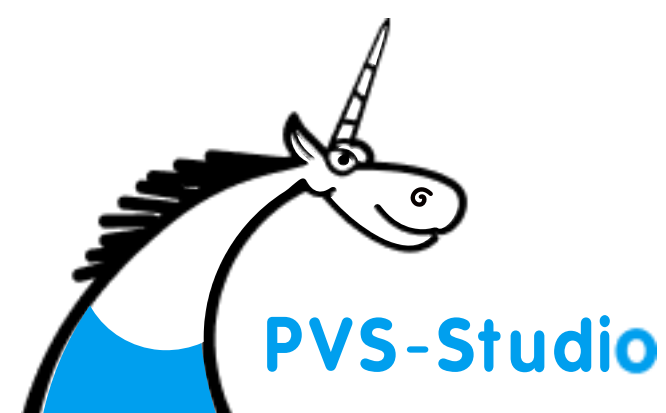


Статический анализ для обеспечения безопасности встраиваемых систем



Александра Уварова
Developer Advocate



Александра Уварова

Developer Advocate

- Разработчик C++ анализатора PVS-Studio
- Рассказываю про качество кода и безопасную разработку на конференциях
- Пишу технические и научные статьи



@AleksandraUvarova



Сегодня в программе

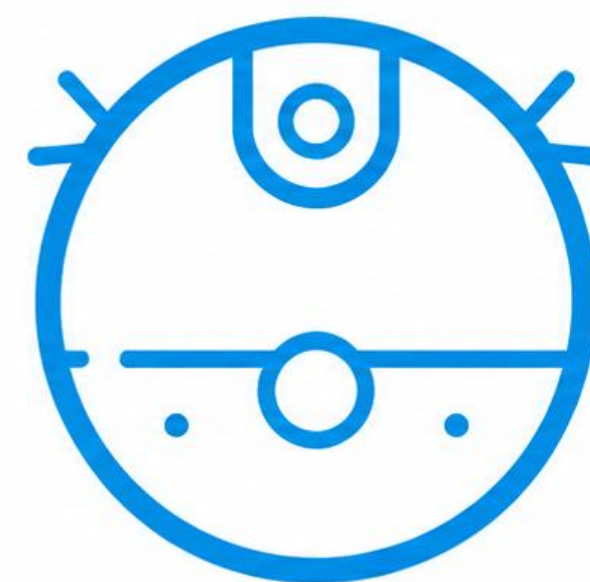
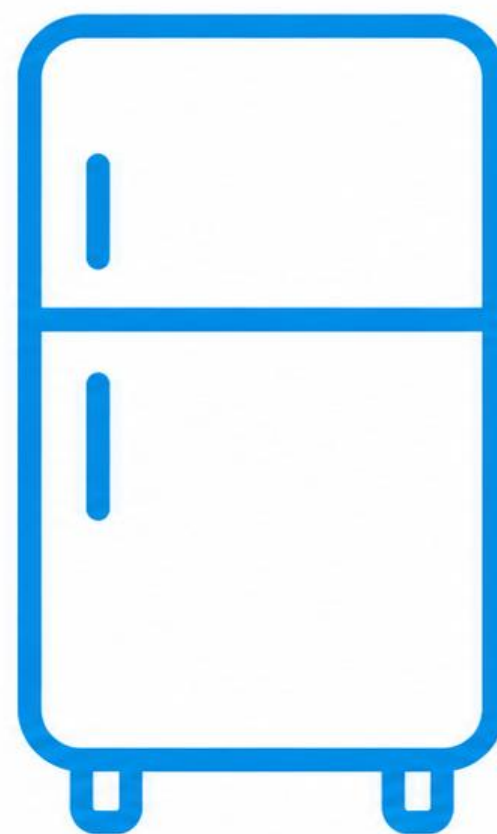
1. Стандарты МЭК 61508 и ИСО 26262
2. Стандарты кодирования MISRA
3. Статический анализ
4. PVS-Studio для обеспечения соответствие стандартам

ГОСТ Р МЭК 61508

Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью

ГОСТ Р МЭК 61508

Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью



ГОСТ Р МЭК 61508

Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью

Правильное выполнение функций
и предотвращение рисков

ГОСТ Р МЭК 61508

Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью

- Представлены требования к программному обеспечению
- Описаны все стадии жизненного цикла системы
- Введены понятия уровней полноты безопасности

Рекомендации по конкретным языкам программирования на разных уровнях полноты безопасности

- HR – Настоятельно рекомендуется
- R – Рекомендуется
- — – Сомнительно, но разрешимо
- NR – НЕ рекомендуется

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	УПБ1	УПБ2	УПБ3	УПБ4
1 ADA	HR	HR	R	R
2 ADA с подмножеством	HR	HR	HR	HR
3 Java	NR	NR	NR	NR
4 Java с подмножеством (без включения сборки мусора или с включением сборки мусора, которая не будет вызывать остановку прикладной программы в течение значительного промежутка времени). Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.	R	R	NR	NR
5 PASCAL (см. примечание 1)	HR	HR	R	R
6 PASCAL с подмножеством	HR	HR	HR	HR
7 FORTRAN 77	R	R	R	R
8 FORTRAN 77 с подмножеством	HR	HR	HR	HR
9 C	R	—	NR	NR
10 C с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
11 C++ (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	R	—	NR	NR
12 C++ с подмножеством и стандартом кодирования, а также использование инструментов статического анализа (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	HR	HR	HR	HR
13 Ассемблер	R	R	—	—
14 Ассемблер с подмножеством и стандартом кодирования	R	R	R	R
15 Многоступенчатые диаграммы	R	R	R	R
16 Многоступенчатая диаграмма с определенным подмножеством языка	HR	HR	HR	HR
17 Диаграмма функциональных блоков	R	R	R	R
18 Диаграмма функциональных блоков с определенным подмножеством языка	HR	HR	HR	HR
19 Структурированный текст	R	R	R	R
20 Структурированный текст с определенным подмножеством языка	HR	HR	HR	HR
21 Последовательная функциональная диаграмма	R	R	R	R
22 Последовательная функциональная диаграмма с определенным подмножеством языка	HR	HR	HR	HR
23 Список команд	R	—	NR	NR
24 Список команд с определенным подмножеством языка	HR	R	R	R

Рекомендации по конкретным языкам программирования на разных уровнях полноты безопасности

- HR – Настоятельно рекомендуется
- R – Рекомендуется
- — – Сомнительно, но разрешимо
- NR – НЕ рекомендуется

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	УПБ1	УПБ2	УПБ3	УПБ4
1 ADA	HR	HR	R	R
2 ADA с подмножеством	HR	HR	HR	HR
3 Java	NR	NR	NR	NR
4 Java с подмножеством (без включения сборки мусора или с включением сборки мусора, которая не будет вызывать остановку прикладной программы в течение значительного промежутка времени). Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.	R	R	NR	NR
5 PASCAL (см. примечание 1)	HR	HR	R	R
6 PASCAL с подмножеством	HR	HR	HR	HR
7 FORTRAN 77	R	R	R	R
8 FORTRAN 77 с подмножеством	HR	HR	HR	HR
9 C	R	—	NR	NR
10 C с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
11 C++ (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	R	—	NR	NR
12 C++ с подмножеством и стандартом кодирования, а также использование инструментов статического анализа (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	HR	HR	HR	HR

