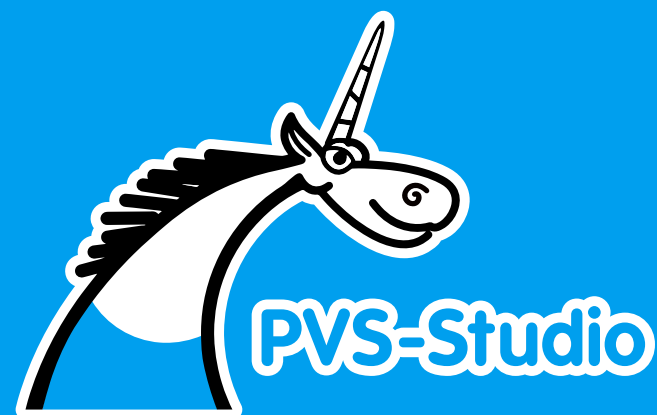
- Вебинар: Техническая сторона первого этапа испытаний статических анализаторов кода под эгидой ФСТЭК  
1 сентября 15:00  
Регистрация: [pvs-studio.ru/ru/webinar/](https://pvs-studio.ru/ru/webinar/)



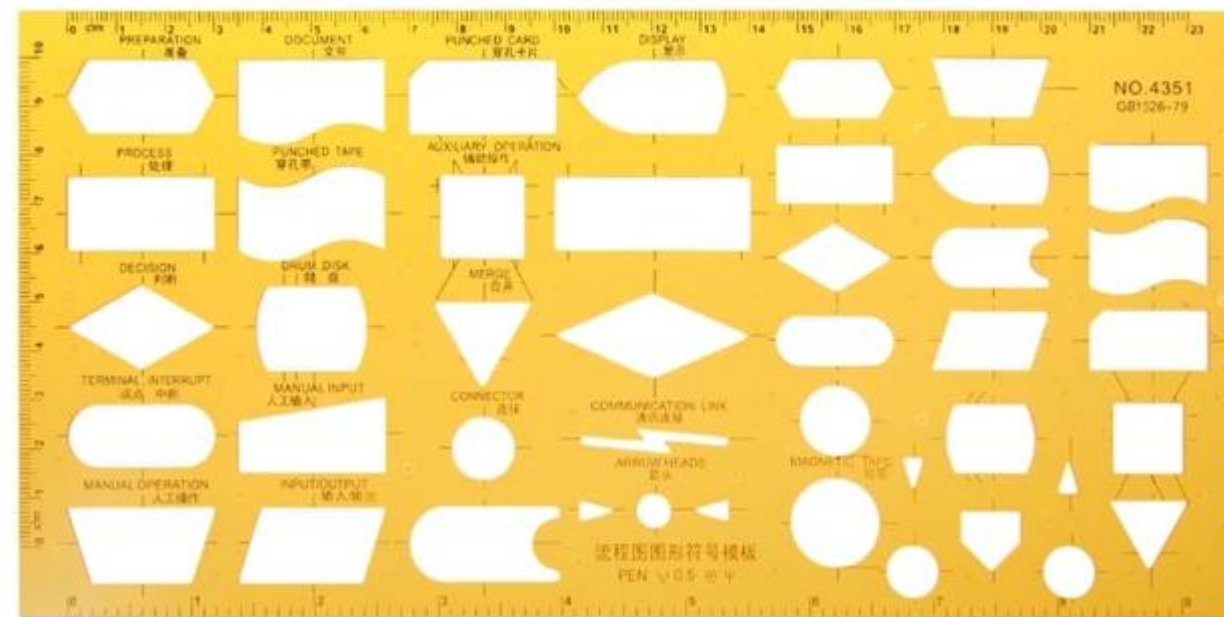


# Процесс 8

Формирование и поддержание в  
актуальном состоянии правил  
кодирования



- 5.8.1.1 Обеспечение эффективной и единообразной организации оформления и использования исходного кода в соответствии с предъявляемыми к ПО требованиями



# Для чего нужно единообразие?

11

- Основное время при работе с кодом программисты тратят не на его написание, а на чтение!
- Единообразие позволяет быстрее понимать код коллег (и собственный старый :)





