

# Особенности разработки встроенного ПО по требованиям ФБ



**Андрей Карпов**  
PVS-Studio



**Андрей Рогов**  
ООО «ФанкСэйфети»



**Елена Рогова**  
ООО «ФанкСэйфети»





# 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](https://habr.com/ua/u/Andrey2008/)
- Telegram канал  
«Бестиарий программирования»  
[t.me/programming\\_tales](https://t.me/programming_tales)



**PVS-Studio**

# Рогова Елена Сергеевна

Ведущий специалист по функциональной безопасности, к.т.н., ООО "ФанкСэйфети"

# Рогов Андрей Сергеевич

Генеральный директор, к.т.н.,  
ООО "ФанкСэйфети"

Консалтинг. Фокус: предоставление услуг в области функциональной безопасности (ФБ) и надёжности — [func-safety.ru](https://func-safety.ru)

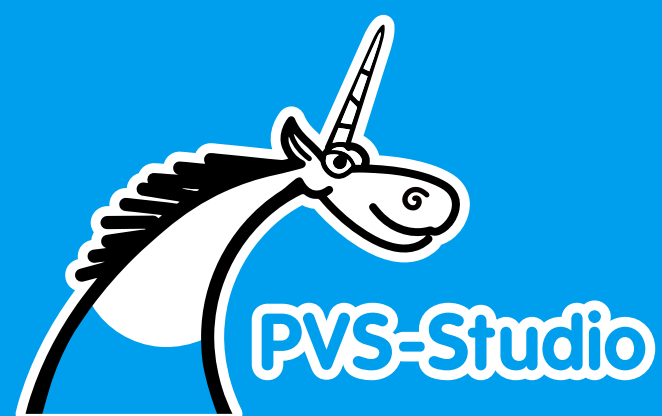


# Особенности разработки встроенного ПО по требованиям ФБ

Елена Рогова

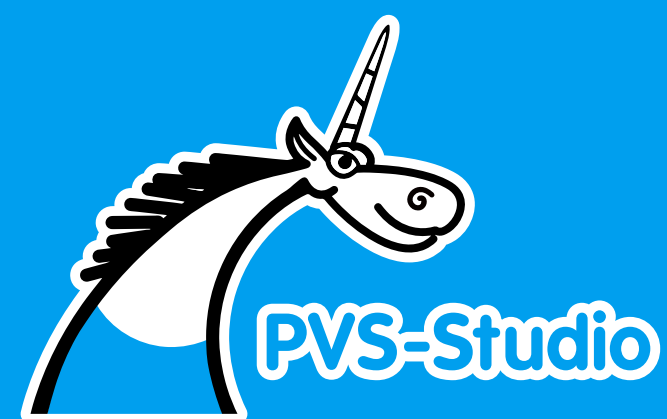


Андрей Рогов





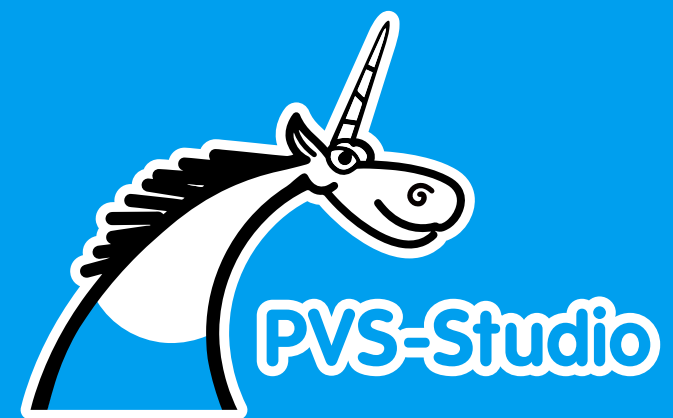
# MISRA и PVS-Studio



Андрей Карпов



# PVS-Studio



- Статический анализатор кода для поиска ошибок и потенциальных уязвимостей
- Поддерживает: **C, C++, C#, Java**
- Запускается на большом количестве ОС, в том числе отечественных:
  - Windows
  - macOS
  - Arch Linux, Astra Linux, CentOS, Debian GNU/Linux, Fedora, Linux Mint, openSUSE, Ubuntu, РЕД ОС.