Рекомендации по конкретным языкам программирования на разных уровнях полноты безопасности

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	УПБ1	УПБ2	УПБ3	УПБ4
9 С	R	—	NR	NR
10 С с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
11 С++ (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	R	—	NR	NR
12 С++ с подмножеством и стандартом кодирования, а также использование инструментов статического анализа (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	HR	HR	HR	HR

Рекомендации по конкретным языкам программирования на разных уровнях полноты безопасности

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	УПБ1	УПБ2	УПБ3	УПБ4
9 С	R	—	NR	NR
10 С с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR
11 С++ (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	R	—	NR	NR
12 С++ с подмножеством и стандартом кодирования, а также использование инструментов статического анализа (Руководящие указания по использованию объектно-ориентированных средств см. в приложении G.)	HR	HR	HR	HR

ГОСТ Р ИСО 26262

- **Адаптация** стандартов МЭК 61508 для электронных систем в дорожных транспортных средствах

Цель: Предотвратить аварии и опасные ситуации, вызванные сбоями в работе электроники или программного обеспечения автомобиля



Какими средствами достигается цель?

Таблица 7

Методы верификации модуля программного обеспечения

Методы		УПБТС			
		A	B	C	D
1a	Сквозной контроль проекта <a>	++	+	o	o
1b	Дублированное программирование <a>	+	+	+	+
1c	Ревизия <a>	+	++	++	++
1d	Полуформальная верификация	+	+	++	++
1e	Формальная верификация	o	o	+	+
1f	Анализ потока управления , <c>	+	+	++	++
1g	Анализ потока данных , <c>	+	+	++	++
1h	Статический анализ кода <d>	++	++	++	++
1i	Статический анализ, основанный на абстрактной интерпретации <e>	+	+	+	+
1j	Тестирование на основе требований <f>	++	++	++	++
1k	Тестирование интерфейса <g>	++	++	++	++
1l	Тестирование методом внесения дефектов <h>	+	+	+	++
1m	Оценка используемых ресурсов <i>	+	+	+	++
1n	Сравнительный тест между моделью и кодом, если он применим <j>	+	+	++	++

Какими средствами достигается цель?

Методы верификации модуля программного обеспечения

Методы		УПБТС		
		A	B	
1h	Статический анализ кода <d>	++	++	
1i	Статический анализ, основанный на абстрактной интерпретации <e>	+	+	

Какими средствами достигается цель?

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	SIL1	SIL2	SIL3	SIL4
1 ADA	HR	HR	R	R
2 ADA с подмножеством	HR	HR	HR	HR
3 MODULA-2	HR	HR	R	R
4 MODULA- с подмножеством	HR	HR	HR	HR
5 PASCAL	HR	HR	R	R
6 PASCAL с подмножеством	HR	HR	HR	HR
7 FORTRAN 77	R	R	R	R
8 FORTRAN 77 с подмножеством	HR	HR	HR	HR
9 C	R	—	NR	NR
10 Язык C с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR

Какими средствами достигается цель?

Т а б л и ц а С.1 — Рекомендации по конкретным языкам программирования

Язык программирования	SIL1	SIL2	SIL3	SIL4
1 ADA	HR	HR	R	R
2 ADA с подмножеством	HR	HR	HR	HR
3 MODULA-2	HR	HR	R	R
4 MODULA- с подмножеством	HR	HR	HR	HR
5 PASCAL	HR	HR	R	R
6 PASCAL с подмножеством	HR	HR	HR	HR
7 FORTRAN 77	R	R	R	R
8 FORTRAN 77 с подмножеством	HR	HR	HR	HR
9 C	R	—	NR	NR
10 Язык C с подмножеством и стандартом кодирования, а также использование инструментов статического анализа	HR	HR	HR	HR

Какими средствами достигается цель?

Вопросы, которые должны быть рассмотрены
в руководствах по моделированию и кодированию

Свойства		УПБТС			
		A	B	C	D
1a	Применение низкой сложности <a>	++	++	++	++
1b	Использование подмножеств языков 	++	++	++	++
1c	Применение строгой типизации <c>	++	++	++	++
1d	Использование методов безопасного программирования <d>	+	+	++	++
1e	Использование проверенных принципов проектирования <e>	+	+	++	++
1f	Использование однозначного графического представления	+	++	++	++
1g	Использование правил оформления	+	++	++	++
1h	Использование соглашений об именовании	++	++	++	++
1i	Аспекты параллельного выполнения <f>	+	+	+	+

Какими средствами достигается цель?

Применение низкой сложности <a>
Использование подмножеств языков
Применение строгой типизации <c>
Использование методов безопасного программирования <d>
Использование проверенных принципов проектирования <e>
Использование однозначного графического представления
Использование правил оформления
Использование соглашений об именовании
Аспекты параллельного выполнения <f>

Какими средствами достигается цель?

- Критерии выбора языков должны быть обеспечены соответствующими руководствами с учетом вопросов, перечисленных в таблице

