Федеральное государственное автономное образовательное
учреждение высшего образования
«Новосибирский национальный исследовательский государственный
университет»
09.03.01 Информатика и вычислительная техника. Программная
инженерия и компьютерные науки

Статический анализ фрагментированных программ на базе абстрактного синтаксического дерева

Выполнил: Курбатов Максим Андреевич

Научный руководитель: Власенко А.Ю., к.т.н., доц. каф. ПВ ФИТ НГУ

Соруководитель: Киреев С.Е., ст. преп. каф. ПВ ФИТ НГУ

Дата защиты: 24.06.2024

Новосибирск 2024

Введение

Проблемы написания параллельных программ:

1. Сложность отладки:
    a. недетерминированность выполнения параллельной программы;
    b. гонки данных;
    c. взаимные блокировки.
2. Организация коммуникаций между узлами.

Введение

LuNA (Language for Numerical Algorithms) – система, разрабатываемая в лаборатории синтеза параллельных программ ИВМиМГ СО РАН. Ее основная цель – это упростить рабочий процесс программиста за счет сокращения трудозатрат на создание параллельных реализаций алгоритмов.

Актуальность

На текущий момент средств отладки на этапе написания программы для LuNA нет, а компилятор во многих случаях не информирует пользователя об ошибке в доступной форме.

Цель и задачи

Цель на ВКР: Проектирование и разработка статического анализатора LuNA-программ на базе абстрактного синтаксического дерева.

Задачи на ВКР:

1. Выявление и анализ ошибок, свойственных фрагментированным программам. Участие в формировании базы типов ошибок LuNA-программ.
2. Обзор методов и инструментальных средств статического анализа.
3. Реализация построения AST (abstract syntax tree) для произвольной LuNA-программы.
4. Разработка средства AST-analyzer, реализующего обнаружение ошибок из перечня.
5. Реализация параллельного поиска ошибок в средстве AST-analyzer.
6. Оценка времени работы и потребляемых ресурсов статического анализатора на различных LuNA-программах.
7. Интеграция программного средства AST-analyzer в систему автоматизированной отладки ADAPT.

Подходы к отладке параллельных программ

1. Диалоговая отладка (TotalView, Allinea DDT).
2. Анализ по трассе (Intel Inspector, Jumshot).
3. **Статический анализ** (Cppcheck, PVS-Studio).
4. Верификация программ на моделях (Spin, NuSMV).
5. Тестирование.

Статический анализ

Достоинства:

1. Ускоряет процесс разработки путем обнаружения ошибок и уязвимостей без запуска программы.
2. Обеспечивает полное покрытие кода, а не анализ определенного варианта его исполнения.
3. Не требует от пользователя построения дополнительных конструкций (моделей, эталонных версий и т.д).

Недостатки:

1. Ложные срабатывания: инструменты статического анализа могут определять корректные участки кода как ошибочные.
2. Ограниченность анализа: не всегда можно найти ошибку до запуска программы.

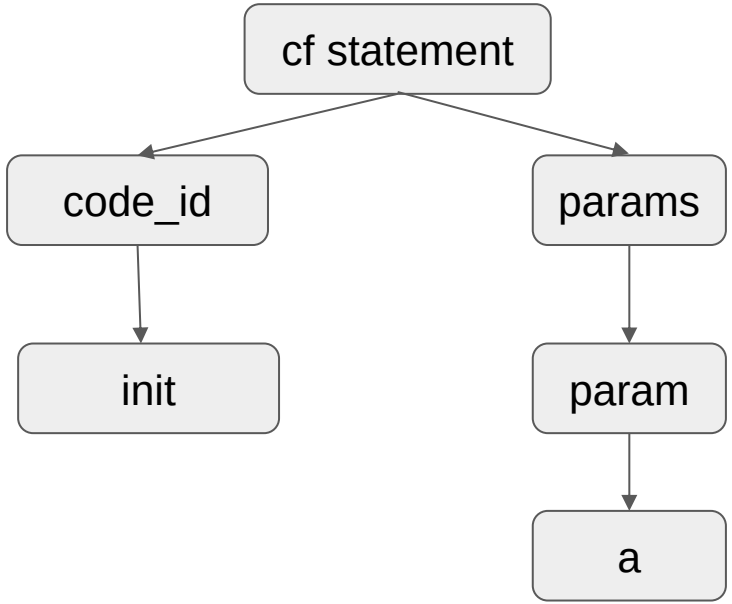
Методы статического анализа

1. **Сопоставление с шаблоном:**
   a. шаблон – параметризованный кусочек кода, который содержит ошибку.
2. **Анализ потока данных:**
   a. изучение данных программы в конкретных точках;
   b. обычно используется граф потока управления.
3. **Вывод типов.** Например: 1 + ‘2’ => int + string => string.
4. **Измерение программных метрик:**
   a. количество строк кода;
   b. количество ФК.