- Включён в Реестр российского ПО: запись № 9837
- Совместим с **ГОСТ Р 71207-2024** (Статический анализ кода)
- Соответствует требованиям «Методики выявления уязвимостей и недекларированных возможностей в программном обеспечении» от 25 декабря 2020 г.
- Может применяться для РБПО согласно **ГОСТ Р 56939-2024**
- Участвует в инициативе ФСТЭК по испытанию статических анализаторов



# Мы сейчас занимаемся наполнением PVS-Studio детекторами MISRA C

9

- Из пресс-релиза PVS-Studio 7.39:
  - V2652. MISRA. Argument of an integer constant macro should have an appropriate form.
  - V2653. MISRA. The small integer variants of the minimum-width integer constant macros should not be used.
  - V2654. MISRA. Initializer list should not contain persistent side effects.
  - V2655. MISRA. The right operand of a logical '&&' or '||' operator should not contain persistent side effects.
  - V2656. MISRA. The Standard Library function memcmp should not be used to compare null terminated strings.
  - V2657. MISRA. Obsolescent language features should not be used.
  - V2658. MISRA. Dead code should not be used in a project.
  - V2659. MISRA. Switch statements should be well-formed.
  - V2660. MISRA. A function declared with a \_Noreturn specifier should not return to its caller.
  - V2661. MISRA. MISRA. A 'for' loop should be well-formed.
  - V2662. MISRA. Any value passed to a function from <ctype.h> should be representable as an unsigned character or be the value EOF.

# MISRA



# Стандарты/регламенты написания кода

11

- Набор правил и соглашений, используемых при написании исходного кода
- Наличие общего стиля облегчает понимание и поддержание исходного кода, написанного несколькими программистами
- Формализуют анти-паттерны (типовые баги)
- Запрет использования некорректных и опасных конструкций
- Примеры: [isocpp.org/wiki/faq/coding-standards](http://isocpp.org/wiki/faq/coding-standards)





- **Выполнять контроль можно вручную, но это трудоёмко и нерационально**
- Проверку многих аспектов стандартов кодирования можно автоматизировать
- Это делают статические анализаторы кода