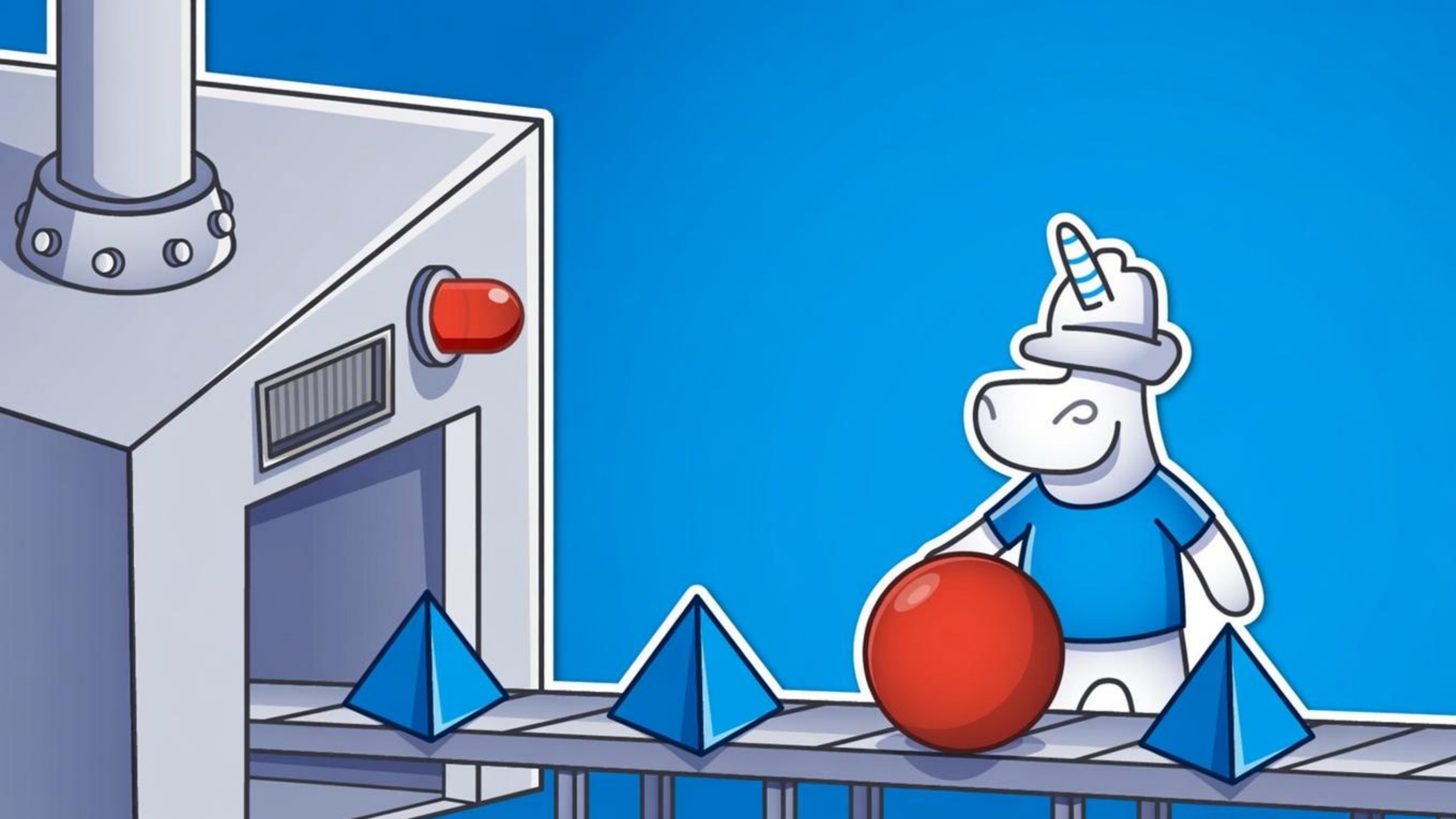
Например MISRA-C:2012

Стандарты MISRA

Руководство для создания безопасного программного обеспечения на языках С и С++ для областей, где требуется высокая надёжность ПО









Как стандарт помогает обеспечивать надежность?

Безопасность достигается не только за счёт **недопущения** ошибок и уязвимостей, но и путем **отказа** от сложных конструкций языка



Правила MISRA

Правила MISRA

- Операторы 'switch' должны быть корректно сформированы

`case-group` – один и более `case`

`switch-clause` – ветка `case`,
заканчивающаяся `break`;

```
switch (expression)
{
  case-group_1:
    switch-clause
  case-group_2:
    switch-clause
  // ....
  case-group_n:
    switch-clause
}
```

Операторы 'switch' должны быть корректно сформированы

- Два и более `case-group`
- Метка `default` в начале или в конце
- Декларации в `{}`

`case-group` – один и более `case`

`switch-clause` – ветка `case`,
заканчивающаяся `break`;

```
switch (expression)
{
  case-group_1:
    switch-clause
  case-group_2:
    switch-clause
  // ....
  case-group_n:
    switch-clause
}
```

Операторы 'switch' должны быть корректно сформированы

```
switch (n)
{
case 1:
default:
case 2: n++; int x = 0; x++; break;

case 3: { n++; break; }
}
```

Операторы 'switch' должны быть корректно сформированы

```
switch (n)
{
case 1:
default:
case 2: n++; int x = 0; x++; break;

case 3: { n++; break; }
}
```

- Неправильно расположена метка `default`

V2659. MISRA. Switch statements should be well-formed.

Операторы 'switch' должны быть корректно сформированы

```
switch (n)
{
case 1:
default:

case 2: n++; int x = 0; x++; break;

case 3: { n++; break; }
}
```

- Неправильно расположена метка `default`
- Декларация не в `{ }`

V2659. MISRA. Switch statements should be well-formed.

Операторы 'switch' должны быть корректно сформированы

```
switch (n)
{
case 10:
case 20: break;
}
```

- Отсутствует метка `default`
- Количество `case-group` меньше двух

V2659. MISRA. Switch statements should be well-formed.

Правила MISRA

- За параметром макроса, к которому применен оператор #, не должен сразу следовать оператор ##

```
#define my_print(a, b) printf("%s", #a ## b)
```

За параметром макроса, к которому применен оператор #, не должен сразу следовать оператор

```
#define my_print(a, b) printf("%s", #a ## b)
```

```
#define HELLO "hello"
```

```
#define WORLD "world"
```

```
#define HELLOWORLD "hello, world"
```

```
int main(int argc, char *argv[])  
{  
    my_print(HELLO, WORLD);  
    return 0;  
}
```

"HELLO"WORLD

V2668. MISRA. The macro parameter immediately following a '#' operator should not immediately be followed by a '##' operator

За параметром макроса, к которому применен оператор #, не должен сразу следовать оператор

```
#define my_print(a, b) printf("%s", #a ## b)
```

```
#define HELLO "hello"
```

```
#define WORLD "world"
```

```
#define HELLOWORLD "hello, world"
```

```
int main(int argc, char *argv[])
{
    my_print(HELLO, WORLD);
    return 0;
}
```

HELLOworld

V2668. MISRA. The macro parameter immediately following a '#' operator should not immediately be followed by a '##' operator

За параметром макроса, к которому применен оператор #, не должен сразу следовать оператор

```
#define my_print(a, b) printf("%s", a ## b)
```

```
#define HELLO "hello"
```

```
#define WORLD "world"
```

```
#define HELLOWORLD "hello, world"
```

```
int main(int argc, char *argv[])  
{  
    my_print(HELLO, WORLD);  
    return 0;  
}
```

HELLOWORLD

V2668. MISRA. The macro parameter immediately following a '#' operator should not immediately be followed by a '##' operator

За параметром макроса, к которому применен оператор #, не должен сразу следовать оператор

```
#define my_print(a, b) printf("%s", a ## b)
```

```
#define HELLO "hello"
```

```
#define WORLD "world"
```

```
#define HELLOWORLD "hello, world"
```

```
int main(int argc, char *argv[])  
{  
    my_print(HELLO, WORLD);  
    return 0;  
}
```

hello, world

V2668. MISRA. The macro parameter immediately following a '#' operator should not immediately be followed by a '##' operator

Правила MISRA

- Результат выполнения функций 'std::remove', 'std::remove_if', 'std::unique', 'std::empty' или функции-члена 'empty' контейнера должен быть обязательно использован в коде

```
void remove_odd(std::vector<int> &v)
{
    std::remove_if(v.begin(), v.end(), /*...*/ );
}
```

Результат выполнения функций должен быть обязательно использован в коде

```
void remove_odd(std::vector<int> &v)
{
    std::remove_if(v.begin(), v.end(),
        [](auto item)
        {
            return (item & 1) != 0;
        });
}
```

2 4 5 6 8

Результат выполнения функций должен быть обязательно использован в коде

```
void remove_odd(std::vector<int> &v)
{
    std::remove_if(v.begin(), v.end(),
        [](auto item)
        {
            return (item & 1) != 0;
        });
}
```

2 4 6 8 5

V2676. MISRA. The result of 'std::remove', 'std::remove_if', 'std::unique' and 'empty' should be used.

