29. Трансляция и выполнение программы

Автор: Александр
26.08.2014 14:42

Если языки программирования имеет уже более или менее короткую историю развития, то сама технология подготовки программ, написанных на любом языке программирования, вообще сформировались в начале 60 годов и с тех пор не претерпела существенных изменений. Заложенные тогда принципы оказывают влияние на способы использования стандартных библиотечных функций и разработки больших программ, текст которой содержится в нескольких файлах (модульное программирование).

Подготовка программы начинается с редактирования файла, содержащего текст этой программы, который имеет стандартное расширение для данного языка. Затем выполняется его трансляция, которая включает в себя несколько фаз: препроцессор , лексический, синтаксический,
семантический анализ
, генерация кода и его оптимизация. В результате трансляции получается объектный модуль -некий "полуфабрикат" готовой программы, который потом участвует в ее сборке. Файл объектного модуля имеет стандартное расширение ".obj". Компоновка (сборка) программы заключается в объединении одного или нескольких объектных модулей программы и объектных модулей, взятых из библиотечных файлов и содержащих стандартные функции и другие полезные вещи. В результате получается исполняемая программа в виде отдельного файла (загрузочный модуль, программный файл) со стандартным расширением -".exe", который затем загружается в память и выполняется.

Трансляция и ее фазы

Собственно трансляция начинается с лексического анализа программы. ЛЕКСИКА языка программирования - это правила "правописания
слов
" программы, таких как идентификаторы, константы, служебные слова, комментарии. Лексический анализ разбивает текст программы на указанные элементы. Особенность любой лексики - ее элементы представляют собой регулярные линейные последовательности
символов
. Например, ИДЕНТИФИКАТОР - это произвольная последовательность букв, цифр и символа "_", начинающаяся с буквы или "_".

Автор: Александр
26.08.2014 14:42

СИНТАКСИС языка программирования - это правила составления предложений языка из отдельных слов. Такими предложениями являются операции, операторы, определения функций и переменных. Особенностью синтаксиса является принцип вложенности (рекурсивность) правил построения предложений. Это значит, что элемент синтаксиса языка в своем определении прямо или косвенно в одной из его частей содержит сам себя. Например, в определении оператора цикла телом цикла является оператор, частным случаем которого является все тот же оператор цикла.

СЕМАНТИКА языка программирования - это смысл, который закладывается в каждую конструкцию языка. Семантический анализ -это проверка смысловой правильности конструкции. Например, если мы в выражении используем переменную, то она должна быть определена ранее по тексту программы, а из этого определения может быть получен ее тип. Исходя из типа переменной, можно говорит о допустимости операции с данной переменной.

ГЕНЕРАЦИЯ КОДА - это преобразование элементарных действий, полученных в результате лексического, синтаксического и семантического анализа программы, в некоторое внутреннее представление. Это могут быть коды команд, адреса и содержимое памяти данных, либо текст программы на языке Ассемблера, либо стандартизованный промежуточный код (например, Р-код). В процессе генерации кода производится и его оптимизация.

Модульное программирование, компоновка

Полученный в результате трансляции ОБЪЕКТНЫЙ МОДУЛЬ включает в себя готовые к выполнению коды команд, адреса и содержимое памяти данных. Но это касается только собственных внутренних объектов программы (функций и переменных). Обращение к внешним функциям и переменным, отсутствующим в данном фрагменте программы, не может быть полностью переведено во внутреннее представление и остается в объектном модуле в исходном (текстовом) виде. Но если эти функции и переменные отсутствуют, значит, они должны быть каким-то образом получены в других объектных модулях. Самый естественный способ -написать их на том же самом Си и оттранслировать. Это и есть принцип МОДУЛЬНОГО ПРОГРАММИРОВАНИЯ - представление текста программы в виде нескольких файлов, каждый из которых транслируется отдельно. С модульным программированием мы сталкиваемся в двух случаях:

- когда сами пишем модульную программу;

- когда используем стандартные библиотечные функции.

БИБЛИОТЕКА ОБЪЕКТНЫХ МОДУЛЕЙ -это файл (библиотечный файл), содержащий набор объектных модулей и собственный внутренний каталог. Объектные модули библиотеки извлекаются из нее целиком при наличии в них требуемых внешних функций и переменных и используются в процессе компоновки программы.

КОМПОНОВКА - это процесс сборки программы из объектных модулей, в котором производится их объединение в исполняемую программу и связывание вызовов внешних функций и их внутреннего представления (кодов), расположенных в различных объектных модулях. Подробно процесс компоновки применительно к языку Си рассмотрен в 4.6.

В заключение отметим, что источником объектного модуля может быть не только Си-программа, но и программа, написанная на любом другом языке программирования, например, на Ассемблере. Но в этом случае необходимы дополнительные соглашения по поводу "стыковки" вызовов функций и обращений к данным в различных языках.

Сущность трансляции. Компиляция и интерпретация

Под трансляцией в самом широком смысле можно понимать процесс восприятия компьютером программы, написанной на некотором формальном языке. При всем своем различии формальные языки имеют много общего и, в принципе, эквиваленты с точки зрения принципиальной возможности написать одну и ту же программу на одном из них.