# С точки зрения РБПО приоритетнее искать критические ошибки

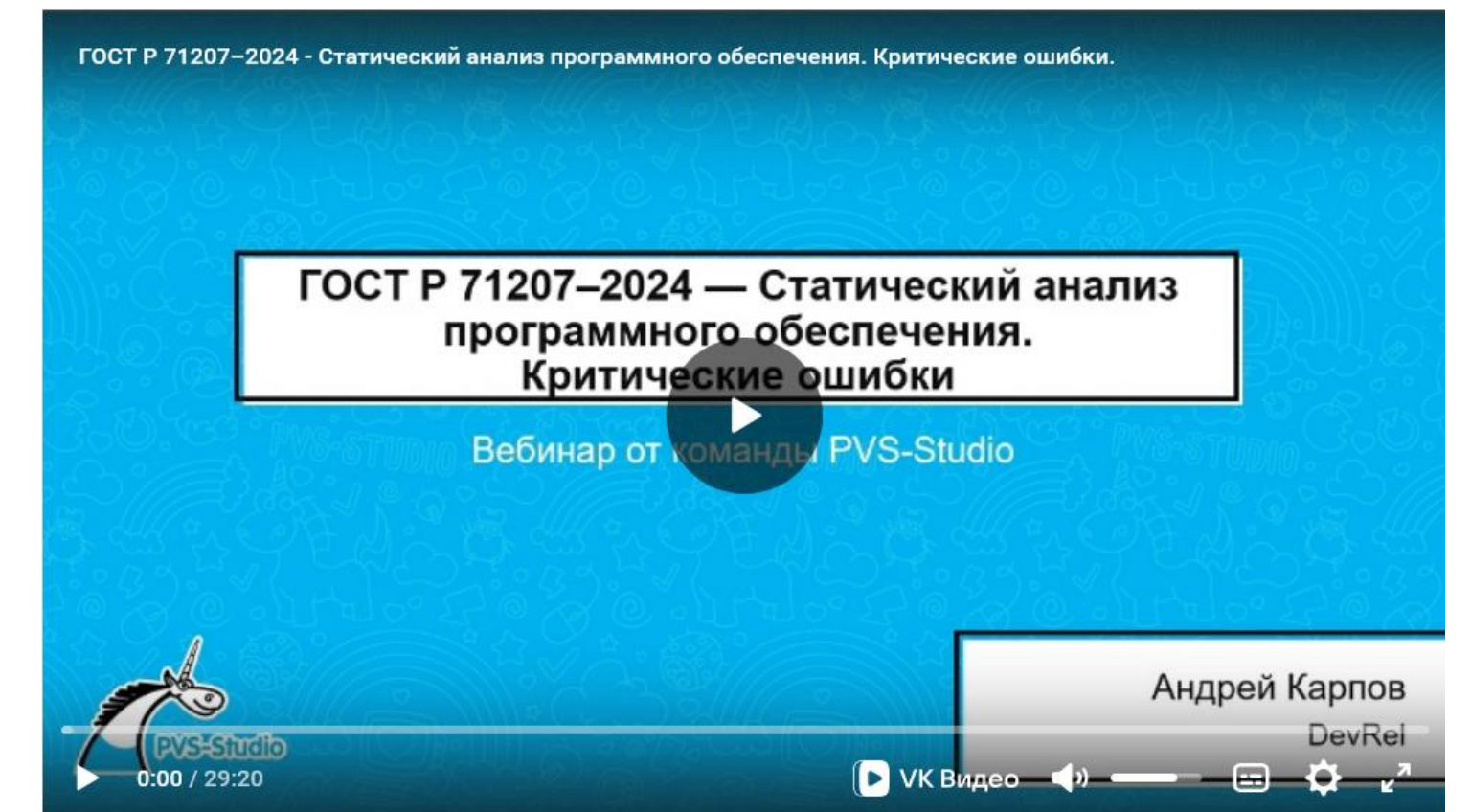
13

- **ГОСТ Р 71207-2024 – Статический анализ программного обеспечения вводит термин: критические ошибки**
- Это ошибки, которые с наибольшей вероятностью приводят к уязвимостям ПО
  - Если ошибка критическая, это не значит, что она приводит к чему-то плохому
  - Если ошибка не критическая, это не значит, что она не может оказаться уязвимостью
  - Надо править все ошибки. Суть — куда обратить больше внимания и с чего начинать

# ГОСТ Р 71207-2024: критические ошибки

14

- Ошибки управления динамической памятью
- Ошибки использования форматной строки
- Ошибки использования  
неинициализированных переменных
- Ошибки непроверенного использования  
чувствительных данных
- И т.д.



Вебинар про критические ошибки  
[pvs-studio.ru/ru/blog/video/11084/](https://pvs-studio.ru/ru/blog/video/11084/)



# Стандарт MISRA – это не только про ошибки, но и про ограничение сложности

15

- В основном за счёт ограничений на использование конструкций языка, библиотек, подходов программирования
- Последние версии:
  - MISRA-C – 2025
  - MISRA-C++ – 2023