Результат выполнения функций должен быть обязательно использован в коде

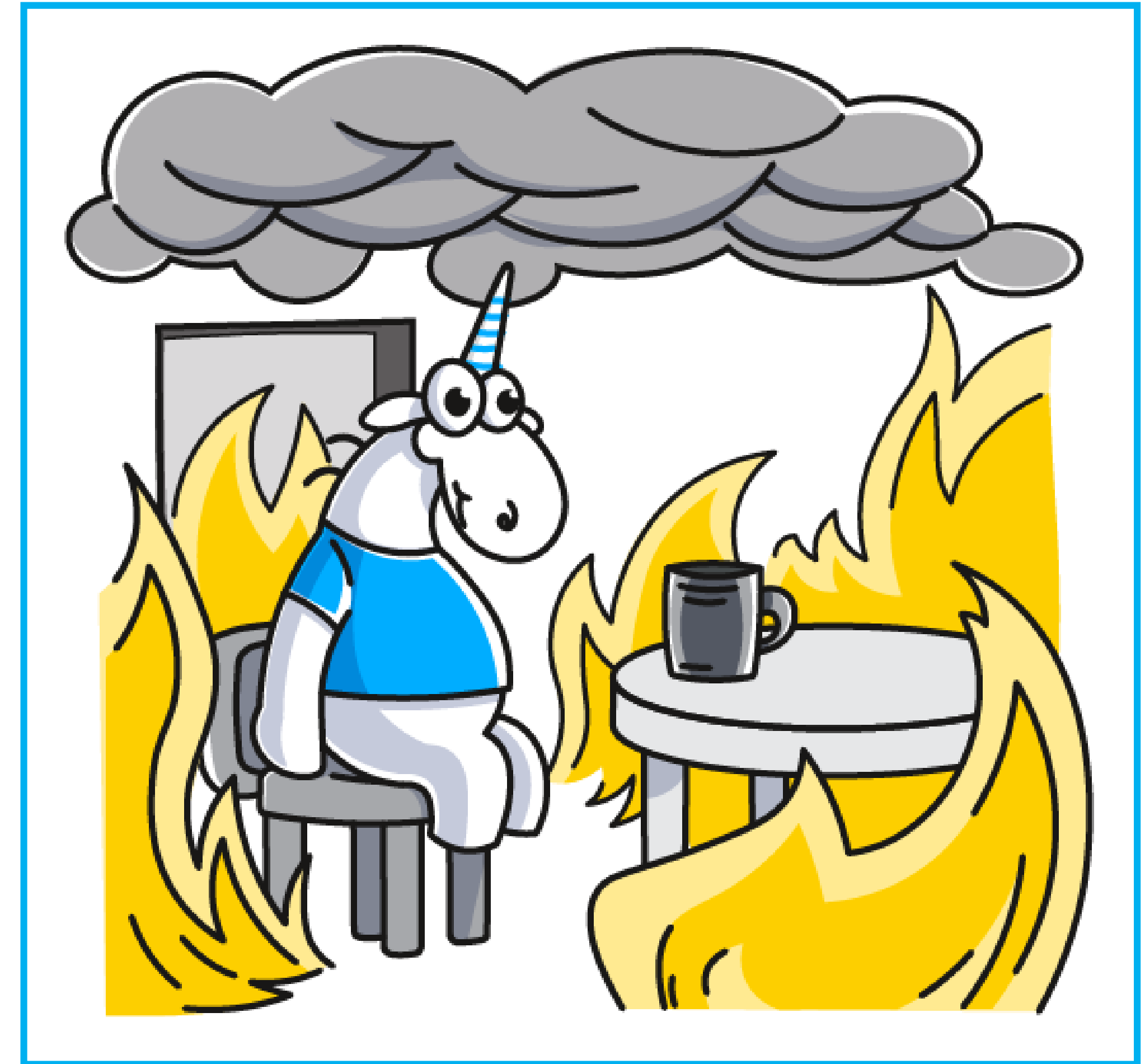
```
void remove_odd(std::vector<int> &v)
{
    auto it = std::remove_if(v.begin(), v.end(),
        [](auto item)
        {
            return (item & 1) != 0;
        });
    v.erase(it, v.end());
}
```

2 4 6 8

V2676. MISRA. The result of 'std::remove', 'std::remove_if', 'std::unique' and 'empty' should be used.

Правила MISRA

- Операторы 'switch' должны быть корректно сформированы
- За параметром макроса, к которому применен оператор '#', не должен сразу следовать оператор '##'
- Результат выполнения функций должен быть обязательно использован в коде
- И многое другое



PVS-Studio							
						High: 1	Medium: 8
						Low: 5	General
						Optimization	64-bit
						Custom	MISRA
						AUTOSAR	OWASP
★	Code	SAST	Message	Project	Position		
★	V2507	MISRA C 15.6	The body of the 'if' statement should be enclosed in braces.	abyss	channel.c:62		
★	V2571	MISRA C 11.5	Pointer to void should not be converted to a pointer to object. Consider inspecting the second argument of the 'cst_cg_write_2d_array' function call.	flite	cst_cg_...e.c:288		
★	V2511	MISRA C 21.3	The function with the 'malloc' name should not be used.	abyss	socket.c:38		
★	V2563	MISRA C++ 5-0-15	Array indexing should be the only form of pointer arithmetic and it should be applied only to objects defined as an array type.	FreeSwitchCoreLib	switch_utils.h:685		
★	V2519	MISRA C 16.4	The 'default' label is missing in 'switch' statement.	abyss	conn.c:97		
★	V2544	MISRA C 10.1	The 'bytesReadThisTime' operand of the ' ' operator should not have the essential 'signed' type.	abyss	conn.c:579		
★	V2544	MISRA C 10.1	The left and right operands of the ' ' operator should not have essential 'signed' and 'signed' types respectively. Consider inspecting operands: left - '0x0080', right - '0x0100'	abyss	file.c:115		

Классификация согласно стандарту MISRA

Классификация согласно стандарту MISRA

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

MISRA Compliance

- подтверждение соответствию стандарту MISRA C и/или MISRA C++ с учётом всех отклонений и категоризаций



<https://pvs-studio.ru/ru/pvs-studio/sast/misra/>

Какими средствами достигается цель?

