Санкт-Петербургский государственный университет

Математико-механический факультет

Кафедра системного программирования

Разработка Eclipse плагина для программирования на языке haXe

Курсовая работа студентки 361 группы

Савенко Марии Олеговны

Научный руководитель

Старший преподаватель …………… /В.С.Полозов/

 /подпись /

Санкт-Петербург

2011

## Содержание

[1. Введение 3](#_Toc294138733)

[2. Постановка задачи 4](#_Toc294138734)

[3. Структура программы 5](#_Toc294138735)

[3.1 Списки 5](#_Toc294138736)

[3.2 Индексирование 5](#_Toc294138737)

[4. Вывод типов 6](#_Toc294138738)

[4.1 Стандартные типы 6](#_Toc294138739)

[4.2 Образование подтипов 6](#_Toc294138740)

[4.3 Динамические типы 6](#_Toc294138741)

[4.4 Порядок поиска определения типа 6](#_Toc294138742)

[5. Заключение 7](#_Toc294138743)

[6. Список литературы 8](#_Toc294138744)

## 1. Введение

haXe[3] – объектно-ориентированный строго типизированный язык программирования высокого уровня, ориентированный на разработку web-приложений. Разработка его началась в конце 2005 года французским программистом Николя Канассом (Niсolas Cannasse). Выход версии 1.0 состоялся в марте 2006 года. На данный момент язык продолжает развиваться, последняя версия - 2.07.

haXe является мультиплатформенным языком и поддерживает платформы:

1. Javascript **–** обычно, по нескольким классам, написанным на языке haXe, генерируется один «js» файл, который будет использоваться в web-приложении.
2. Flash **–** haXe может быть скомпилирован в swf-файл 6, 7, 8 и 9-ой версии.
3. Neko – может быть скомпилирован в байткод Neko Virtual Machine[4]
4. PHP – можно скомпилировать программу на haXe в .php-файл.
5. C++ –  можно сгенерировать код на C++ из кода на haXe, включая Makefile.
6. Обещают ввести и поддержку C# и Java.

Синтаксис haXe покажется знакомым тем, кто знаком с ActionScript и JavaScript. Cтрогая типизация позволяет находить ошибки до тестирования в браузере и, таким образом, заметно ускоряет разработку для динамически языков.

Немаловажным фактором является и удобная среда разработки. Из бесплатных проектов в этой области стоит упомянуть: FashDevelop[12], EclihX[9] и различные текстовые редакторы. Есть так же и платная среда FDT[16], с испытательным периодом в 30 дней. Однако бесплатные проекты не позволяют отслеживать ошибки до компиляции.

Данная работа продолжает труд Кондратьев А.Е.[10] по созданию интегрированной среды разработки для языка haXe на базе платформы Eclipse, главным достоинством которой, была бы возможность уведомлять пользователя о возможных ошибках прямо в момент написания кода и тем самым повысить удобство среды и скорость разработки программных продуктов.

## 2. Постановка задачи

Как уже было упомянуто, работа была начата не с «нуля». Однако все базовые алгоритмы были реализованы на AST-дереве.

Основной целью работы была реализация вывода типов. Для успешного выполнения данной задачи необходимо было ввести новые структуру – таблицы переменных. Таблицы переменных должны были заменить AST-дерево, как источника информации о программе. Поэтому дальнейшей задачей являлось переложение существующих алгоритмов на новую систему.

Задачи:

1. Создание структуры – таблица переменных
	1. Сама таблица
	2. Классы для записей таблицы – переменная, функция, перечисление и т.д.
2. Адаптация реализованных интерфейсов IMP:
	1. Визуальное представление кода (Outline)
	2. Всплывающие подсказки по элементам кода
	3. Переход от переменной к ее определению по ctrl-щелчок
	4. Вывод сообщений об ошибках
3. Алгоритмы вывода типов
	1. Переопределение переменной с незаданным типом
	2. Проверка корректности операций присвоения, возвращения значений, а так же арифметических действий над объектами с конкретным типом
4. Диагностика ошибок

## 3. Структура программы

# 3.1 Таблицы

Изначально основным элементом всех алгоритмов в программе было абстрактное синтаксическое дерево. Однако работать с деревом нерационально, особенно когда приходиться обращаться к нему несколько раз в рамках одной задачи. Деревья так же обычно содержат много лишней информации, необходимой, только на первых этапах работы. Поэтому для повышения скорости работы с уже обработанными файлами, в частности и с библиотечными, было решено сохранять информацию в более удобной форме, а дерево хранить только для текущего обрабатываемого файла и использовать для таких целей как, например, подсветка синтаксиса.

Таким образом, были введены таблицы переменных, каждая запись которых хранит информацию об имени, области видимости, месте в коде исходной программы и типе внесенного объекта. Если объект является Классом, то он может содержать собственную таблицу внутри. Таким образом, создается простая иерархия вложенных таблиц, верхняя из которых принадлежит текущему модулю.

Для построения Outline достаточно пройтись по таблицы верхнего уровня, заглядывая во вложенные в зависимости от настроек или по требованию пользователя.

Для вычисления типов таблицы облегчают задачу поиска определения переменной и всех ее вхождений.