# Познакомившись с MISRA в 2016 году, мы решили, что это не наше направление

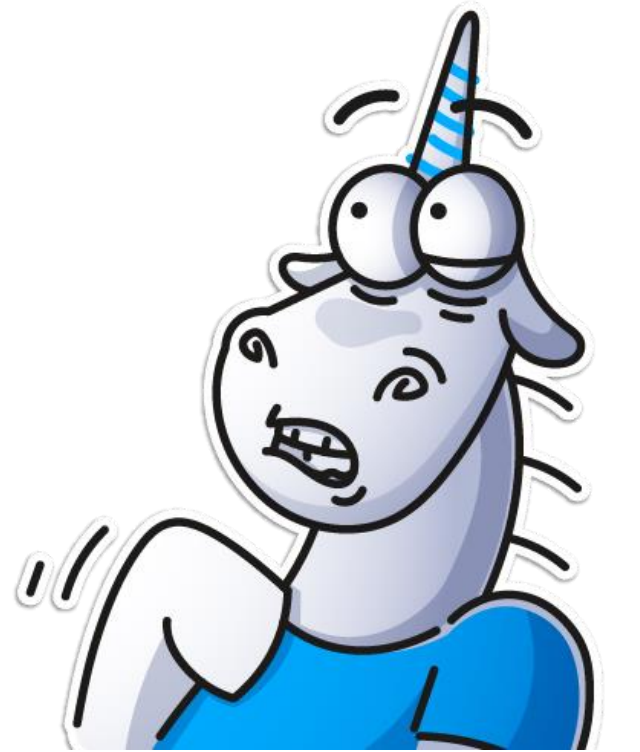
16

- Из статьи «Краткий ответ про MISRA» (2016)
  - Мы ориентированы на поиск уже имеющихся в коде ошибок, а не на предотвращение потенциальных проблем ценой ограничения программистов.
  - Например, мы не заставляем разработчика обязательно делать ветку default в switch.

- Мы считаем малополезной рекомендацию не писать комментарии вида `/* */`, а использовать `//`.
- Подобные диагностики очень быстро "замусоривают" вывод анализатора, и вместо поиска настоящих ошибок человек начинает бороться с тысячами хоть и теоретически хороших, но ни на что не влияющих предложений по улучшению кода.



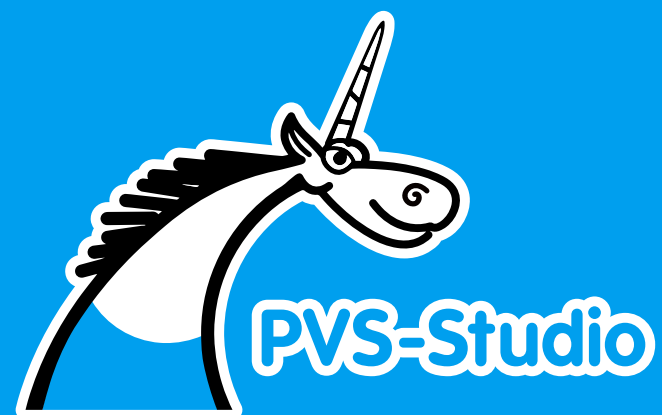
- Про MISRA-детекторы сложно писать статьи для продвижения
  - Нельзя написать статью про «ужас, они используют goto!»
- Мы позиционировали PVS-Studio как инструмент для внедрения в существующие проекты, где статический анализ ещё не применяется
- Очень трудоёмко внедрить MISRA в уже существующий долгое время проект



- Мы искали критические ошибки  
(только называли их потенциальными уязвимостями)
- Искали опечатки
- Искали частные случаи неопределённого поведения  
(в обобщённом виде это задача не решается)
- И писали про них статьи  
[pvs-studio.ru/ru/blog/inspections/](https://pvs-studio.ru/ru/blog/inspections/)



Но MISRA – это больше не про  
поиск проблем, а про их  
предупреждение





# MISRA C/C++ тоже служит целям РБПО

21

- Код будет легче читаться и поддерживаться
- Код не будет содержать редких и нестандартных конструкций
- Снижается цикломатическая сложность
- Код более переносимый

# Устраняет нестыковку сущностей

22

- Если функция не возвращает управление, то странно, что она возвращает значение
- Неправильно:

```
_Noreturn int sum(int a, int b)
{
    . . . .
    return a + b;
}
```

# Меньше вероятность неправильного понимания кода

23

- Последовательности восьмеричных и шестнадцатеричных чисел внутри строковых и символьных литералов должны быть завершёнными

```
const char* str = "\x0exit";           // Неправильно
```

```
const char* str1 = "\x0" "exit";       // Правильно
```