Методы верификации модуля программного обеспечения

Методы		УПБТС		
		A	B	
1h	Статический анализ кода <d>	++	++	
1i	Статический анализ, основанный на абстрактной интерпретации <e>	+	+	

Виды статического анализа

«d» **Статический анализ.** Например поиск текста исходного кода или модели **шаблонов**, соответствующих известным сбоям, или соблюдения руководства по кодированию

«e» **Статический анализ, основанный на абстрактной интерпретации**, является собирательным термином для **расширенного статического анализа** который включает в себя такие виды анализа, как расширение **дерева синтаксического разбора** компилятора путем добавления **семантической информации**, которая может проверяться на предмет нарушения определенных правил (например, при возникновении проблем с типами данных, при появлении неинициализированных переменных), формирование **графа потока управления и анализ потока данных** (например, для обнаружения сбоев, связанных с гонками потоков и взаимоблокировками, ошибками в указателях) или даже **метакомпиляция и абстрактная интерпретация** кода или модели.

Что такое статический анализ?

Что такое статический анализ?

- код-ревью



Что такое статический анализ?

- ЭТО **автоматическое** код-ревью

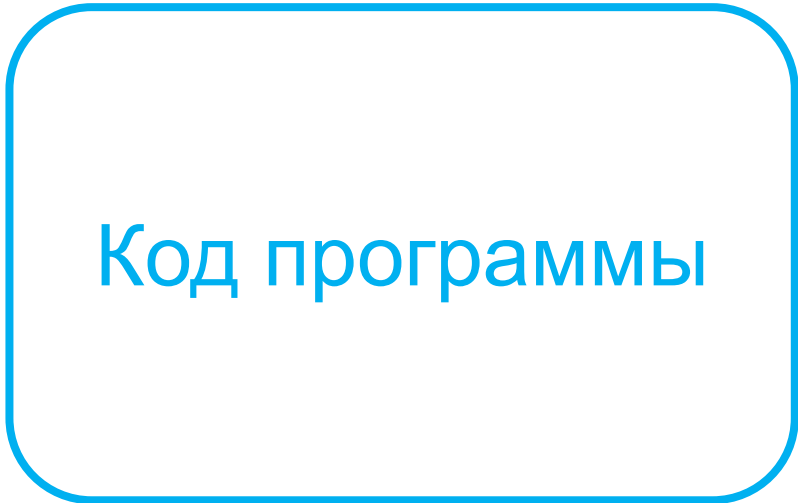


Что такое статический анализ?

- это автоматическое код-ревью
- без запуска программы

Что такое статический анализ?

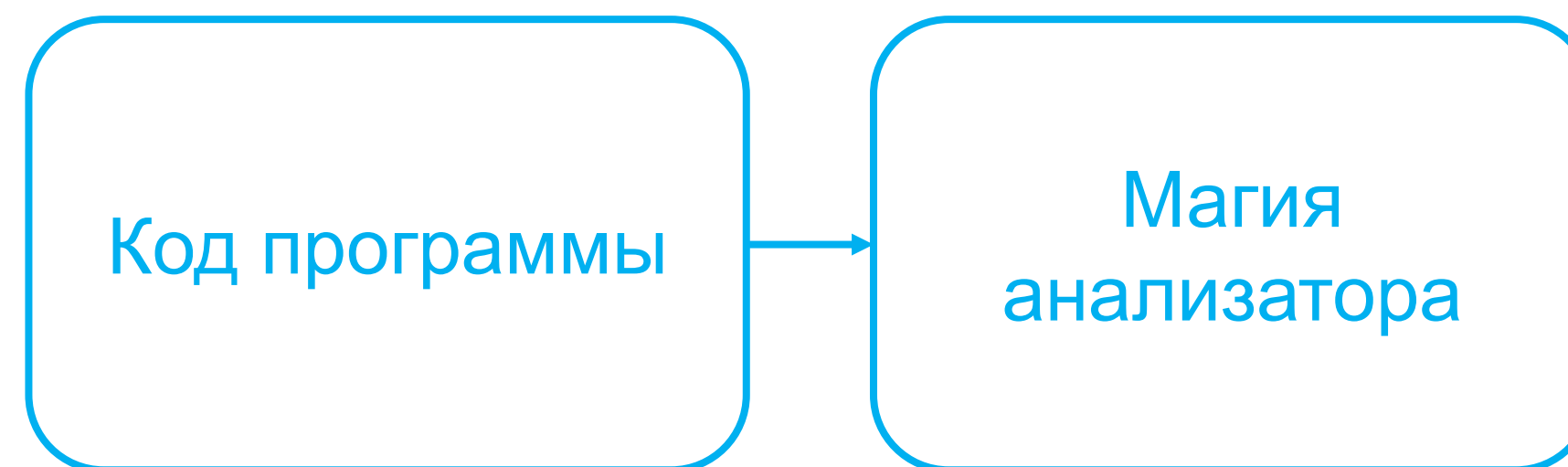
- это автоматическое код-ревью
- без запуска программы



Код программы

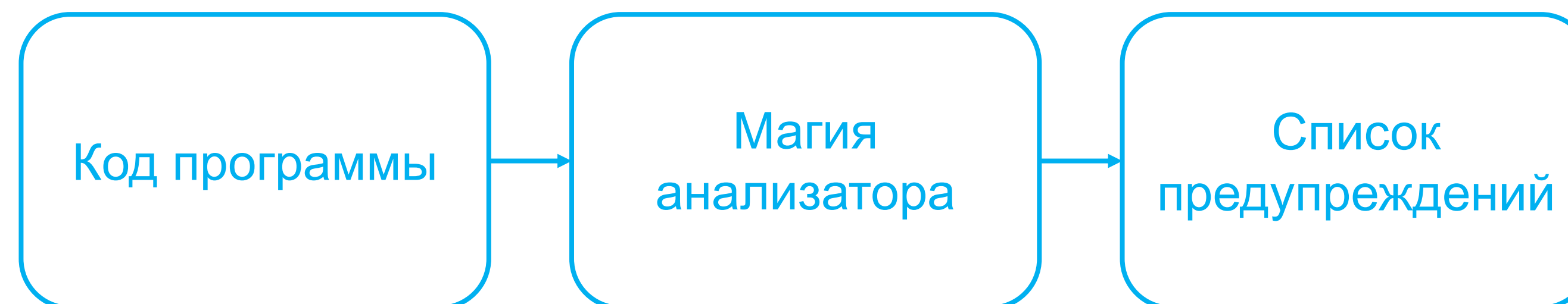
Что такое статический анализ?

- это автоматическое код-ревью
- без запуска программы



Что такое статический анализ?

- это автоматическое код-ревью
- без запуска программы



Магия анализатора

Как найти деление на ноль?

$$x / 0$$

Как найти деление на ноль?