# 3.2 Индексирование

В таких структурах, как таблица, очень быстро возникает в необходимости специального именования переменных. Главная задача переименования – выделить связанные переменные. Но так как настоящие имя переменной нам все равно бы потребовалось, решено было лишь проиндексировать их. Свободным переменным присваивается нулевой индекс. Для определений генерируется уникальный индекс (в зависимости от того, определялась ли она раньше), а всем вхождениям переменной присваивается индекс ее определения.

## 4. Вывод типов

# 4.1 Стандартные типы

haXe – объектно-ориентированный язык программирования . Однако он не совсем обычный объектно-ориентированный язык. В каждый тип в haXe вкладывается больше смысла чем в единственный тип Объект. Типы делятся на группы:

* Классы
* Структуры
* Перечисления (Enums)
* Функции
* Определения типов
* Динамические
* Неопределенные (Unknown)
* Абстрактные

Тип переменной указывается при определении, но для локальных переменных существует возможность не указывать его явно. По умолчанию такие переменные получают тип Unknown, что делает их все равно строго типизированными. При первом обращении к ним для чтения или записи, их тип будет автоматически переопределен.

# 4.2 Образование подтипов

Передаваемому значению, конечно же, вовсе не обязательно быть в точности того же типа, какой требовался. Но в этом случае способность переопределения типа локальных переменных может сыграть дурную шутку. Например, тип Float является подтипом типа Integer. Если неизвестной переменной будет сначала значение типа Int, а затем будет попытка присвоить значение типа Float, то должно быть выведено сообщение об ошибке.

После построения иерархии стандартных типов, она может быть позже дополнена типами, определенными пользователем, для чего достаточно скопировать нужные свойства из уже существующих элементов.

# 4.3 Динамические типы

Динамический тип можно подставлять вместо любого типа без проверки во время компиляции. Однако во время выполнения, разные языки-цели могут не распознать или даже не разрешить некоторые операции с конкретной переменной, поэтому их использование не поощряется. К примеру, на Flash 9, когда динамической переменной присваивается конкретное значение (Integer или скажем Boolean), бросается исключение. На менее строгих платформах, как Javascript, ошибки может и не возникнуть, но значение не сохранится.

# 4.4 Порядок поиска определения типа

При обнаружении имени типа, его определение ищется в следующем порядке:

1. параметризующие типы текущего класса
2. стандартные типы
3. типы, объявленные в текущем файле
4. типы, объявленные в импортируемых файлах (если в данном пакете объявление не найдено)
5. если определение все еще не обнаружено, загружается соответствующий файл и определение типа ищется внутри него

## 5. Заключение

Данная работа продолжает труд Кондратьев А.Е. по созданию интегрированной среды разработки для языка haXe на базе платформы Eclipse. В версии, доступной к моменту окончанию курсовой работы, следующая функциональность была добавлена или адаптирована для работы со списками объектов:

1. Визуальное представление кода – построение идет по таблице переменных
2. Адаптация ранее реализованных интерфейсов IMP:
	1. Outline
	2. Всплывающие подсказки по элементам кода
	3. Переход от переменной к ее определению по ctrl-щелчок
	4. Вывод сообщений об ошибках
3. Частичная диагностика ошибок:
	1. Повторное определение переменной
	2. Использование неопределенной переменной или функции
	3. Переопределение переменных с неизвестным типом при использовании
	4. Ошибки приведения примитивных типов
	5. Проверка наличия и соответствия нужному типу возвращаемых значений функции

Основным результатом работы было положено начало для канонического вывода типов.

## 6. Список литературы

1. **Бенджамин Пирс «Типы в языках программирования»**
2. **Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман. «Компиляторы: принципы, технологии и инструментарий»**
3. Официальный сайт языка haXe, <http://haxe.org>
4. Neko Virtual Machine, <http://nekovm.org/>
5. Eclipse Java Development Tools, <http://www.eclipse.org/jdt/>
6. Официальный сайт on-line игры MyMiniCity; создан при использовании haXe, <http://myminicity.com/>
7. Официальный сайт проекта comapping, <http://www.comapping.com/>
8. Официальный сайт on-line игры Alpha Bounce; создан при использовании haXe, <http://www.alphabounce.com/>
9. Красько Н.Л. Разработка отладчика для программ на языке haXe и целевой платформы Adobe Flash 9. Дипломная работа, СПбГУ, 2008. <http://se.math.spbu.ru/SE/diploma/2008/index_html>
10. Кондратьев А.Е. Eclipse-плагин для программирования на haXe. Дипломная работа, СПбГУ, 2010. <http://se.math.spbu.ru/SE/diploma/2010>
11. Официальный сайт проекта ANTLR и среды ANTLRWorks, <http://www.antlr.org/>
12. Официальный сайт IDE FlashDevelop, http://www.ﬂashdevelop.org/
13. Официальная страница плагина для ANTLR под Eclipse, <http://antlreclipse.sourceforge.net/>
14. Официальная страница проекта IMP, <http://www.eclipse.org/imp/>
15. Официальная страница проекта eclihx, <http://code.google.com/p/eclihx/>
16. FDT haXe плагин – альфа версия, <http://fdt.powerflasher.com/blog/2010/07/fdt-haxe-plugin-public-alpha-version-released/>