- Нельзя использовать variable-length array (VLA)
- Опасность переполнения стека и потенциально более уязвимый код
- Неправильно:

```
void foo(size_t n)
{
    int arr[n];
    // ...
}
```



- Использование оператора `goto`, осуществляющего переход к метке, находящейся выше по коду, ухудшает читаемость кода и, как следствие, усложняет его поддержку
- Не получится создать цикл с помощью `goto`
- Неправильно:

```
void init(...) {  
    again:  
        ...  
        if (...)  
            goto again;  
        ...  
}
```

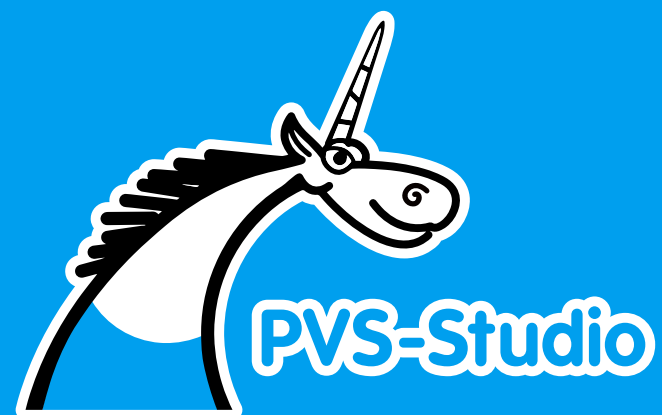
# Выявление аномалий, например, мёртвого кода

26

```
void set_duration_style(duration_style style)
{
    duration_style_ == style;
}
```

- Может быть как **просто лишним**
- Так и **скрывать ошибку**, как в рассмотренном примере
- Эту ошибку мы, кстати, нашли в реальном коде библиотеки Boost

# Что важно понимать и помнить про MISRA



- Попытка использовать MISRA-правила в мире прикладного ПО вызовет недоумение и неудобства
- Многие рекомендации будут бесполезны или вредны
- Пример: предупреждение про использование функций и операторов выделения памяти (malloc, realloc, new, ...)



# Будет крайне трудоёмко адаптировать проект под MISRA

29

- Уже упоминал ранее, но ещё раз
- MISRA предполагает, что вы сразу пишете код с учётом требований стандарта



# Не все правила MISRA могу выявляться анализаторами кода

30

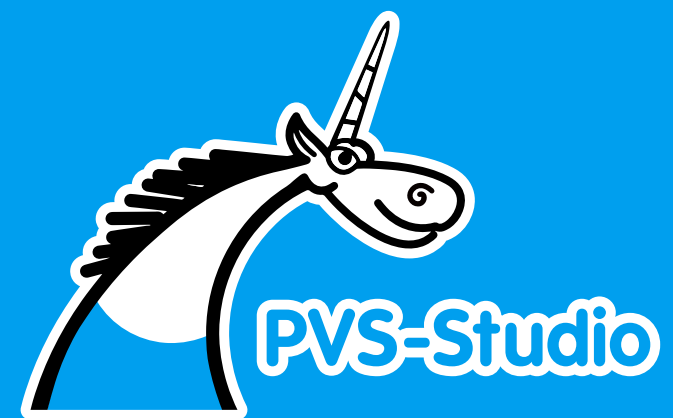
- Некоторые правила высокоуровневые или обобщённые
- Нет анализаторов, которые на 100% покрывают MISRA
- Поэтому пишут:  
`X MISRA checker covers 100% of all automatically testable MISRA rules`
- Следствие: часть работы по соблюдению правил MISRA будет всё равно выполняться людьми на обзорах кода

# Многие MISRA-правила не абсолютны

31

- Их можно нарушать
- Иначе многие программы просто не написать
- Но каждое отхождение от стандарта должны быть объяснено и задокументировано (например, комментарием)

# PVS-Studio и MISRA



## Классификация предупреждений PVS-Studio согласно стандартам: MISRA C, MISRA C++

- MISRA C 2012
- MISRA C 2023
- MISRA C++ 2008

MISRA C и MISRA C++ — это стандарты разработки программного обеспечения, созданные организацией MISRA (Motor Industry Software Reliability Association). Цель стандартов: улучшение безопасности, переносимости и надёжности программ для встраиваемых систем.