# Причины, почему с ростом проекта всё больше требуется времени на чтение кода

12

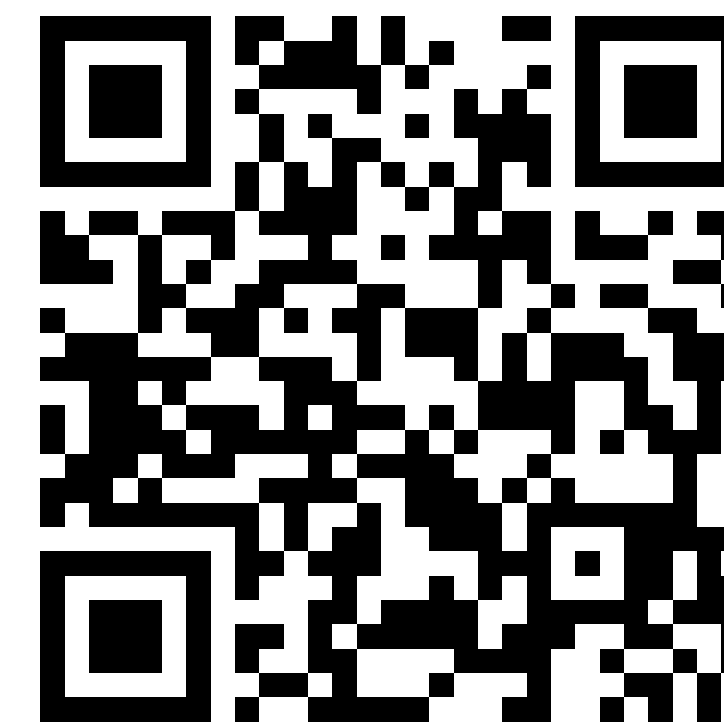
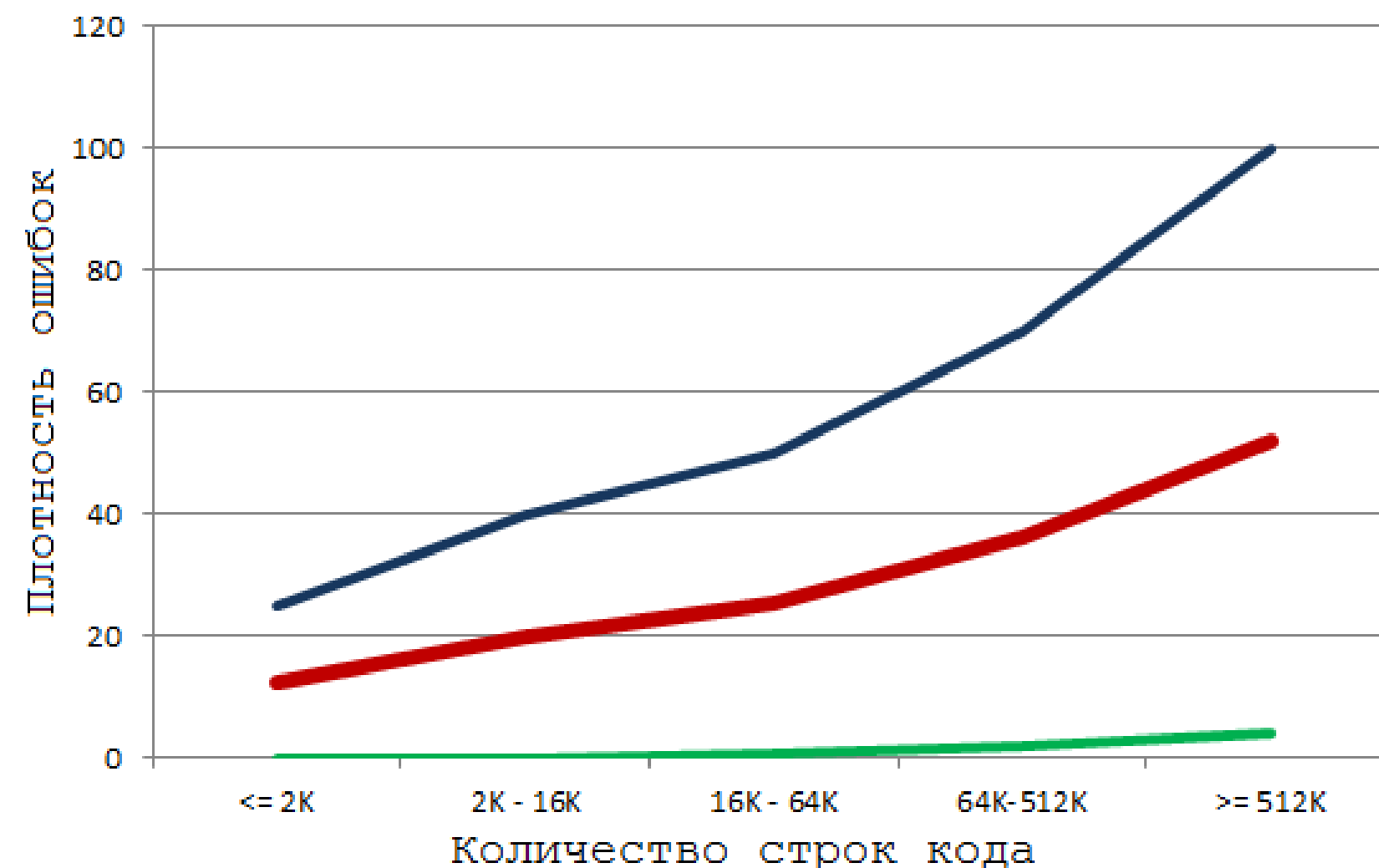
- В большом (legasy) проекте чаще нужно изменить старое поведение, а не создать что-то принципиально новое
- В большом проекте почти весь код «чужой», и бывает трудно найти нужное место и понять, как всё устроено
- В большом проекте, если вы долго с ним работаете, даже ваш собственный код со временем становится «чужим»



# Продолжаем перечислять причины

13

- В legacy-проектах можно наблюдать «годовые кольца». Разнородность добавляет сложность к пониманию
- С ростом проекта плотность ошибок растёт нелинейно. Всё аккуратнее следует писать и править код



