Динамический анализатор кода

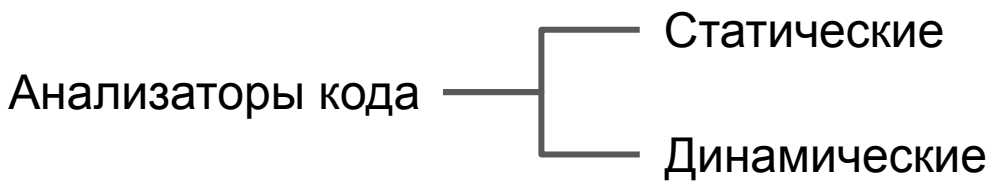
Команда:
Бриллиантов Кирилл Юрьевич
Лочмелис Денис Юрьевич
Соловьев Глеб Константинович
Ментор: Безбородов Павел Андреевич

ВШЭ СПб ПМИ,
весна 2020

Введение

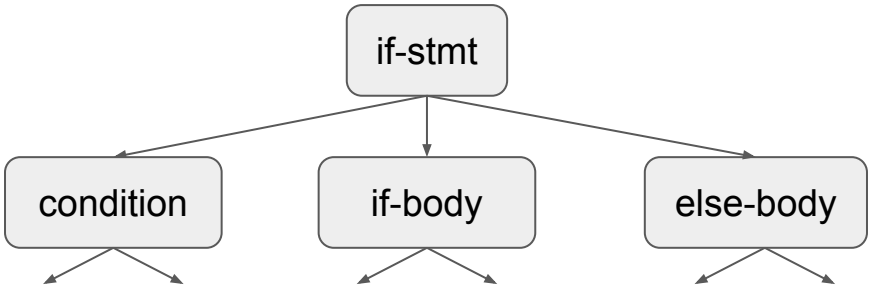
Ошибки в программах сложно обнаруживать

Решение - **анализаторы кода**, инструменты для нахождения ошибок и ошибкоопасных мест в программе



Введение. Основные понятия

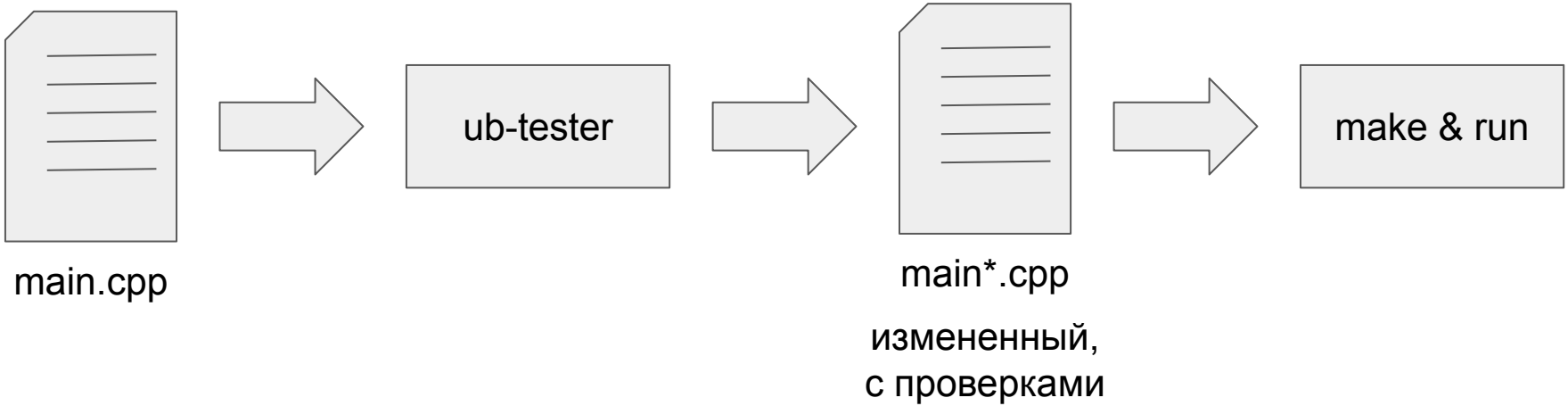
- Неопределенное поведение (UB) - нарушение правил языка, приводящее к неопределенному результату выполнения программы
- Абстрактное синтаксическое дерево (AST) - дерево, внутренние вершины которого - операторы языка, а листья - операнды
  Является удобной абстракцией для работы с исходным кодом



Описание проекта

Цель: создать анализатор кода, выявляющий некоторые UB в коде

Схема работы приложения:



Сравнение с аналогами

Название	**Описание**	**Статический/динамический**
clang static analyzer	большой функционал	статический
PVS studio	≈ clang, но платная	статический
valgrind	работает с памятью	динамический
ub-tester tool	работает с UB	динамический

Главное отличие нашего проекта — мы **изменяем** код!