Анализатор ориентирован на версии стандартов: MISRA C:2012, MISRA C:2023 и MISRA C++:2008.

### MISRA C 2012

Поддержка стандарта MISRA C 2012:

- процент покрытия Mandatory + Required: 85%;
- процент покрытия Mandatory + Required + Advisory: 78%.

Error Code	Error Description	Mapping
V2501	Octal constants should not be used.	Rule 7.1
V2502	The 'goto' statement should not be used.	Rule 15.1
V2503	Implicitly specified enumeration constants should be unique – consider specifying non-unique constants explicitly.	Rule 8.12



[pvs-studio.ru/ru/pvs-studio/sast/misra/](https://pvs-studio.ru/ru/pvs-studio/sast/misra/)



- Процент покрытия MISRA C++ 2008
  - Required + Advisory: 32%
- Процент покрытия MISRA C 2012
  - Mandatory + Required: 85%
  - Mandatory + Required + Advisory: 78%
- **Процент покрытия MISRA C 2023**
  - Mandatory + Required: 82%
  - Mandatory + Required + Advisory: 74%

# Поддержка генерации отчёта MISRA Compliance для MISRA

35

- MISRA Compliance — это стандарт, который позволяет понять, соответствует ли проект стандарту MISRA C / C++ с учётом всех отклонений и рекатегоризаций