- Как вносить правки в код коллег, оформленный в их собственном стиле?
- Хорошего варианта нет:
  - Написать нужный код в своём стиле. Результат – «каша» стилей
  - Постараться подстроиться под код коллег и написать в их стиле. Нереально
  - Отрефакторить код коллег под свой стиль. Большая трата времени и большие diff'ы, в которых невозможно понять, где рефакторинг, а где полезные изменения







- Молодые специалисты могут возражать против принятых правил, обосновывая это тем, что они ограничивают их творчество и скорость написание кода
- Или тем, что правила устарели и не учитывают какие-то моменты

- Даже не очень хороший стандарт лучше, чем его отсутствие
- Однако это не значит, что надо преодолевать неудобства!
- Можно и нужно совершенствовать регламенты
- Это даже отражено в названии восьмого процесса РБПО –  
формирование и **поддержание в актуальном  
состоянии** правил кодирования





- 5.8.2.1. **Принять и использовать** в процессе разработки кода ПО **регламент** оформления исходного кода и безопасного кодирования для используемых разработчиком языков программирования.
- Примечание — Под безопасным кодированием здесь и далее понимаются практики разработки ПО в соответствии с предъявляемыми в указанном выше регламенте требованиями по безопасности.

- 5.8.2.2 Учитывать при разработке регламента оформления исходного кода и безопасного кодирования примеры опасных и безопасных конструкций для используемых в ПО языков программирования.
- P.S. Это может проверяться в рамках обзоров кода, про которые мы поговорим в рамках 9 процесса «Экспертиза исходного кода»

- 5.8.2.3 Учитывать при разработке регламента оформления исходного кода и безопасного кодирования общепринятые стандарты и рекомендации разработчиков (экспертов, специалистов) для соответствующих языков программирования.



- 5.8.2.4 При разработке ПО рекомендуется использовать программные средства автоматической проверки правил кодирования.
- Примечание – Допускается реализовывать проверку правил кодирования средствами компиляции или статического анализа.