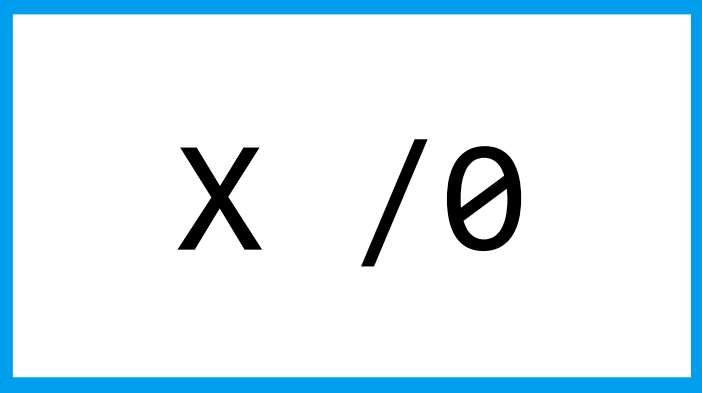
```
if some.Expr.contains("/ 0") then raise Error
```

$$X / 0$$

Как найти деление на ноль?

```
if some.Expr.contains("/ 0") then raise Error
```

```
if some.Expr.contains("/0 ") then raise Error
```



X / 0

Как найти деление на ноль?

```
if some.Expr.contains("/ 0") then raise Error
```

```
if some.Expr.contains("/0 ") then raise Error
```

```
if some.Expr.contains("/      0") then raise Error
```

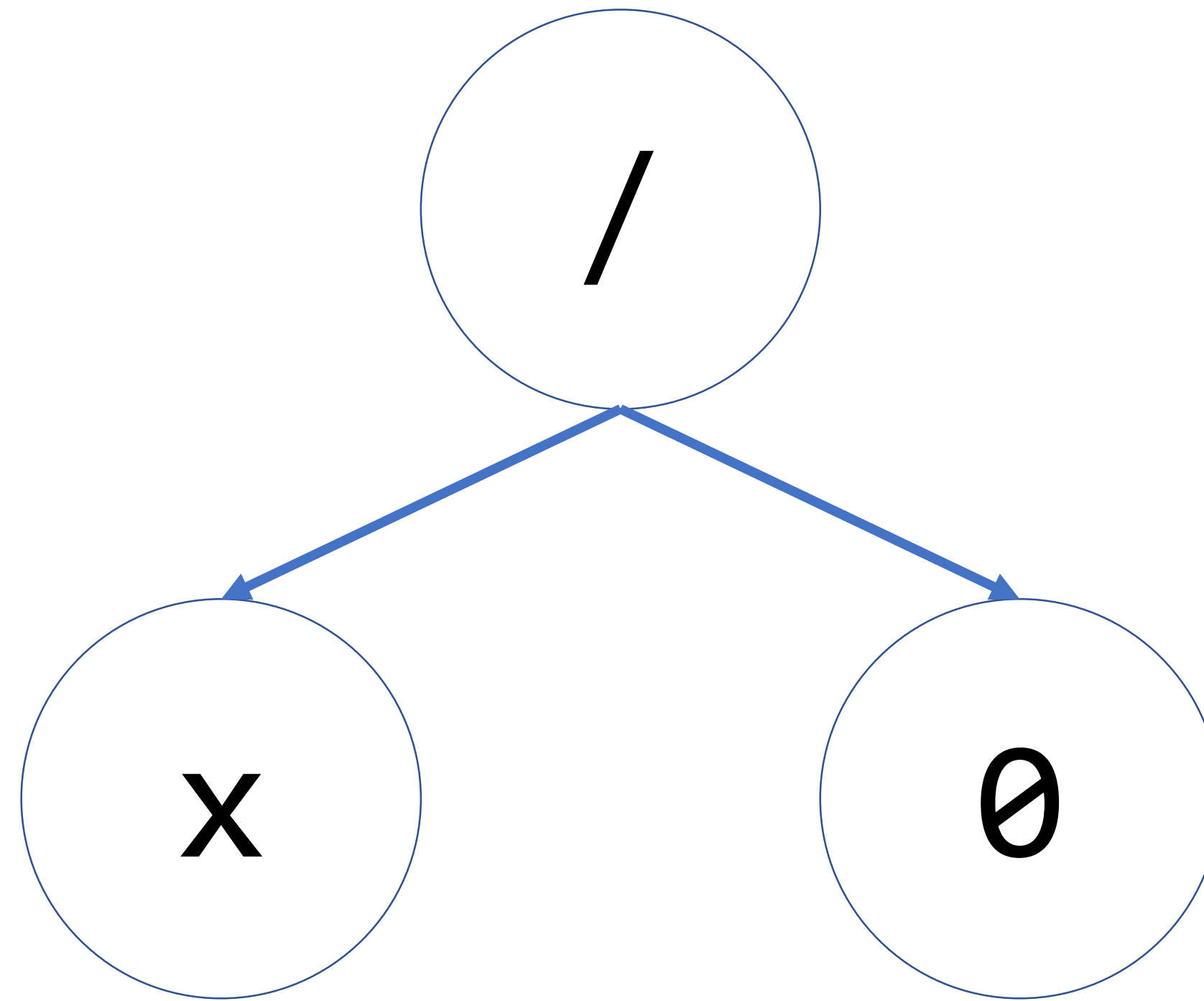


X / 0

Как найти деление на ноль?

