Компилятор Flow и его оптимизация

Доклад посвящен языку программирования Flow, разработанному компанией Area9 Innovation, компилятору этого языка, а также оптимизации компилятора в процессе разработки.
Язык Flow разрабатывается компанией Area9 Innovation с 2010 года и представляет собой кросс-платформенный язык программирования, предназначенный для создания сложных графических интерфейсов. Продукты, созданные с помощью этого языка, использует более 12 миллионов человек по всему миру. В магазинах App Store и Google Play представлено несколько мобильных приложений, также разработанных с помощью Flow.
В докладе будут даны описание языка, описание внутренней структуры компилятора, а также описание исполнительной машины и библиотеки времени выполнения языка.
Рассматриваются следующие вопросы:

- общая структура компилятора;
- структура лексического и синтаксического анализатора;
- исполнительная машина Flow;
- генераторы кода под разные платформы.

Также в докладе будут представлены методы, применявшиеся при оптимизации компилятора и исполнительной машины, описаны ряд использованных высокоуровневых и низкоуровневых способов оптимизации.

Автор доклада – Дмитрий Игнатьевич Соломенников, инженер-программист. Сотрудничает с ООО «Ланит» с 2013 года. Участвует в разработке компилятора языка Flow с 2016 года.

Тел.: +7(906)952-25-22
Skype: d.solomennikov99
dmitrys99@mail.ru

Язык Flow

Краткое описание языка

- Мотивация к разработке языка
- Текущее положение дел
- Open-source Flow

Компилятор Flow

Структура компилятора

- Анализатор Lingo
- Desugaring
- Типизация
- Оптимизация
- Генерация

Анализатор

- Использование PEG-парсера с генерацией конечного автомата разбора в виде исходного кода на Flow.

Исполнительная машина Flow

- QtByteRunner: desktop
- QtByteRunner: mobile
- JS runtime
- Native-методы
- WebAssembly

Генераторы кода

- Bytecode
- Javascript
- Экспериментальные генераторы
- Nim, D, Lisp, C++, ML
- Java

Высокоуровневая оптимизация

- Упрощение структур данных после вывода типов (FcModule -> FiModule)
- Удаление мертвого кода
- Удаление мертвых структур (в работе)
- Дублируемые строки
- Инкрементальная сборка
- Linter

Низкоуровневые оптимизации

- Использование правильных структур данных
- Расход памяти:
    - Двухбайтные строки
    - Эмодзи
    - Сохранение промежуточных файлов
- JIT компиляция
- Массивы против списков
- Оптимизация native-методов