# Что взять за основу регламента?

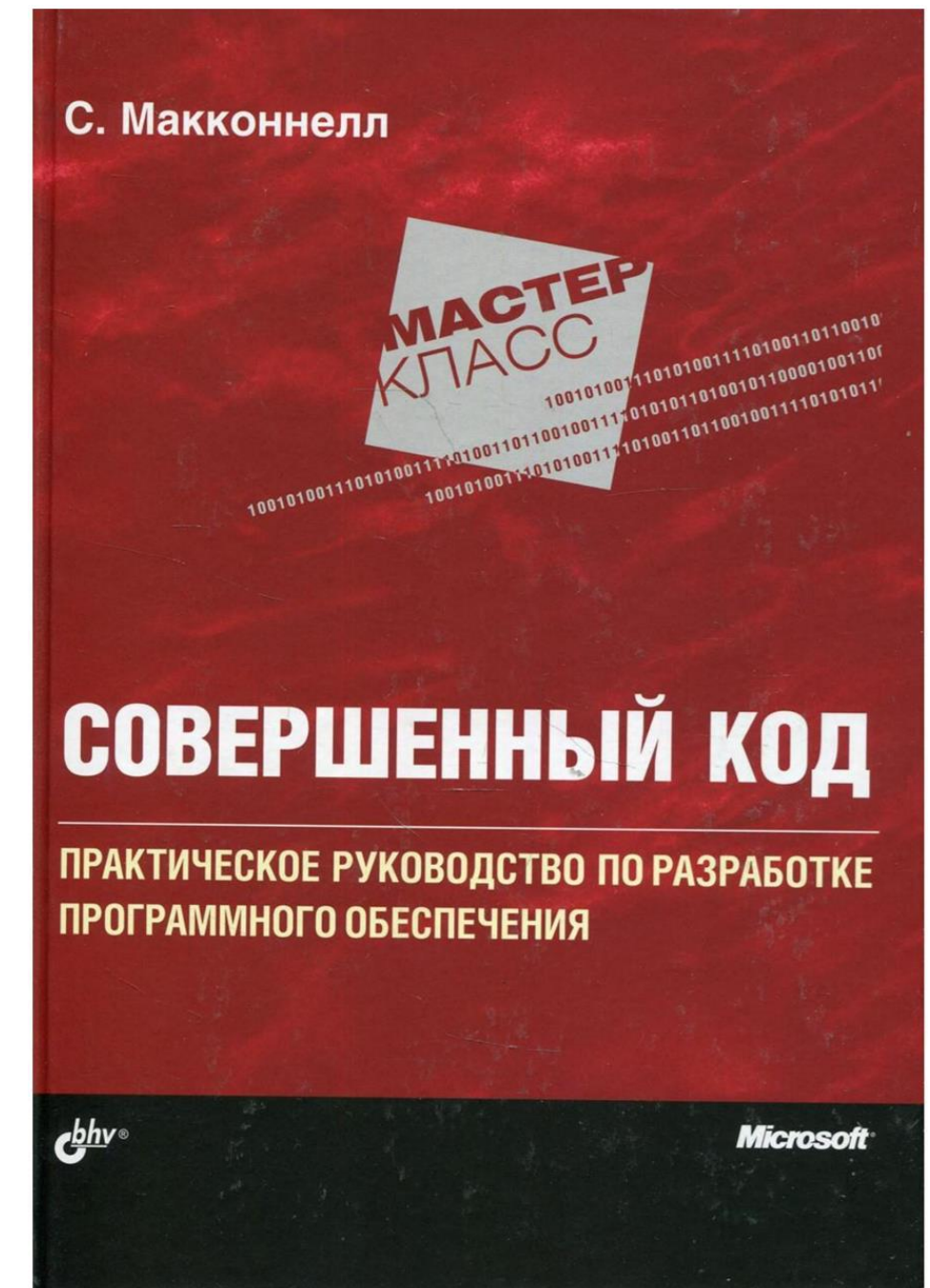
22

- Нет простого ответа (образца), да и нет смысла копировать чужое
- Воспользуйтесь своим опытом, привлечите своих экспертов, учтите особенности своего проекта
- О вреде необдуманного заимствования на примере стандартов [MISRA C и MISRA C++](#)

# Что взять за основу регламента?

23

- Книга "Совершенный код" Стива Макконнелла ("Code Complete", Steve McConnell)
- С++ программисты могут почерпнуть идеи из этого документа с хорошими мыслями и отсылками к другим стандартам:  
[Coding Standards](#)



# Не рекомендую погружаться в такие стандарты, как :

24

- MISRA C/C++
- CWE (Common Weakness Enumeration)
- OWASP Application Security Verification Standard (ASVS)
- SEI CERT Coding Standard
- И так далее



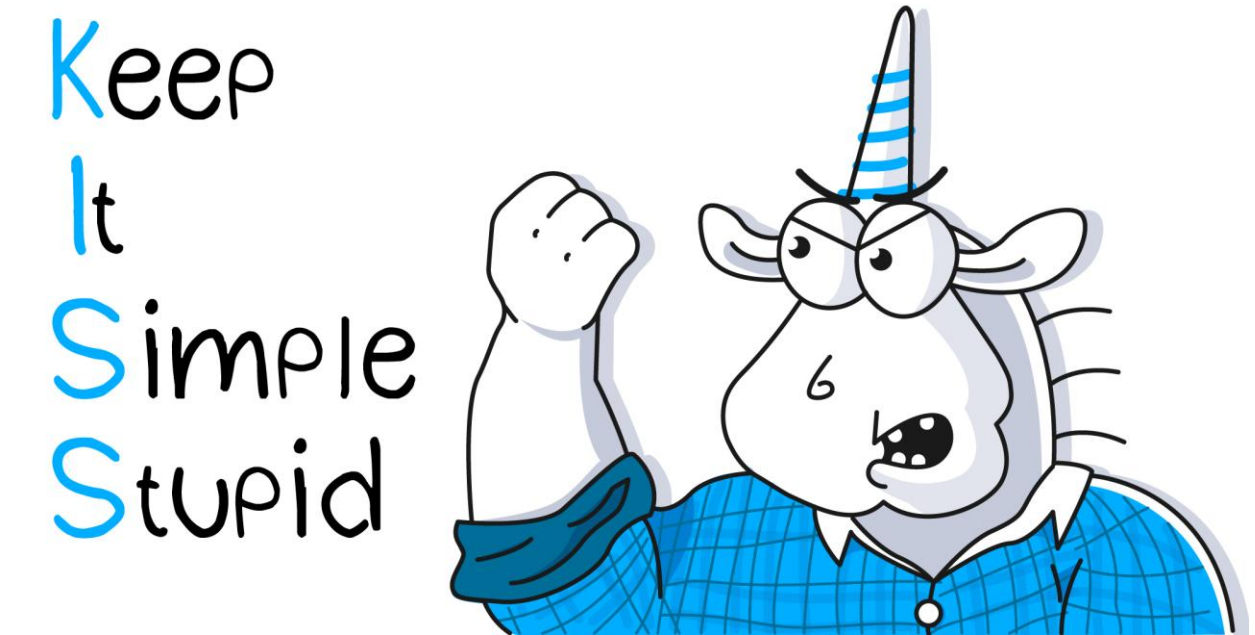


- Очень долго и сложно всё это читать, перерабатывать
  - Выпишите 100500 и очевидных правил «не делите на ноль», «не логируйте пароли» и т.д.
  - По отдельности вроде всё верно и полезно. Но чек-лист из 1000 пунктов на практике бесполезен!
  - Для этого есть статические анализаторы кода!
- Примечание стандарта:
- Допускается реализовывать проверку правил кодирования средствами компиляции или статического анализа*

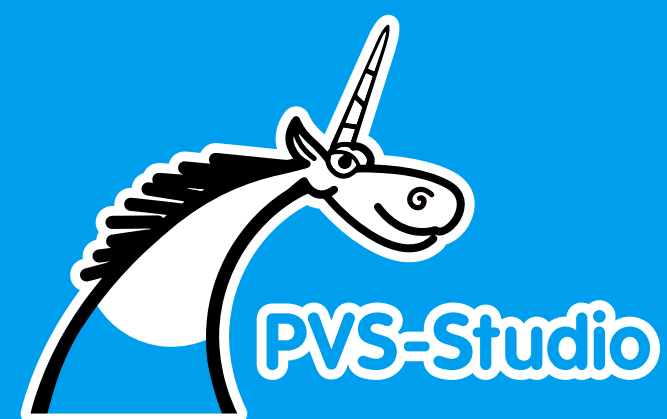
# Если ничего нет, начните с простого

26

- Соберите экспертную группу
- Сначала запишите то, как у вас принято писать сейчас
- Внесите полезные улучшения/рекомендации из книг и т.п.
- Утвердите с коллегами и начните неуклонно использовать
- Постепенно усовершенствуйте, делайте вариации для новых языков и т.д.