Абстрактное синтаксическое дерево

1. AST – одно из базовых промежуточных представлений программы.
2. Структура AST: внутренние вершины – операторы, листья – операнды.

init(a);



Требования к AST-analyzer

1. Наличие функций для автоматического построения AST для **произвольной** LuNA-программы.
2. Наличие функций, использующих AST для поиска ошибок в программе.
3. Возможность продолжать анализ после обнаружения очередной ошибки.
4. Наличие функций, позволяющих генерировать отчёты об ошибках в унифицированном формате.
5. Наличие консольного интерфейса.

Алгоритм анализа AST в AST-analyzer

1. Построение AST:
    a. лексический анализ – Flex;
    b. синтаксический анализ – Bison.
2. Обход AST в глубину и сбор нужных узлов из AST в контейнер.
3. Семантический анализ этих узлов:
    a. проверка узлов на уникальность;
    b. проверка существования нужных узлов.

Выполняемые проверки на ошибки

За ВКР была собрана база из 37 типов ошибок для LuNA программ.
Используется 6 видов обхода AST для поиска 13 ошибок.

Примеры ошибок LuNA, присущие другим языкам:

1. Два или более объявлений ФД в подпрограмме (LUNA7).
2. Несоответствие типов аргументов при вызове атомарного ФК (LUNA36).
3. Отсутствие ФК main (LuNA12).

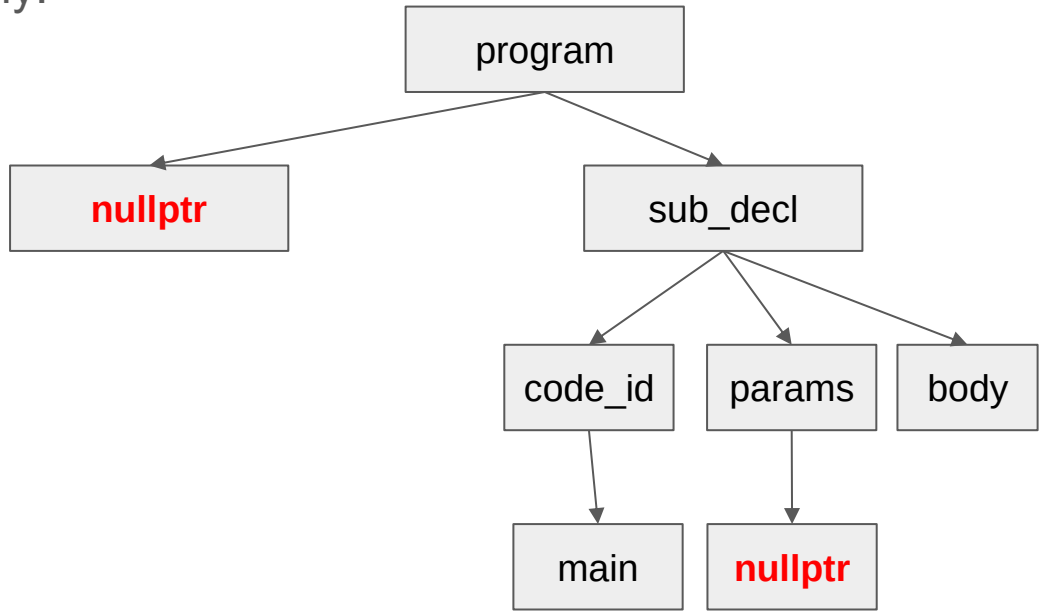
Примеры уникальных ошибок LuNA:

1. Импорт нескольких разных ФК по одним алиасом (LUNA11).
2. Циклическая зависимость по данным (LUNA15).
3. Попытка индексации объекта, не являющегося ФД (LUNA36).