# Генерация отчёта MISRA Compliance в PVS-Studio

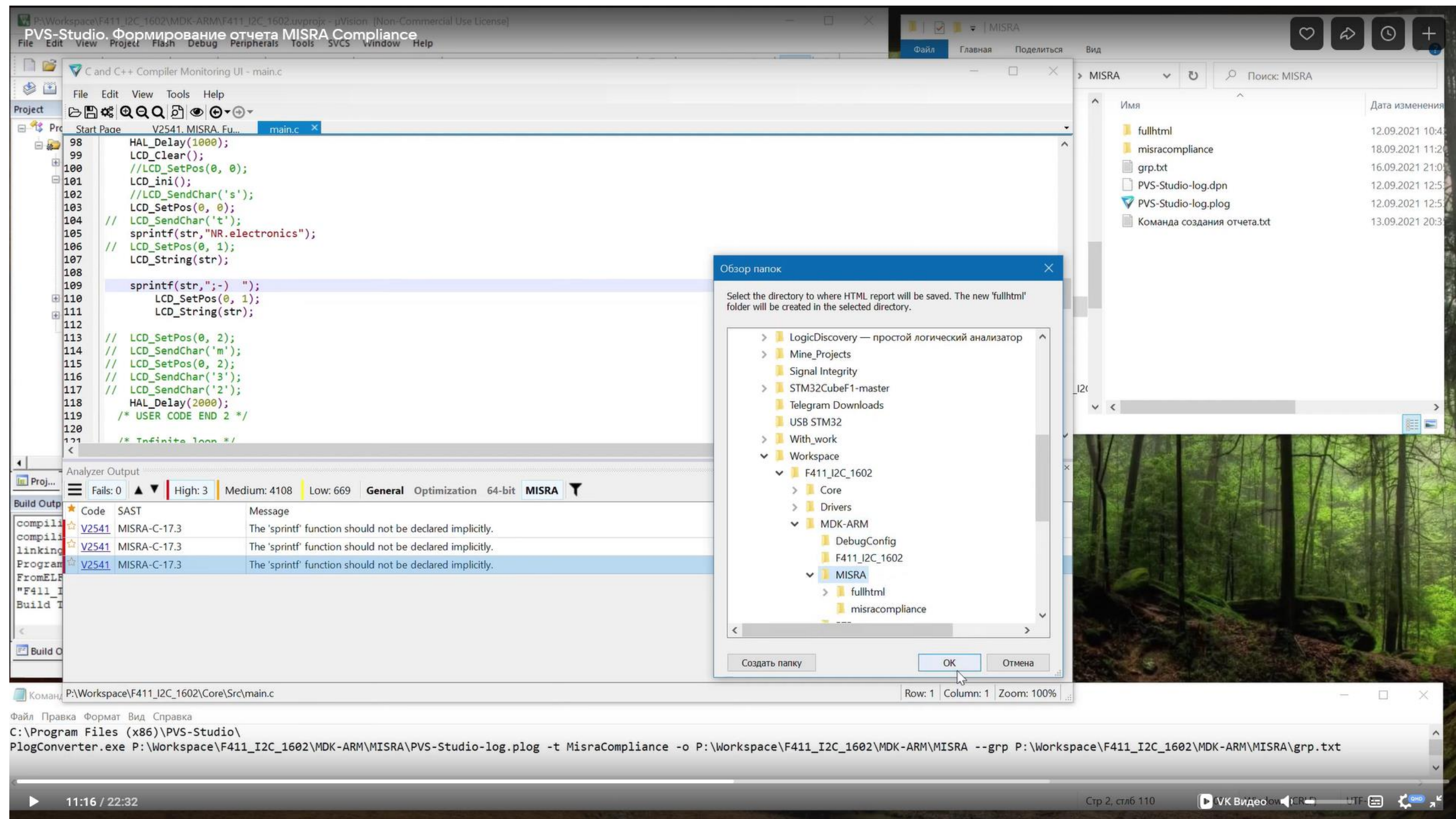
36

- Раздел документации PVS-Studio:  
MISRA Coding Standards and Compliance  
<https://pvs-studio.ru/ru/docs/manual/6966/>
- Пример запуска PlogConverter.exe:
  - "C:\Program Files (x86)\PVS-Studio\PlogConverter.exe" "path\_to\_report\_file" ^ -t MisraCompliance -o "path\_to\_MISRA\_report" --grp "path\_to\_grp.txt"





# Electronics Lab: PVS-Studio. Формирование отчёта MISRA Compliance



[vk.com/video537164717\\_456239979](https://vk.com/video537164717_456239979)

# Пример отчёта, когда проект соответствует / не соответствует MISRA C 2012



### MISRA Guideline Compliance Summary


Guidelines: MISRA C 2012

Checking tool: PVS-Studio

Result: **Compliant**

Summary: There were violations of 1 advisory guideline. There were deviations of 1 required guideline, 1 advisory guideline.

Guideline	Category	Recategorication	Compliance
Rule 13.6	Mandatory		Compliant
Rule 14.1	Required		Compliant
Rule 14.2	Required		Not Supported
Rule 14.3	Required		Compliant
Rule 14.4	Required		Compliant
Rule 15.1	Advisory		Compliant
Rule 15.2	Required		Compliant
Rule 15.3	Required		Compliant
Rule 15.4	Advisory		Compliant
Rule 15.5	Advisory		Compliant
Rule 15.6	Required		Compliant
Rule 15.7	Required		Compliant
Rule 16.1	Required		Not Supported
Rule 16.2	Required		Compliant
Rule 16.3	Required		Compliant
Rule 16.4	Required		Compliant
Rule 16.5	Required		Compliant
Rule 16.6	Required		Compliant
Rule 16.7	Required		Compliant
Rule 17.1	Required		Not Supported
Rule 17.2	Required		Compliant
Rule 17.3	Mandatory		Compliant



### MISRA Guideline Compliance Summary

Guidelines: MISRA C 2012

Checking tool: PVS-Studio

Result: **Not compliant**

Summary: There were violations of 5 mandatory guidelines, 29 required guidelines, 13 advisory guidelines. There were deviations of 1 required guideline, 1 advisory guideline.