# Артефакты реализации требований



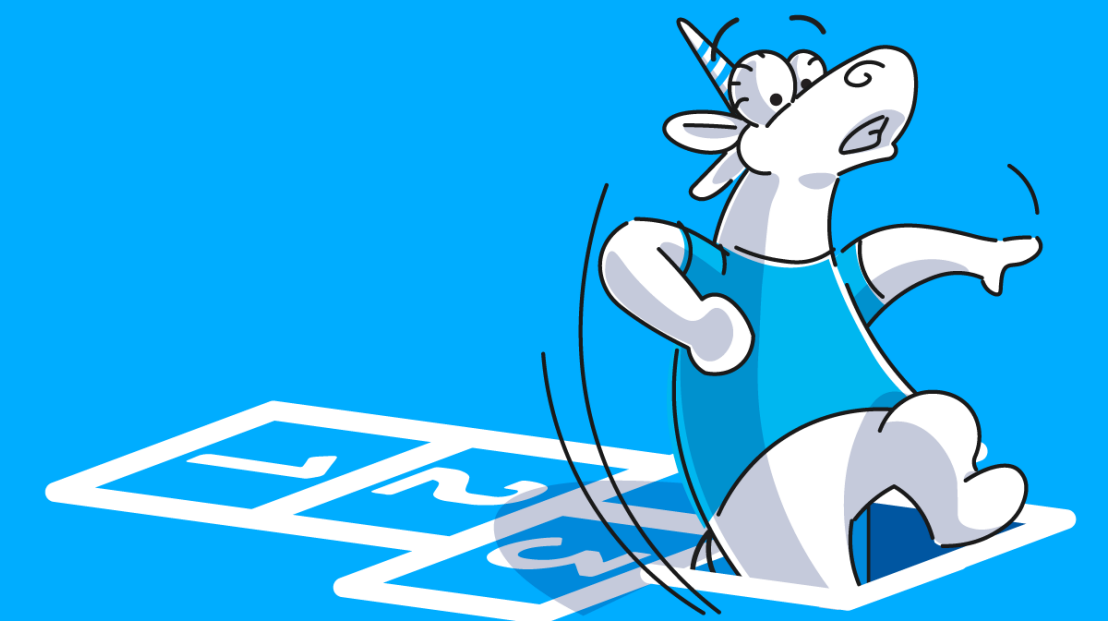
## 5.8.3.1 Регламент оформления исходного кода и безопасного кодирования должен содержать:

28

- информацию о способах оформления исходного кода (например,
  - способы выбора наименований переменных, функций, классов и т. п.;
  - стиль отступов при оформлении логических блоков;
  - способы ограничения логических блоков;
  - правила использования пробелов при оформлении логических и арифметических выражений;
  - стиль комментариев и правила документирования кода;
  - ограничения [например, размер строк кода по горизонтали, строк в модуле и т. п. ]);



- перечень запрещённых способов кодирования, конструкций и т. п. (например,
  - указание паролей в исходном коде ПО в явном виде,
  - использование «магических чисел» и т. п.);



- примеры опасных и безопасных конструкций для используемых языков программирования;
- область применения правил кодирования;
- порядок проверки выполнения правил кодирования для вносимых изменений в исходный код ПО.

# Артефакты. Регламент должен содержать

31

- рекомендации разработчиков языков программирования по использованию стандартов кодирования (языков программирования, в том числе собственной разработки), принятые разработчиком ПО.



# Не хочу делать доклад слишком длинным, поэтому ссылки для ознакомления

32

- Типовые паттерны опечаток в коде и как с ними бороться  
[pvs-studio.ru/ru/blog/video/11394/](https://pvs-studio.ru/ru/blog/video/11394/)

Типовые паттерны опечаток в коде и как с ними бороться

## Ошибка заметнее, но много пробелов

36

```
static bool rpcNoDelayMsg(tmsg_t msgType) {  
    if (msgType == TDMT_VND_FETCH_TTL_EXPIRED_TBS ||  
        msgType == TDMT_VND_S3MIGRATE ||  
        msgType == TDMT_VND_S3MIGRATE ||  
        msgType == TDMT_VND_QUERY_COMPACT_PROGRESS ||  
        msgType == TDMT_VND_DROP_TTL_TABLE) {  
        return true;  
    }  
    return false;  
}
```

13:17

VK Видео







# Стандарт кодирования PVS-Studio и приёмы при разработке эффективных C++ диагностик

33

- [pvs-studio.ru/ru/blog/video/10005/](https://pvs-studio.ru/ru/blog/video/10005/)


**Андрей Карпов**  
Лекция 8. Стандарт кодирования PVS-Studio










**Стандарт: именование типов**

- Классы именуются с большой буквы. Слова отделяются в названии также большой буквой. Примеры: `Ptree`, `NonLeaf`, `PtreeArray`
- Если нужно создать синоним простого типа, то правила формирования имён такие же, как и для классов, например: `using LineNumber = size_t;`
- Имена перечислений начинаются с буквы `E`. В остальном правила именования совпадают с правилами именования классов. Пример: `EGetArrayName`

 viva64.com

 support@viva64.com

    @pvsstudio\_rus



- P.S. Видео 2019 года, поэтому кое-что у нас уже поменялось. Сейчас я бы рассказал немного по-другому и привёл бы иные примеры. Тем более учитывая, что за это время мы переписали ядро C++ анализатора. Но в целом ок, полезное. Предлагаю к просмотру.

