СРАВНЕНИЕ ОСНОВНЫХ ХАРАКТЕРИСТИК ЯЗЫКОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь

И. Л. Сергеева

Ю. А. Луцик – к. т. н., доцент

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования (языки Java, C#, C++, JavaScript, ActionScript и др.). C++ и два его языка-потомка Java, C# были разработаны из различных соображений и пошли вследствие этого по разным путям. В этой связи представляет интерес сравнение данных языков

Исполнение программы. C++ изначально ориентировано на компиляцию в машинный код заданной платформы.

Важнейшим же принципом Java-платформы является трансляция исходного кода Java в байт-код, выполняемый на виртуальной машине JVM. Тот факт, что окончательная фаза компиляции выполняется машинно-зависимым устройством, поддерживаемым конечным пользователем, освобождает разработчика от необходимости создавать несколько исходных тестов программ для различных платформ. Процесс интерпретации также позволяет подключать данные на этапе выполнения, и это является основой динамической природы языка Java.

В C#, отвечающем спецификации CLR, при которой исходный текст программы также переводится не в машинный код, а в промежуточный язык, достигается высокая совместимость между различными языками, а также независимость от архитектуры компьютера и его операционной системы. Такой подход гарантирует безопасность исполнения программ, так как для каждой выполняемой программы создается своя виртуальная машина.

Но ценой межъязыкового взаимодействия и переносимости в Java и C# становится заметное снижение скорости работы программ и использование большего количества оперативной памяти.

Управление памятью. Java и C# работают в среде со сборкой мусора, которая автоматически отслеживает прекращение использования объектов и освобождает занимаемую ими память. C++ следует классической технике управления памятью, что предпочтительнее в системном программировании, где требуется полный контроль программиста над используемыми программой ресурсами. Первый же вариант удобнее в прикладном программировании, поскольку в значительной степени освобождает программиста от необходимости отслеживать момент прекращения использования ранее выделенной памяти.

Сборка мусора очень удобна, так как устраняет недостатки некорректная работа с памятью, но за ее использование приходится расплачиваться большим потреблением памяти и низкой производительностью. Еще одна причина, по которой сборка мусора является более дорогостоящей, чем непосредственное управление памятью программистом, – это утрата информации. В C++ программе программист знает и местонахождение своих блоков памяти (сохраняя указатели на них), и когда они перестанут быть ему нужными. В Java-программе последняя информация недоступна для JVM (даже если она известна программисту), поэтому JVM должна перебирать все блоки на предмет отсутствующих указателей.

Исключение ошибок. Очень часто можно проследить такую связь: чем более язык защищен и устойчив к ошибкам, тем меньше производительность программ, написанных на нем.

В программах, написанных на Java, за счет отсутствия перегрузки операций, объявления классов и методов как final, статической и динамической проверки преобразования типов (typecasting) гарантируется определенная безопасность кода.

В C# также существуют характерные особенности для обхода возможных ошибок. Например, помимо упомянутой выше «сборки мусора», все переменные автоматически инициализируются средой и обладают типовой защищенностью, что позволяет избежать неопределенных ситуаций в случае, если программист забудет инициализировать переменную в объекте или попытается произвести недопустимое преобразование типов. Были также приняты меры для исключения ошибок при обновлении программного обеспечения и изменения кода, реализуемого через поддержку совместимости версий.

Важными особенностями, которые сближают языки программирования C# и Java и обеспечивают определенную безопасность кода, являются механизмы интерфейсов, призванные заменить множественное наследование C++. Оно хотя и встречается с рядом проблем (коллизия имен, дублирующее наследование/от общего предка), его отсутствие существенно снижает выразительную мощь языка.

Используя Java, вы получаете защищенность, независимость от платформы, но, к сожалению, скорость программы вряд ли совместима со сложившимся представлением о скорости, например, какого-либо отдельного клиентского приложения.

C++ с этой точки зрения – соотношение в скорости и защищенности близко к желаемому результату, но чаще всего, как показывает практика, лучше понести незначительную потерю в производительности программы и избежать многих потенциальных ошибок в приложении.

Переносимость. Основное преимущество языка Java выражается в переносимости Java-приложений, т.е. способности работать на любых аппаратных платформах и операционных системах, поскольку все JVM, независимо от того, на какой платформе они работают, способны исполнять один и тот же байт-код. Среды исполнения .NET также кроссплатформенна, поэтому программы, написанные на Java и C#, можно запускать под разными ОС без предварительной перекомпиляции.

C++ – кроссплатформенный язык на уровне компиляции, то есть для него существуют компиляторы под различные платформы.

Синтаксис. C++ сохраняет совместимость с C, насколько это возможно. Java сохраняет внешнее подобие C и C++, но, в действительности, сильно отличается от них: из языка удалено большое число синтаксических средств, объявленных необязательными. В результате программы на Java бывают более громоздки по сравнению с их аналогами на C++. С другой стороны, Java проще, что облегчает как изучение языка, так и создание трансляторов для него.

Язык C# имеет довольно сложный синтаксис (можно утверждать, что примерно 75 % его синтаксических возможностей аналогичны языку программирования Java, 10 % подобны языку программирования C++, а 5 % заимствованы из языка программирования Visual Basic). Объем действительно свежих концептуальных идей в языке C# относительно невысок (по мнению некоторых исследователей, он, составляет около 10% от общего объема конструкций языка).

Web-интеграция. C++, предоставляя преимущества в скорости, оправдан для разработки Web-приложений, гарантируя безопасность и высокую производительность, но не в том случае, когда Web-проект разрабатывается как масштабируемый. Затраты на содержание и поддержку приложения, сложность разработки не оправдают технических достоинств C++.

C# является Web-ориентированным. Компоненты могут быть легко превращены в Web сервисы, к которым можно будет обращаться из Internet посредством любого языка на любой операционной системе. Дополнительные возможности и преимущества перед другими языками приносит в C# использование передовых Web-технологий, таких как XML (Extensible Markup Language) и SOAP (Simple Object Access Protocol).

Что касается Java, то язык и платформа этого языка обладают великолепной масштабируемостью. Java подходит для разработки серверных Web-приложений, при помощи которых пользователь может получать доступ к вычислительным ресурсам в Web. Возможность безопасного выполнения кода, загруженного через сеть, была изначально заложена в конструкцию Java, поэтому этот язык обеспечивает высокий уровень безопасности при работе через Интернет. На Java написан ряд серверов приложений, которые взаимодействуют с базами данных и динамически формируют содержимое Web-страниц, а также предоставляют средства для обеспечения безопасности, связности и необходимого уровня доступности, производительности и масштабируемости. К сожалению, ориентация на Internet не дает возможности использовать Java как язык системного программирования.

Заключение. Языки ООП призваны облегчить программисту процесс решения определенных задач. Но использование даже самых современных технологий не гарантирует успеха конкретного проекта. Однако, сумев грамотно поставить задачу, написать ясный код и сделать работающее решение, удастся обойти небезопасность C++, медлительность Java и сложность C# и воспользоваться сильными сторонами каждого языка программирования.

Список использованных источников

1. Эккель, Б. Философия Java. Библиотека программиста / Б. Энкель. – 4-е изд. – СПб. : Питер, 2009. – 640 с.
2. Страуструп, Б. Язык программирования C++. Специальное издание / Б. Страуструп. – Минск, 2008. – 1130 с.