AST в случае синтаксических ошибок

Пример программы и AST к нему:

```
1 // missing ";"
2 import c_foo() as foo
3
4 sub main() {
5 // missing ":"
6         cf f  foo();
7 }
```



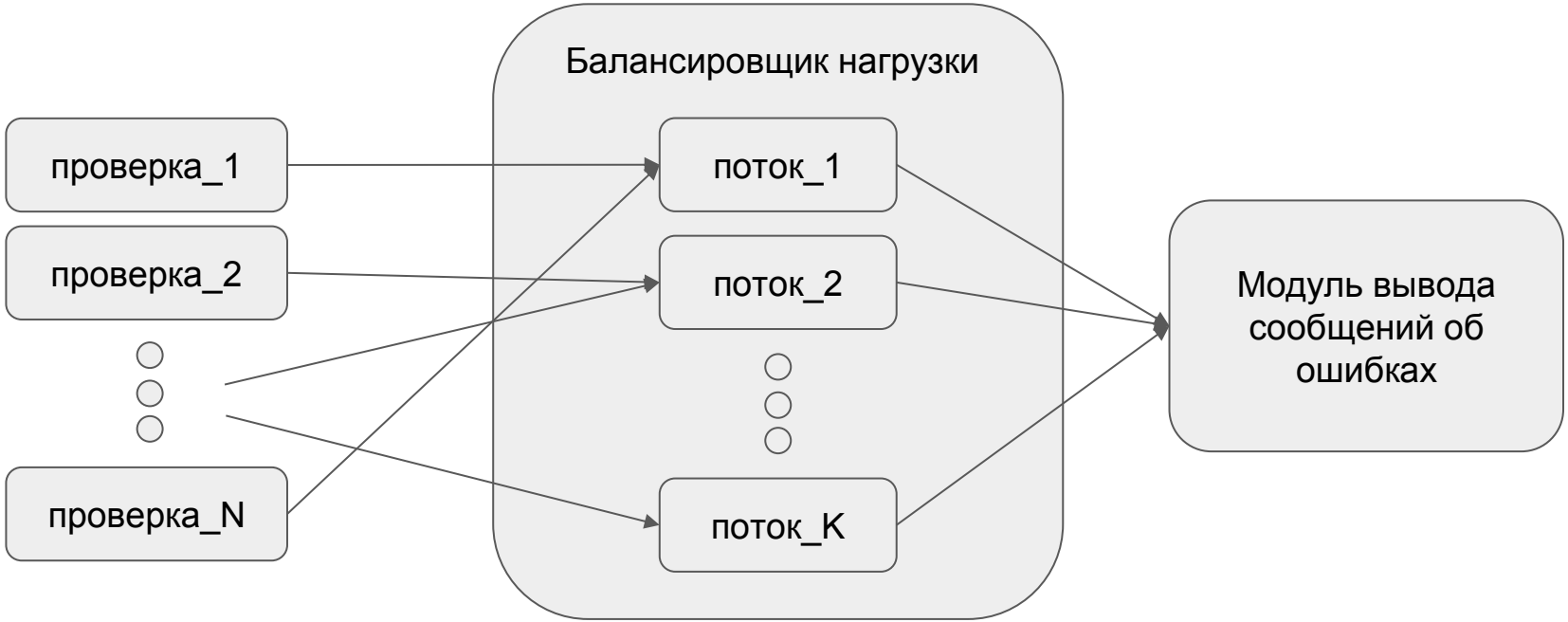
AST в случае синтаксических ошибок

Правило Bison для восстановления после грамматической ошибки:

statement:

cf_statement	cf init : init(a, b, c);
| let_statement	let x=y[k][l], message="Success" {}
| for_statement	for i=1..N {}
| while_statement	while (x[i] < 0) , i = 0 .. out N {}
| if_statement	if (a > b) {}
| error SCOLON	не подошло ни под одно правило выше