- PDF: [t.me/programming\\_tales/387](https://t.me/programming_tales/387)
- Когда будете смотреть, обратите внимание, как много частностей и специфики!
- Вновь призываю не копировать, а сделать свой, удобный и полезный для себя
- Наш C# стандарт похож на C++
- Java-команда придерживается [Google Java Style Guide](#) с небольшими модификациями

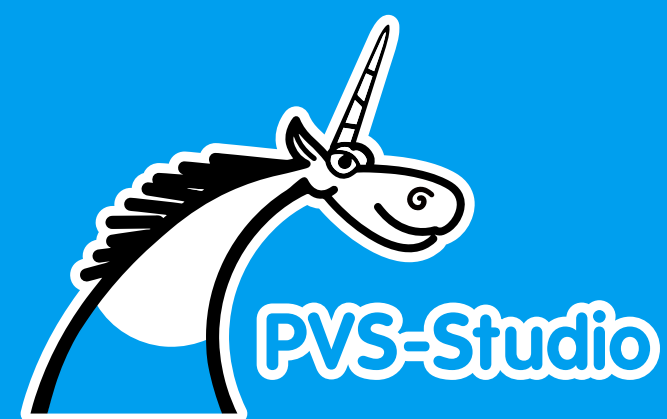


# Инструменты автоматизированного форматирования кода

35

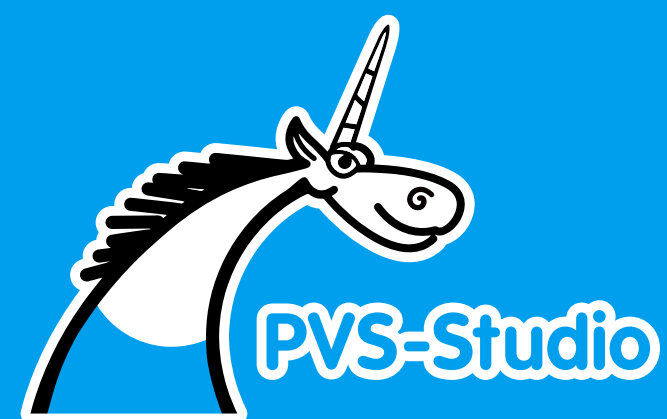
- ClangFormat - [clang.llvm.org/docs/ClangFormat.html](http://clang.llvm.org/docs/ClangFormat.html)
- Uncrustify - [github.com/uncrustify/uncrustify](https://github.com/uncrustify/uncrustify)

Пишите простой и красивый код:  
его будет легче понять, а  
ошибки в нём будут заметнее