Автор: Александр
26.08.2014 14:42

КОМПИЛЯЦИЯ - преобразование объектов (данных и операций над ними) с входного языка в объекты на другом языке для всей программы в целом с последующим выполнением полученной программы в виде отдельного шага.

ИНТЕРПРЕТАЦИЯ - анализ отдельного объекта на входном языке с одновременным выполнением (интерпретацией).

Следовательно, компиляция и интерпретация отличаются не характером и методами анализа и преобразования объектов программы, а совмещением фаз обработки этих объектов во времени. То есть при компиляции фазы преобразования и выполнения действий разнесены во времени, но зато каждая из них выполняется над всеми объектами программы одновременно. При интерпретации, наоборот, преобразование и выполнение действий объединены во времени, но для каждого объекта программы.

Если посмотреть на эти различия несколько с другой стороны, то можно заметить, что интерпретатор непосредственно выполняет действия, связанные с определением или преобразованием объектов программы, а компилятор - переводит их на другой (не обязательно машинный язык). Отсюда можно сделать несколько выводов:

- для выполнения программы, написанной на определенном формальном языке после ее компиляции необходим интерпретатор, выполняющий эту программу, но уже записанную на выходном языке компилятора ;

- процессор и память любого компьютера (а в широком смысле и вся программная среда, создаваемая операционной системой, является ИНТЕРПРЕТАТОРОМ машинного кода);

- в практике построения трансляторов часто встречается случай, когда программа компилируется со входного языка на некоторый промежуточный уровень (внутренний язык), для которого имеется программный интерпретатор. Многие языковые системы программирования, называемые интерпретаторами, на самом деле имеют фазу компиляции во внутренне представление, на котором производится интерпретация.

Автор: Александр
26.08.2014 14:42

Выходной язык компилятора может быть машинным языком для компьютера с другой архитектурой, нежели тот, в котором работает компилятор. Такой компилятор называется КРОСС-КОМПИЛЯТОРОМ, а сама система программирования КРОСС-СИСТЕМОЙ. Такие системы используются для разработки программ для архитектур, не имеющих собственных операционных систем или систем программирования (контроллеры, управляющие микропроцессоры).

Таким образом, граница между компиляцией и интерпретацией в трансляторе может перемещаться от входного языка (тогда мы имеем чистый интерпретатор) до машинного кода (тогда речь идет о чистом компиляторе).

Создание слоя программной интерпретации для некоторого промежуточного языка в практике построения трансляторов обычно встречается при попытке обеспечить совместимость для имеющегося многообразия языков программирования, операционных систем, архитектур и т.д. То есть определяется некоторый внутренний промежуточный язык, достаточно простой, чтобы для него можно было написать интепретатор для всего имеющегося многообразия операционных систем или архитектур. Затем пишется одни (или несколько) компиляторов для одного (или нескольких) входных языков на этот промежуточный уровень. Приведем примеры такой стандартизации:

- для обеспечения совместимости и переносимости трансляторов на компьютеры с различной архитектурой или с различными операционными системами был разработан универсальный внутренний язык (P-код). Для каждой такой архитектуры необходимо реализовать свой интерпретатор P-кода. При этом все разнообразие имеющихся компиляторов с языков высокого уровня на P-код может быть использовано без каких-либо изменений.

- язык программирования Java аналогично был разработан для обеспечения переносимости различных приложений в среде Internet.

Автор: Александр
26.08.2014 14:42

Структура транслятора

Самое главное в процессе трансляции состоит в том, что он не является линейным, то есть последовательным преобразованием фрагмента программы одного языка на другой. На процесс трансляции одного фрагмента обязательно оказывают влияние другие фрагменты программы. Поэтому трансляция представляет собой несколько последовательных фаз анализа программы, на каждой из которой текст программы разделяется на все более "тонкие" компоненты, а информация о них группируется в некоторое внутреннее представление программы (деревья, таблицы). Затем, или параллельно с этим осуществляется синтез программы уже на выходном языке программирования с использованием информации из внутреннего представления.

Отдельные фазы трансляции могут быть связаны между собой различным образом, через данные в памяти или через файл, что не меняет сущности процесса:

- каждая фаза транслятора получает файл данных от предыдущей фазы, обрабатывает его (линейным или каким-либо другим, например, рекурсивным алгоритмом), создает внутренние таблицы данных и по ним формирует выходной файл с данными для следующей фазы;

- фазы трансляции вызывают одна другую в процессе обработки соответствующих языковых единиц. Синтаксический анализ является обычно центральным в такой структуре. То есть основной программой транслятора является синтаксический анализатор, который при анализе структурной единицы языка, называемой предложением (выражение, оператор, определение типа или переменной), вызывает лексический анализатор, для чтения очередной лексической компоненты (идентификатора, константы), а по завершении разбора - семантическую процедуру, процедуры генерации кода или интерпретации. Из этой схемы выпадает только препроцессор
, который обычно представляет собой независимую предварительную фазу трансляции.