Используемые технологии

Мы пользовались библиотеками clang и llvm

Предоставляют удобный интерфейс для

- взаимодействия с AST
- создания инструментов, работающих с исходным кодом

Альтернативы:

- github.com/foonathan/cppast - маленькое сообщество
- разбирать исходный код и строить дерево самим

Архитектура. Шаблоны проектирования

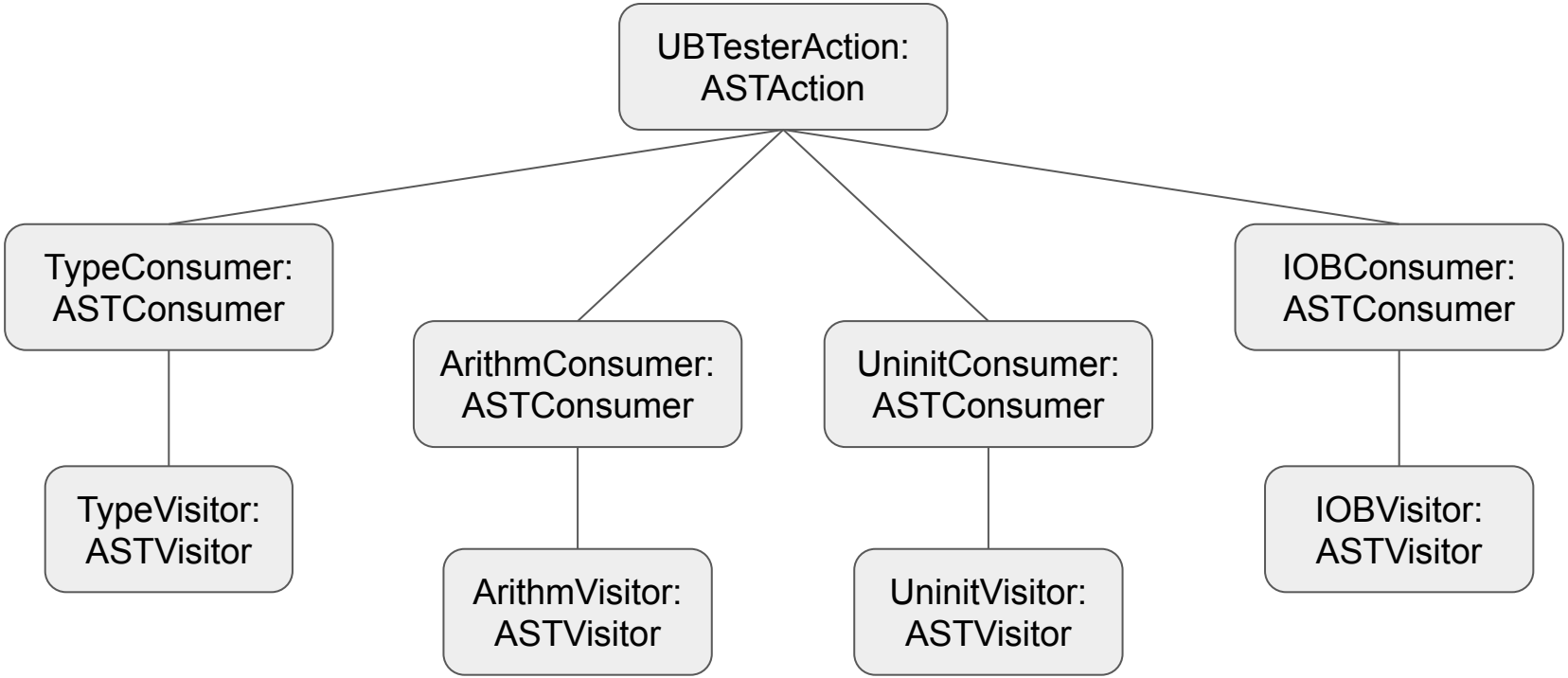
- Посетитель (Visitor) - шаблон, позволяющий оперировать объектами других классов, не изменяя их состояния

  Активно применяется в clang для взаимодействия с узлами AST

- Производитель-Потребитель (Producer-Consumer) - шаблон, описывающий обмен данными между несколькими потоками

  Используется в clang, в том числе для создания инструментов

Архитектура



Задачи Бриллиантова Кирилла

- Реализация интерфейса для изменения пользовательского кода. Класс CodeInjector, который лениво применяет переданные ему “замены”
- Реализация объекта TypeSubstituterVisitor, который заменяет все интересующие нас типы на безопасные, реализованные нами, обертки
- Обработка UB, возникающих при работе с C-style массивами и указателями

Задачи Лочмелиса Дениса

- Проверка доступа к значению неинициализированных переменных
  - по ссылке
  - с использованием классов
  - поддержка доступных функций

- Создание консольного пользовательского интерфейса

Задачи Соловьева Глеба

- Обнаружение UB в бинарных и унарных операторах и в составных операторах присваивания (+=, -= и т. д.) для целочисленных типов
- Обнаружение потери значения во время неявных приведений между целочисленными типами
- Реализация логики вывода сообщений пользователю

Планы на будущее

- Поддержка арифметики типов с плавающей точкой
- Поддержка проверок в шаблонах
- Реализация графического интерфейса (или плагина для редактора), с помощью которого пользователь сможет указать места в своем коде, которые он хочет проверить

ДЕМО

github.com/KirillBrilliantov/ub-tester-tool