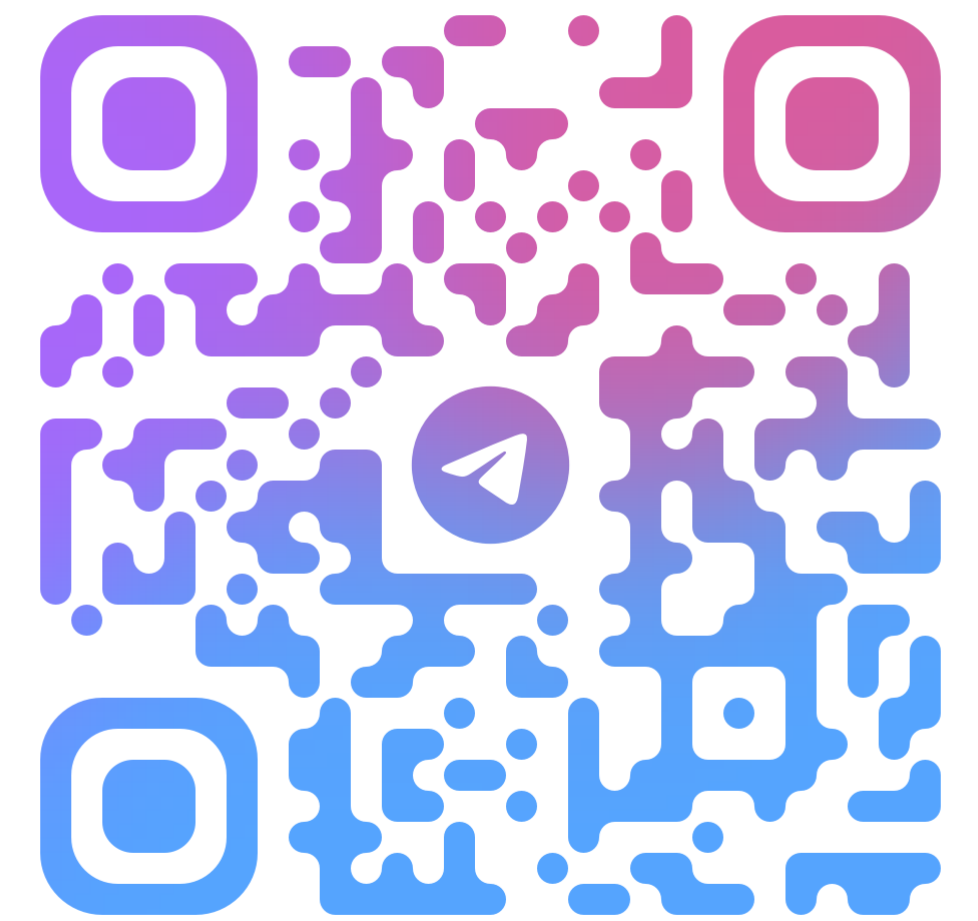




# Дополнительные материалы



- Приглашаю подписаться на мой TG-канал «Бестиарий программирования», где я публикую цикл постов, посвящённых РБПО [@programming\\_tales](https://www.instagram.com/programming_tales)



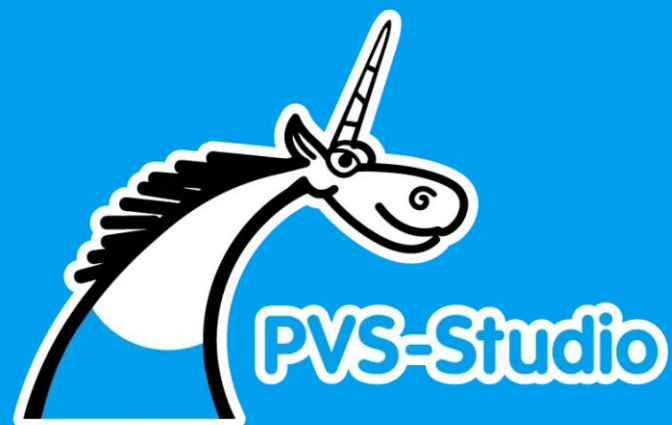
@PROGRAMMING\_TALES

- Главный вопрос программирования, рефакторинга и всего такого [pvs-studio.ru/ru/blog/posts/cpp/0391/](https://pvs-studio.ru/ru/blog/posts/cpp/0391/)
- Учимся рефакторить код на примере багов в TDengine:
  - часть 1: про колбасу – [pvs-studio.ru/ru/blog/posts/cpp/1230/](https://pvs-studio.ru/ru/blog/posts/cpp/1230/)
  - часть 2: макрос, пожирающий стек – [pvs-studio.ru/ru/blog/posts/cpp/1238/](https://pvs-studio.ru/ru/blog/posts/cpp/1238/)
  - часть 3: плата за лень – [pvs-studio.ru/ru/blog/posts/cpp/1242/](https://pvs-studio.ru/ru/blog/posts/cpp/1242/)
- Рефакторим легаси при помощи ООП [pvs-studio.ru/ru/blog/posts/csharp/1159/](https://pvs-studio.ru/ru/blog/posts/csharp/1159/)
- Форматирование кода таблицей [pvs-studio.ru/ru/blog/terms/7003/](https://pvs-studio.ru/ru/blog/terms/7003/)

Сделай свой проект  
чистым и безопасным  
вместе с PVS-Studio



**VOKRUG\_RBPO25**



Получи 10% скидку  
на курсы «М БРПО»  
в Учебном Центре «МАСКОМ»