Guideline	Category	Recategorication	Compliance
Rule 14.1	Required		Violations (14)
Rule 14.2	Required		Not Supported
Rule 14.3	Required		Compliant
Rule 14.4	Required		Compliant
Rule 15.1	Advisory		Violations (462)
Rule 15.2	Required		Violations (94)
Rule 15.3	Required	Mandatory	Violations (6)
Rule 15.4	Advisory		Violations (127)
Rule 15.5	Advisory		Violations (2647)
Rule 15.6	Required		Deviations (52)
Rule 15.7	Required		Violations (669)
Rule 16.1	Required		Not Supported
Rule 16.2	Required		Compliant
Rule 16.3	Required		Violations (175)
Rule 16.4	Required	Mandatory	Violations (178)
Rule 16.5	Required		Violations (4)
Rule 16.6	Required		Violations (13)
Rule 16.7	Required		Compliant
Rule 17.1	Required		Not Supported
Rule 17.2	Required		Violations (65)
Rule 17.3	Mandatory		Violations (2)
Rule 17.4	Mandatory		Violations (1)
Rule 17.5	Advisory	Required	Compliant
Rule 17.6	Mandatory		Compliant



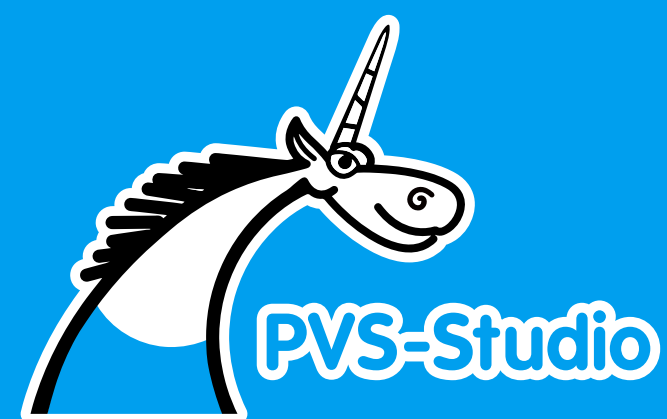
Сделай свой проект чистым  
и безопасным вместе  
с PVS-Studio



FuncSafety  
Консалтинг в области  
Функциональной Безопасности



# Дополнительные материалы



- Что такое MISRA и как её готовить  
[pvs-studio.ru/ru/blog/posts/cpp/0702/](https://pvs-studio.ru/ru/blog/posts/cpp/0702/)
- PVS-Studio соответствует требованиям ГОСТ Р 71207-2024 (статический анализ программного обеспечения)  
[pvs-studio.ru/ru/blog/posts/1204/](https://pvs-studio.ru/ru/blog/posts/1204/)
- Необходимость статического анализа для РБПО на примере 190 багов в TDengine  
[pvs-studio.ru/ru/blog/posts/cpp/1249/](https://pvs-studio.ru/ru/blog/posts/cpp/1249/)

- Интеграция PVS-Studio в PlatformIO  
[pvs-studio.ru/ru/blog/posts/cpp/0714/](https://pvs-studio.ru/ru/blog/posts/cpp/0714/)
- Исследуем качество кода операционной системы Zephyr  
[pvs-studio.ru/ru/blog/posts/0721/](https://pvs-studio.ru/ru/blog/posts/0721/)
- Espressif IoT Development Framework: 71 выстрел в ногу  
[pvs-studio.ru/ru/blog/posts/cpp/0790/](https://pvs-studio.ru/ru/blog/posts/cpp/0790/)

- Цикл «Вокруг РБПО за 25 вебинаров: ГОСТ Р 56939-2024»  
[pvs-studio.ru/ru/webinar/rbpo/](https://pvs-studio.ru/ru/webinar/rbpo/)
- Статический анализ C++ кода по ГОСТ Р 71207-2024 на примере PVS-Studio  
[pvs-studio.ru/ru/blog/video/11236/](https://pvs-studio.ru/ru/blog/video/11236/)
- Интеграция PVS-Studio с комплектом разработчика Нейтрино  
[pvs-studio.ru/ru/blog/video/11355/](https://pvs-studio.ru/ru/blog/video/11355/)



# Карпов Андрей Николаевич

44

- Карпов Андрей Николаевич, 1981
- ООО «ПВС», директор по развитию бизнеса
- Более 18 лет занимаюсь 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](#)