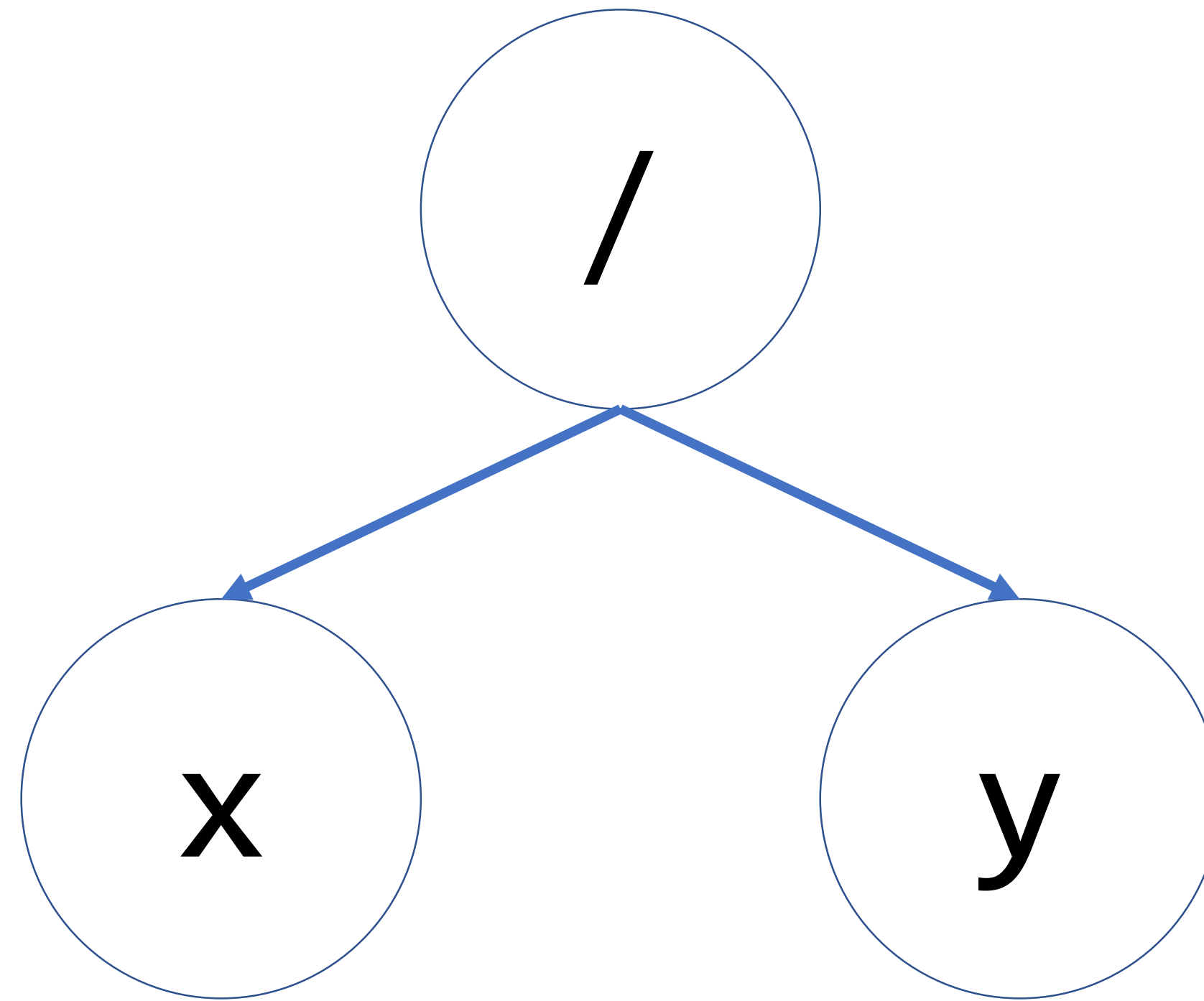
$$x / 0$$

Как найти деление на ноль?



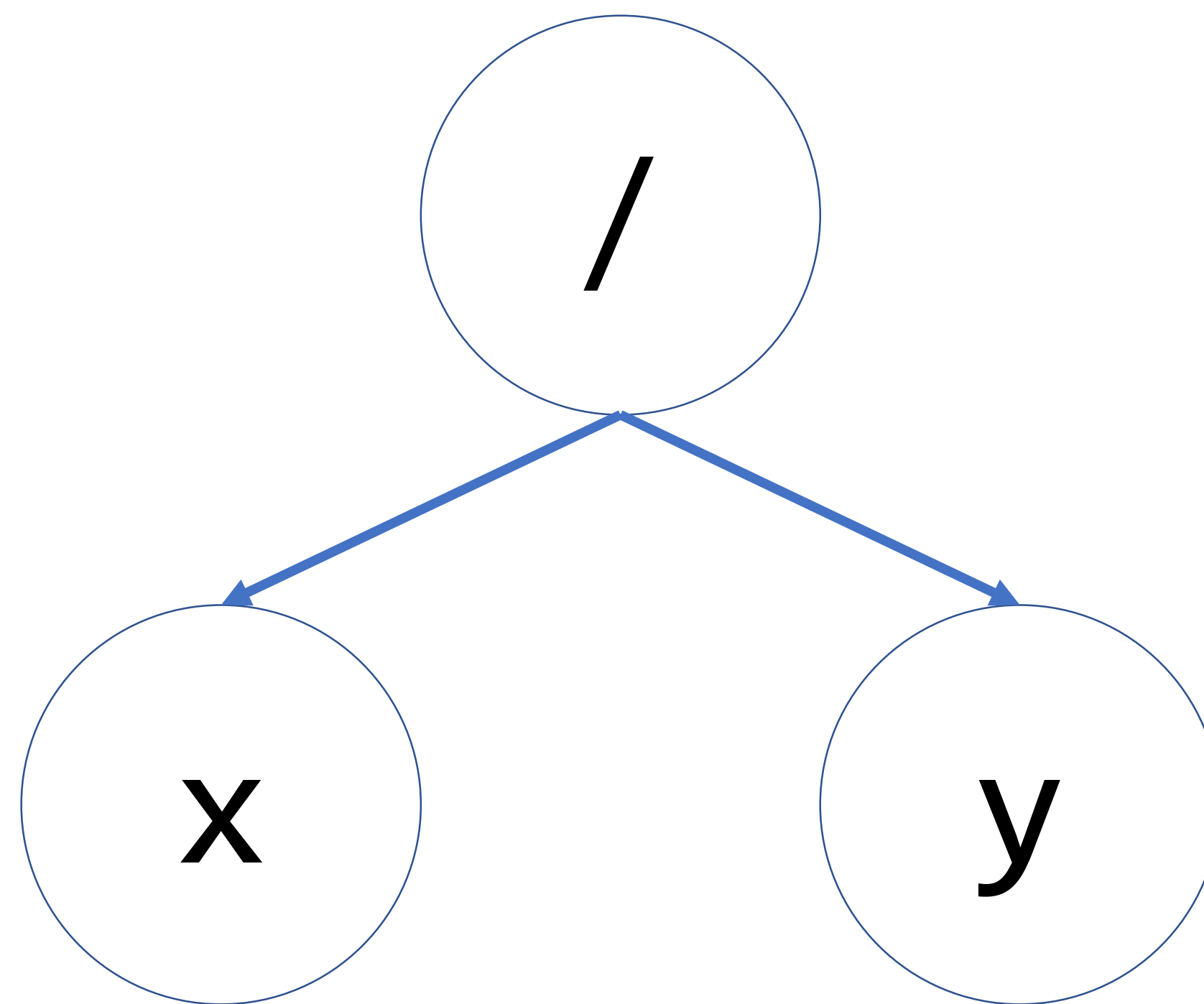
$$x / 0$$

Как найти деление на ноль?