Параллельный анализ



ADAPT – комплекс автоматизированной отладки

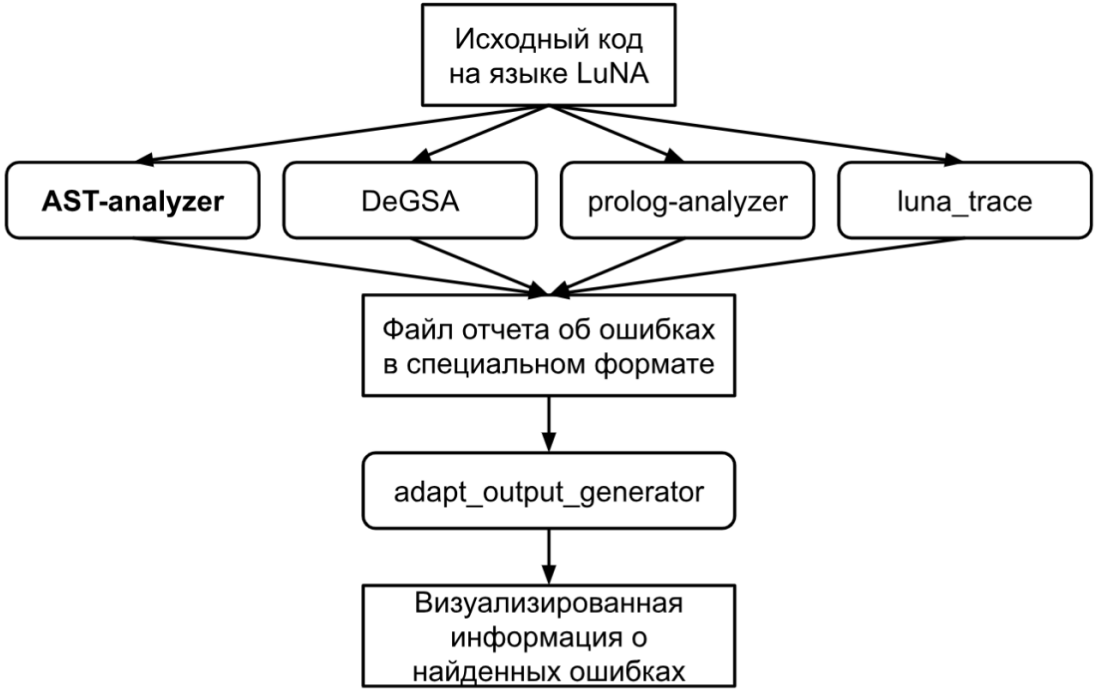


Схема работы ADAPT

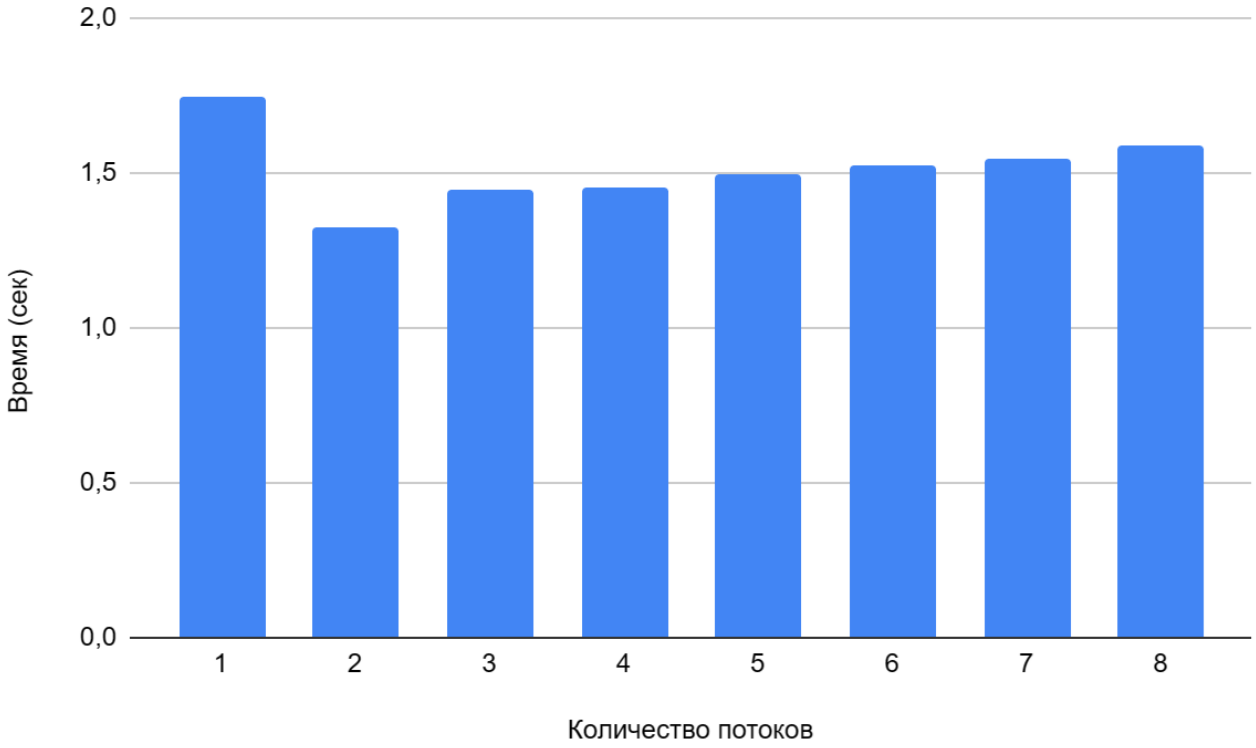
Тестирование

Задачи для тестирования:

1. Решение СЛАУ методом прогонки.