**VOKRUG\_RBPO25**





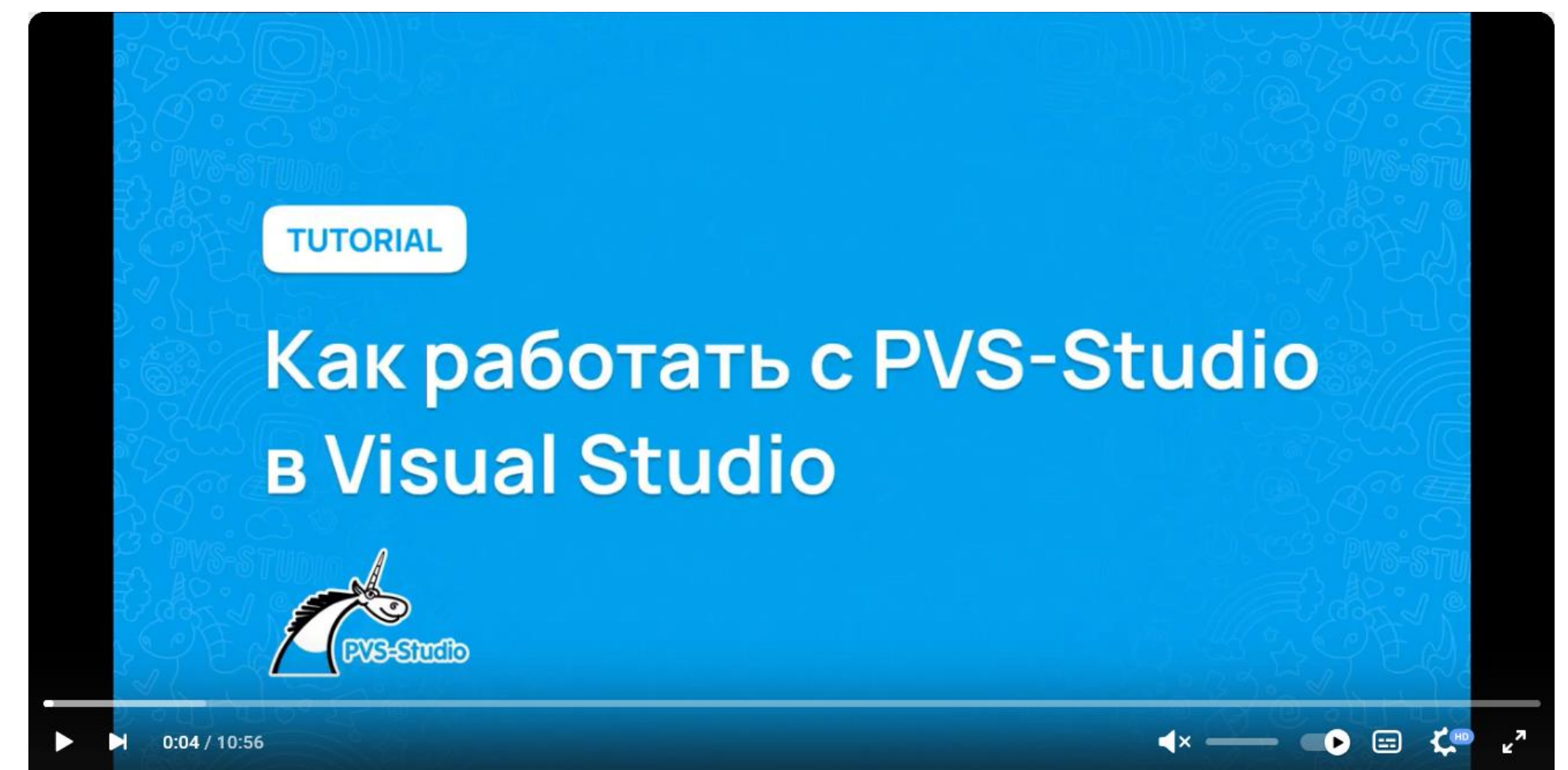
# Карпов Андрей Николаевич

41

- Карпов Андрей Николаевич, 1981
- ООО «ПВС», директор по развитию бизнеса
- Более 18 лет занимается темой статического анализа кода и качества программного обеспечения. Автор большого количества статей, посвящённых написанию качественного кода на языке C++. Один из основателей проекта PVS-Studio. Долгое время являлся СТО компании и занимался разработкой C++ ядра анализатора. Основная деятельность на данный момент — развитие компании, обучение сотрудников и DevRel деятельность
- [Другая информация и контакты](#)



- Статический анализатор кода для поиска ошибок и потенциальных уязвимостей
- Поддерживает: **C, C++, C#, Java**
- Краткое знакомство:



[vkvideo.ru/video-11805870\\_456239344](https://vkvideo.ru/video-11805870_456239344)

- Включён в Реестр российского ПО: запись № 9837
- Совместим с **ГОСТ Р 71207-2024** (Статический анализ кода)
- Соответствие требованиям «Методики выявления уязвимостей и недекларированных возможностей в программном обеспечении» от 25 декабря 2020 г.
- Может применяться для РБПО согласно **ГОСТ Р 56939-2024**
- Бесплатные лицензии для студентов и преподавателей





**ПИКОВ**  
**Виталий**  
**Александрович**

**Общий стаж работы:** более 26 лет.

**Стаж преподавательской работы:** более 10 лет.

**Образование:** высшее, Тамбовский военный авиационный инженерный институт по специальности «Автоматизированные системы обработки информации и управления».

Заслуженный доцент Российского нового университета, преподаватель высшей школы.

В 2017 году прошёл профессиональную переподготовку в МГТУ им. Н. Э. Баумана по направлению подготовки «Информационная безопасность».

В 2019 году прошёл профессиональную переподготовку по программе «Противодействие иностранным техническим разведкам».

В 2020 году прошёл профессиональную переподготовку по программе «Педагогика профессионального обучения, профессионального образования и дополнительного профессионального образования».

В 2021 году прошёл профессиональную переподготовку по дополнительной профессиональной программе «ТЗИ».

В 2022 году прошёл профессиональную переподготовку по программе «Практическая психология».

**Microsoft Certifications Earned: MCT, MCPS, MCSA, MCTS.**

**Автор более 30 научных публикаций.**

Постоянный участник, спикер, эксперт на мероприятиях по информационной безопасности: Positive Hack Days Fest 2, Национальный форум информационной безопасности «Инфофорум», Международный военно-технический форум «АРМИЯ», Международная выставка InfoSecurity Russia, Международная научная конференция «Цивилизация знаний: российские реалии» (РосНОУ) и некоторых других.

Имею награды и звания Минобороны России.

**Авторизованный преподаватель по продуктам «Группы Астра» с правом проведения курсов по ОС Astra Linux Special Edition 1.8**

Читаю курсы, провожу занятия в области информационной безопасности, защиты информации и информационных технологий.