$$x / y$$

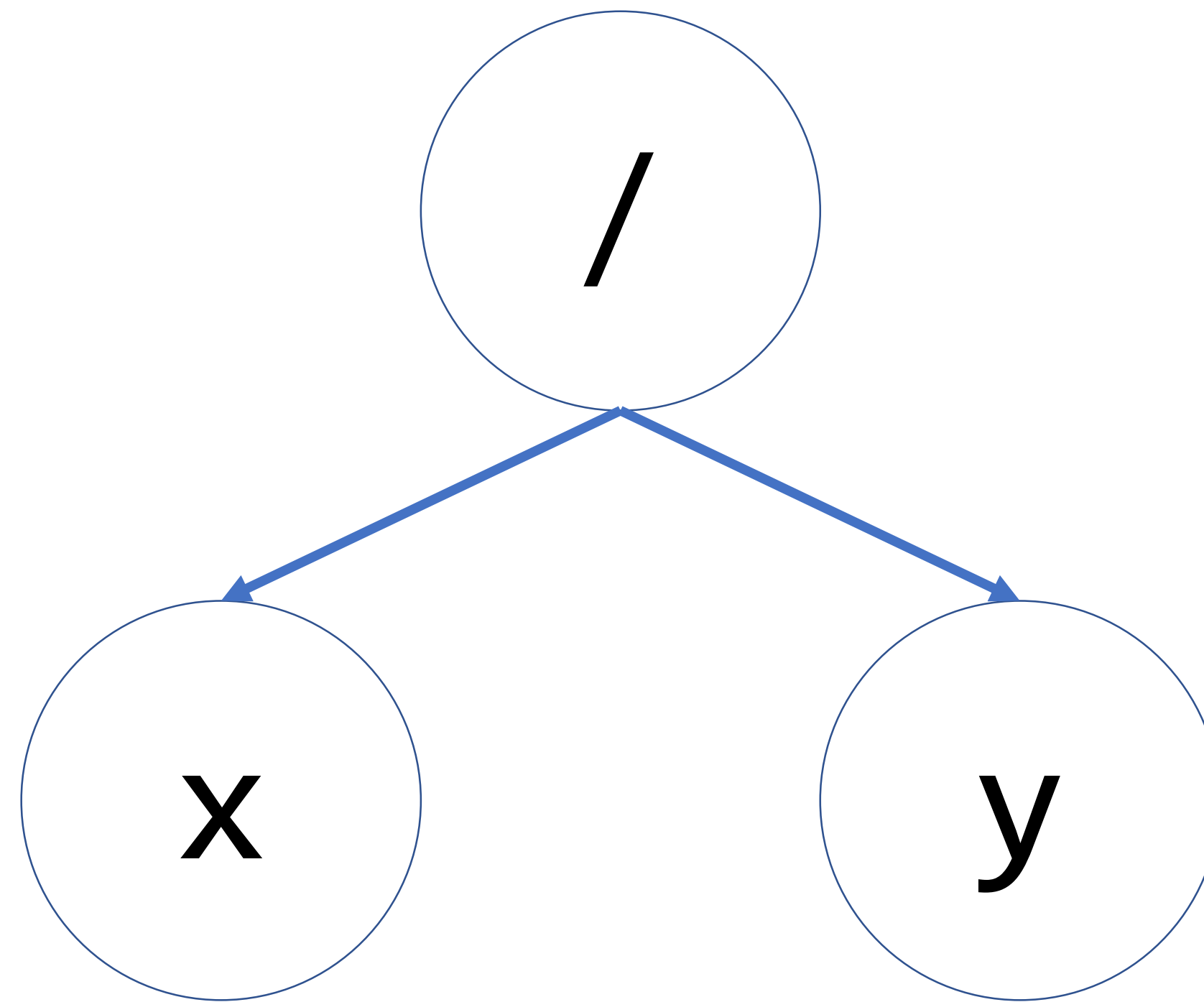
Как найти деление на ноль?



Type: `s_int_64`
Range: `[100, 256)`

Type: `s_int_64`
Range: `[-30, 12)`

Как найти деление на ноль?



Type: `s_int_64`
Range: `[100, 256)`

Type: `s_int_64`
Range: `[-30, 12)`

Possible division by zero!

```
Geom::PathVector LPEFilletChamfer::doEffect_path(...)\n{\n    // ....\n    if(/**...*/){\n        // ....\n    } else if (type >= 3000 && type < 4000) {\n        unsigned int chamferSubs = type-3000;\n        ....\n        double chamfer_stepTime = 1.0 / chamferSubs;\n        // ....\n    }\n    // ....\n}
```



Предупреждение PVS-Studio: V609 Divide by zero. Denominator range [0..999].

```
Geom::PathVector LPEFilletChamfer::doEffect_path(...)\n{\n    // ....\n    if(/**...*/){\n        // ....\n    } else if (type >= 3000 && type < 4000) {\n        unsigned int chamferSubs = type-3000;\n        ....\n        double chamfer_stepTime = 1.0 / chamferSubs;\n        // ....\n    }\n    // ....\n}
```

type = [3000, 3999]



Предупреждение PVS-Studio: V609 Divide by zero. Denominator range [0..999].

```
Geom::PathVector LPEFilletChamfer::doEffect_path(...)\n{\n    // ....\n    if(/**...*/){\n        // ....\n    } else if (type >= 3000 && type < 4000) {\n        unsigned int chamferSubs = type-3000;\n        ....\n        double chamfer_stepTime = 1.0 / chamferSubs;\n        // ....\n    }\n    // ....\n}
```

chamferSubs = [0, 999]



Предупреждение PVS-Studio: V609 Divide by zero. Denominator range [0..999].

```
Geom::PathVector LPEFilletChamfer::doEffect_path(...)\n{\n    // ....\n    if(/*.....*/){\n        // ....\n    } else if (type >= 3000 && type < 4000) {\n        unsigned int chamferSubs = type-3000;\n        ....\n        double chamfer_stepTime = 1.0 / chamferSubs;\n        // ....\n    }\n    // ....\n}
```

chamferSubs = [0, 999]



Предупреждение PVS-Studio: V609 Divide by zero. Denominator range [0..999].

```
struct RawRGBABitmap
{
    sal_Int32          mnWidth;
    sal_Int32          mnHeight;
    std::shared_ptr< sal_uInt8 > mpBitmapData;
};
```

```
RawRGBABitmap bitmapFromVCLBitmapEx( const ::BitmapEx& rBmpEx )
{
    // ....
    aBmpData.mnWidth      = aBmpSize.Width();
    aBmpData.mnHeight     = aBmpSize.Height();
    aBmpData.mpBitmapData.reset( new sal_uInt8[ 4*aBmpData.mnWidth
                                                *aBmpData.mnHeight ] );

    // ....
}
```

Предупреждение PVS-Studio: V554 Incorrect use of shared_ptr.

The memory allocated with 'new []' will be cleaned using 'delete'.

```

struct RawRGBABitmap
{
    sal_Int32                mnWidth;
    sal_Int32                mnHeight;
    std::shared_ptr< sal_uInt8 > mpBitmapData;
};

RawRGBABitmap bitmapFromVCLBitmapEx( const ::BitmapEx& rBmpEx )
{
    // ....
    aBmpData.mnWidth      = aBmpSize.Width();
    aBmpData.mnHeight     = aBmpSize.Height();
    aBmpData.mpBitmapData.reset( new sal_uInt8[ 4*aBmpData.mnWidth
                                                *aBmpData.mnHeight ] );

    // ....
}

```

Предупреждение PVS-Studio: V554 Incorrect use of shared_ptr.

The memory allocated with 'new []' will be cleaned using 'delete'.

На каких механизмах работает анализ?

Работает на множестве методов анализа

Data-flow
анализ

Символьное
выполнение

Выведение
типов

Аннотирование
методов

Межмодульный
анализ

Анализ помеченных
данных

Чтобы находить множество возможных проблем в коде

Путаница с приоритетом операций

Переполнение буфера

Copy-paste

Недостижимый код

Ошибки синхронизации

Ошибки при работе с исключениями

Неправильная работа с методами

Выход за границу массива

Опечатки

Микрооптимизации

Неправильная работа с типами

Ошибки сериализации / десериализации

Статический анализатор кода PVS-Studio

Решение для улучшения качества,
защищённости (SAST) и безопасности кода

Попробовать бесплатно

Запросить ВКС

ГОСТ Р 71207-2024

ФСТЭК №117

MISRA C 2023

РБПО

ГОСТ Р 56939-2024

АСУ ТП

Q/A