Дегтярев Александр Александрович

952004-7

Контрольная работа №2

Объектно-ориентированное программирование

Тема: Перспективные технологии ООП

**Содержание:**

* Введение
* Событийно-ориентированное программирование
* Функциональное программирование
* Языки с динамической типизацией
* Стандартные библиотеки
* Технологии Java и .net
* Заключение

**Введение**

Парадигма ООП впервые формально была представлена в языке Simula67, разработанном в 60ых. Впоследствии этот язык оказал сильное влияние на такие языки как C++, Objective Pascal, Smalltalk.

С тех пор прошло много времени, а эта концепция сильно укрепилась и является на сегодняшний день, наверное, самым широкораспространенным методом создания программных средств. Популярные объектно ориентированные языки программирования постоянно развиваются, приобретают новые возможности: большинство из них являются мультипарадигменными, их развитие чаще всего заключается в адаптации механизмов из других парадигм.

**Событийно-ориентированное программирование**

Так с ростом популярности приложений с графическими пользовательскими интерфейсами в 90ых, возрасла необходимость в поддержке событийно-ориентированной методологии программирования.

Языки приобрели новые инструменты, дающие возможность создавать события, и реагировать на них: делегаты, события. Так же появились совершенно новые языки, которые имели сильную зависимость с графическими библиотеками, которые поставлялись вместе с ними. Например Cocoa Framework и язык Objective C, Delphi и VCL. В это время С получил поддержку динамического оповещения(dynamic messaging), основанного на аналогичной системе языка Smalltalk в виде расширения. Многие считают, что такая тесная ассоциация с графическими интерфесами обеспечила популярность ООП.

**Функциональное программирование**

Будучи чисто академической, функциональная парадигма, сейчас получила достаточно широкое распространение и в коммерческих продуктах. Как следствие появились такие языки как Scala, работающие в JVM и F# в .net которые часто используются для разработки высоконагруженных серверных решений. Эти языки являются мульти-парадигменными и помимо ООП также поддерживают функциональную парадигму программирования. Существовавшие ранее языки, например python, c# (начиная с версии 3.0) и Java 8 приобрели несколько инструментов, относящихся к функциональным языкам: лямдба и анонимные методы, map/reduce, ленивые вычисления. Что свидетельствует о смещении в сторону объектно-функциональных языков.

**Языки с динамической типизацией**

Также в последнее время, в связи с развитием Веб-технологий становится все больше объектно ориентированных языков с динамической типизацией. таких как Ruby, Python, Groovy, Dart. Так в C# с версии 4 также появились dynamic типы. Это позволяет избежать статической проверки типа, и дает возможность конструировать объекты во время выполнения программы. Динамические языки используют механизм позднего связывания для поиска методы, которые будет вызван у конкретного типа. Для этого не используются таблицы виртуальных методов, поиск осуществляется по имени метода во время выполнения, это позволяет компилятору не ссылаться на библиотеки содержащие объект во время компиляции. Это делает процесс компиляции устойчивым к конфликтам версий.

**Стандартные библиотеки**

Так же не стоит забывать про развитие стандартных библиотек, например, с помощью нескольких библиотек с++ можно легко использовать приемы функционального программирования. Бьерн Страуструп, создатель языка С++, утверждает, что это самый перспективный путь для развития его языка. Сейчас практически в каждом активно развивающемся языке, стандартная библиотека содержит классы для работы со строками, коллекциями, потоками ввода-вывода, возможностями параллельного программирования: потоками, процессами, сопроцессами, сериализацией, сетевыми протоколами, базами данных и т.д.

**Технологии Java и .net**

Наиболее значимыми технологиями, разработанными в последнее время являются Java и .net. Программы на java компилируются в байткод, который запускается на любой системе, под которую написана виртуальная Java-машина. Программа может быть написана на нескольких языках, которые компилируются в байткод, например на Scala,JRuby, Jython и т.д.

.Net использует IL (Intermidiet Language) в качестве посредника между высокоуровневым языком в системе типов clr и машинными инструкциями. Все языки, базирующиеся на системе типов .net могут взаимодействовать друг с другом без каких либо сложностей. Класс написанный на IronRuby, затем, может быть использован в С# или vb.net. Существует аналог .net под названием Mono, который стремительными темпами развивается при поддержки Linux-сообщества. Программы с mono runtime запускаются на огромном количестве платформ, и предоставляют доступ к множеству библиотек, написанных для .net.

**Заключение**

В качестве заключения хотелось бы отметить, что большинство современных языков являются мульти-парадигменными. О предоставляют инструменты позволяющие решить задачу множеством разных способов, свободно смешивая различные стили и конструкции из разных парадигм. Смысл существования таких языков в том, чтобы дать возможность выбрать лучший способ решения конкретной проблемы, ведь ни одна парадигма отдельно не способна решить все проблемы сразу наиболее легким и эффективным путем.

**Литература**

<http://en.wikipedia.org/wiki/Multi-paradigm_programming_languages>

<http://en.wikipedia.org/wiki/Dynamic_dispatch>

<http://en.wikipedia.org/wiki/Late_binding>

 Floyd, R. W. (1979). ["The paradigms of programming"](http://www.ias.ac.in/resonance/May2005/pdf/May2005Classics.pdf)

http://msdn.microsoft.com/