2. Расчет движения протопланетного диска методом PIC.

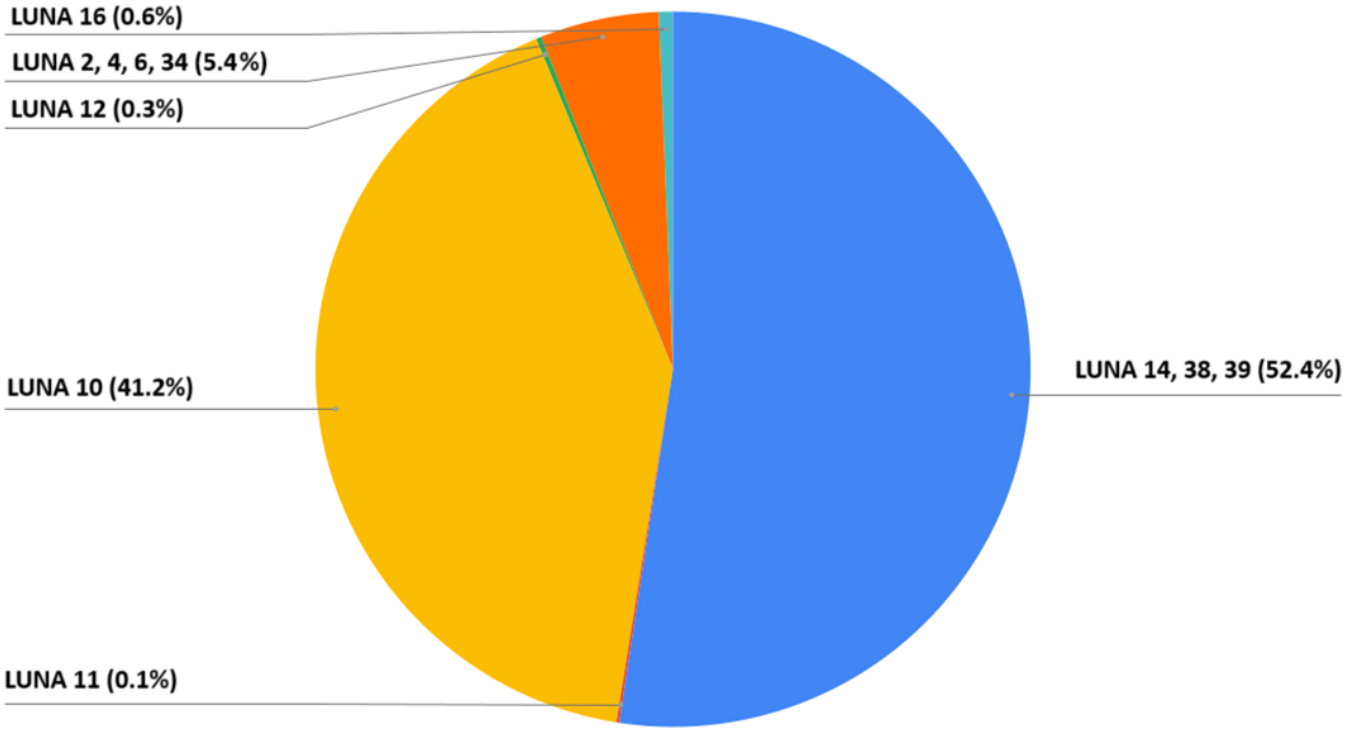
	Метод прогонки	Протопланетный диск
Исходный код (Мб)	0,0214	0,593
Исходный код (строки)	471	10401
Токенов в дереве разбора	11936	398969
Токенов в AST	3799	131236

