Особенности компиляции приложений в Visual Studio C#

В дальнейшем при упоминании среды разработки мы не будем указывать конкретную версию, так как на начальном этапе работы практически все версии имеют сходные возможности.

C# спроектирован и разработан специально для применения с платформой .NET[1] Framework. Она обеспечивает совместное использование разных языков программирования, а также безопасность, переносимость программ и общую модель программирования для платформы Windows. Приложения .NET можно создавать с помощью разных языков программирования (C#, Visual Basic, C++, Visual F# и др.). При этом в .NET код, написанный на любом языке, компилируется в код на промежуточном языке (Intermediate Language - IL). В .NET поддерживается межъязыковое наследование, межъязыковая обработка исключений и межъязыковая отладка кода. Платформа .NET использует общий исполняющий механизм, основным аспектом которого является хорошо определенный набор типов, который способен понимать каждый, поддерживающий .NET язык.

Центральной частью каркаса .NET является его общеязыковая исполняющая среда, называемая Common Language Runtime (CLR) или .NET runtime. Код, выполняемый под управлением CLR, часто называют управляемым кодом.

Однако перед тем как код сможет выполняться CLR, любой исходный текст (на C# или другом поддерживаемом языке) должен быть скомпилирован. Компиляция в .NET состоит из двух шагов:

1. Компиляция исходного кода в Microsoft Intermediate Language (IL)
2. Компиляция IL в специфичный для платформы код с помощью CLR

Такой двухшаговый процесс компиляции очень важен. Microsoft Intermediate Language (промежуточный язык Microsoft) определяет идею

[1] Читается как «дот нет»

низкоуровневого языка с простым синтаксисом, который может быть очень быстро оттранслирован в родной машинный код.

Таким образом, при компиляции кода, в котором используется библиотека .NET, он не преобразуется сразу же в код для конкретной операционной системы. Вместо этого он сначала преобразуется в код CIL (код на промежуточном языке). Этот код не является специфическим, ни для какой операционной системы, ни для языка C#. Код, написанный на других языках, например на Visual Basic или C++ тоже компилируется в код на этом языке.

Затем оперативный компилятор (JIT – Jast-in-Time compiler) преобразует (компилирует) CIL в машинный код, отвечающий требованиям конкретной операционной системы и архитектуры компьютера. Только после этого операционная система сможет запустить приложение.

Такой двухуровневый процесс позволяет легко совмещать в одной разработке модули, написанные на разных языках программирования, и позволяет программисту не думать о специфических особенностях их согласования.

Введем несколько определений.

Проект – набор всех файлов, необходимых для создания приложения. Компилятор последовательно обрабатывает файлы проекта и строит из них выполняемый файл.

Решение – это набор всех проектов, которые будут образовывать определенный программный пакет (решение задачи). Решение может состоять из нескольких проектов.

Сборка – в состав сборки входят исполняемые файлы приложений (*.exe-файлы), файлы библиотек (с расширением *.dll), метаинформация (о содержащихся в сборке данных) и файлы с ресурсами (например, текстовые или звуковые файлы). Метаинформация позволяет сборкам быть полностью самостоятельными: для использования сборок не нужно никакой дополнительной информации.

Процесс компиляции приложения, написанного на C#, можно представить следующим образом (Рисунок 1).

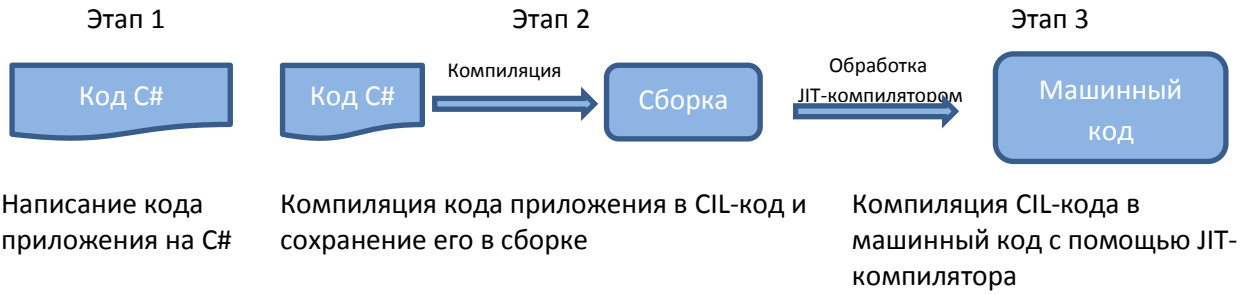

Рисунок 1. Схема процесса двухуровневой компиляции кода приложения на C#

При разработке приложений с помощью Visual Studio требуется создавать решения, состоящие из проектов. Одно решение может содержать проекты разных типов: консольные, Windows, библиотеки, в том числе, написанные на разных языках платформы .NET. Это очень удобно, поскольку позволяет работать совместно нескольким разработчикам.