Тестирование. Оценка времени анализа (Прогонка)



Тестирование. Доля времени обходов (Прогонка)

Тип ошибки	Время, с
LUNA16	0,07
LUNA14, 38, 39	41
LUNA11	0,1
LUNA10	32
LUNA12	0,3
LUNA34, 4, 2, 6	4,2



Научная новизна и практическая значимость

Научная новизна: для фрагментированных программ предложена методика построения абстрактного синтаксического дерева (AST), устойчивого к синтаксическим ошибкам, а также разработаны оптимизированные алгоритмы обнаружения ряда синтаксических и семантических ошибок на основе построенного AST.

Практическая значимость состоит в сокращении трудозатрат на написание корректного кода программ на языке LuNA.

Статьи и тезисы

1. Курбатов. М.А., **Статический анализ исходного кода программ на примере языка LuNA**, Сборник материалов IV Международной научно-практическ ой конференции «Вызовы глобализации и развитие цифрового общества в условиях новой реальности» [Электронный ресурс]. — 2022. — 243 с. — С. 153-158. — Режим доступа к журн.: https://www.elibrary. ru/item.asp?id=5013 6476&selid=501383 73, свободный. – Загл. с экрана.
2. Курбатов. М.А., **Поиск ошибок во фрагментированных программах с помощью абстрактного синтаксического дерева**, 62-я Междунар. студ. конф. (г. Новосибирск, 17 - 26 апреля 2024 г.): Тез. докл. / Новосиб. гос. унт. – Новосибирск : ИПЦ НГУ (принято в печать).
3. Курбатов. М.А., **Статический анализ фрагментированных программ на базе абстрактного синтаксического дерева**, Конференция «Молодёжные Марчуковские научные чтения 2023» (г. Новосибирск, 18 - 20.12.2023 г.): Тез. докл. / ИВМиМГ СО РАН – Новосибирск : ИВМиМГ СО РАН (принято в печать).
4. Курбатов. М.А., Власенко А. Ю., Царев В. Д., Мичуров М. А., **Построение комплекса автоматизированной отладки фрагментированных программ**, Вестник НГУ. Серия: Информационные технологии. — 2024. — Т. 22. — № 1. (принято в печать).

Результаты

1. Принято участие в составлении базы ошибок LuNA.
2. Изучены средства и методы статического анализа.
3. Реализован поиск ошибок 13 типов.
4. Оценено время анализа и потребление ресурсов.
5. Реализован параллельный поиск ошибок.
6. Проведена интеграция AST-analyzer в ADAPT.
7. Принято участие в научных конференциях, написаны 2 тезиса и 2 статьи.

Спасибо за внимание!

Планы на будущее

1. Обеспечить параллельную работу на уровне веток AST.
2. Поиск новых типов ошибок LuNA-программ.
3. Реализовать новые типы ошибок.

Структура ВКР

1. 4 главы
    a. Ошибки в LuNA программах
    b. Статический анализ
    c. Анализатор AST языка LuNA
    d. Тестирование
2. Объем работы: 55 страниц
3. Количество иллюстраций: 12
4. Количество таблиц: 1
5. Количество литературных источников: 25
6. Количество приложений: 3

Ошибки, которые обнаруживает AST-analyzer.

1. Несоответствие типов аргументов при вызове атомарного ФК.
2. Несуществующая LuNA-подпрограмма.
3. Несоответствие типов аргументов при вызове структурированного ФК (неверный порядок аргументов).
4. Несовпадение количества аргументов при объявлении ФК и его вызове.
5. Два или более объявлений ФД в подпрограмме.
6. Несоответствие типов LuNA при присваивании.
7. Неиспользуемое имя.
8. Импорт нескольких разных функций по одним алиасом.
9. Отсутствие ФК main.
10. ФД с одинаковыми названиями в одной области видимости.
11. Безусловная рекурсия.
12. Неверные границы циклов (for i=10..1).
13. Использование нецелых переменных в границах циклов.