**Тема№13. Компиляция и связывание программ**

Цель: изучить технологию программирования компилятор, его возможности, настройку среды, опции компилятора и линкера.

План:

1. Инструментальное программное средство компилятор, компоновщик.
2. Виды компиляторов.
3. Виды компиляции.
4. Структура компилятора.
5. Трансляция и компановка.

**1 Инструментальное программное средство компилятор, компоновщик**

**Компиля́тор** — [компьютерная программа](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) или техническое средство, выполняющее преобразование исходного текста программы, написанного на языке высокого уровня, в машинный язык, язык близкий к машинному, или в объектный модуль. Процесс работы компилятора называется [*компиляцией*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F).

Другие определения компилятора

* Машинная программа, используемая для компиляции.
* [Транслятор](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80), выполняющий преобразование программы, составленной на исходном [языке](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в [объектный модуль](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C).
* Программа, переводящая текст программы на [языке высокого уровня](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%B2%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F), в эквивалентную программу на [машинном языке](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA).
* Программа, предназначенная для трансляции высокоуровневого языка в [абсолютный код](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4&action=edit&redlink=1) или, иногда, в язык ассемблера. Входной информацией для компилятора ([исходный код](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)) является описание алгоритма или программа на [проблемно-ориентированном языке](http://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1), а на выходе компилятора — эквивалентное описание алгоритма на [машинно-ориентированном языке](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) ([объектный код](http://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4&action=edit&redlink=1)).

## 2 Виды компиляторов

* *Векторизующий*. Транслирует [исходный код](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) в [машинный код](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) компьютеров, оснащённых [векторным процессором](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80).
* *Гибкий*. Составлен по [модульному](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) принципу, управляется таблицами и запрограммирован на [языке высокого уровня](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) или реализован с помощью компилятора компиляторов.
* *Диалоговый*.
* *Инкрементальный*. Повторно транслирует фрагменты программы и дополнения к ней без перекомпиляции всей программы.
* *Интерпретирующий (пошаговый)*. Последовательно выполняет независимую компиляцию каждого отдельного [оператора](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (команды) исходной программы.
* [*Компилятор компиляторов*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80%D0%BE%D0%B2). Транслятор, воспринимающий формальное описание [языка программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и генерирующий компилятор для этого языка.
* *Отладочный*. Устраняет отдельные виды [синтаксических ошибок](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B0&action=edit&redlink=1).
* *Резидентный*. Постоянно находится в основной памяти и доступен для повторного использования многими задачами.
* *Самокомпилируемый*. Написан на том же языке, с которого осуществляется трансляция.
* *Универсальный*. Основан на формальном описании синтаксиса и семантики входного языка. Составными частями такого компилятора являются: ядро, [синтаксический](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [семантический](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) загрузчики.

## 

## 3 Виды компиляции

* *Пакетная*. Компиляция нескольких исходных модулей в одном пункте задания.
* *Построчная*. То же, что и [интерпретация](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).
* *Условная*. Компиляция, при которой транслируемый текст зависит от условий, заданных в исходной программе. Так, в зависимости от значения некоторой константы, можно включать или выключать трансляцию части текста программы.

## 

## Основы

Большая часть компиляторов переводят программу с некоторого [высокоуровневого языка программирования](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) в [машинный код](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), который может быть непосредственно выполнен [центральным процессором](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80). Как правило, этот код также должен выполняться в среде конкретной [операционной системы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), поскольку использует предоставляемые ей возможности ([системные вызовы](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2), библиотеки функций). Архитектура (набор программно-аппаратных средств), для которой производится компиляция, называется *целевой машиной*.

Некоторые компиляторы (например, [Java](http://ru.wikipedia.org/wiki/Java)) переводят программу не в машинный код, а в программу на некотором специально созданном [низкоуровневом языке](http://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Такой язык — *байт-код* — также можно считать языком машинных команд, поскольку он подлежит интерпретации *виртуальной машиной*. Например, для языка Java это [JVM](http://ru.wikipedia.org/wiki/JVM) (язык [виртуальной машины](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) Java), или так называемый *байт-код* Java (вслед за ним все промежуточные низкоуровневые языки стали называть байт-кодами). Для языков программирования на платформе [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) ([C#](http://ru.wikipedia.org/wiki/C_Sharp), Managed C++, Visual Basic .NET и другие) это [MSIL](http://ru.wikipedia.org/wiki/MSIL) (Microsoft Intermediate Language, «Промежуточный язык фирмы Майкрософт»).

Программа на байт-коде подлежит [интерпретации](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%86%D0%B8%D1%8F) *виртуальной машиной*, либо ещё одной компиляции уже в машинный код непосредственно перед исполнением. Последнее называеется «Just-In-Time компиляция» (JIT), по названию подобного компилятора для Java. [MSIL](http://ru.wikipedia.org/wiki/MSIL)-код компилируется в код целевой машины также JIT-компилятором, а библиотеки [.NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) компилируются заранее).

Для каждой целевой машины ([IBM](http://ru.wikipedia.org/wiki/IBM), [Apple](http://ru.wikipedia.org/wiki/Apple) и т. д.) и каждой операционной системы или семейства операционных систем, работающих на целевой машине, требуется написание своего компилятора. Существуют также так называемые [*кросс-компиляторы*](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), позволяющие на одной машине и в среде одной ОС получать код, предназначенный для выполнения на другой целевой машине и/или в среде другой ОС. Кроме того, компиляторы могут быть оптимизированы под разные типы [процессоров](http://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) из одного семейства (путём использования специфичных для этих процессоров инструкций). Например, код, скомпилированный под процессоры семейства [i686](http://ru.wikipedia.org/wiki/I686), может использовать специфичные для этих процессоров наборы инструкций — [MMX](http://ru.wikipedia.org/wiki/MMX), [SSE](http://ru.wikipedia.org/wiki/SSE), [SSE2](http://ru.wikipedia.org/wiki/SSE2).

Также существуют компиляторы, переводящие программу с языка высокого уровня на [ассемблер](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80).

Существуют программы, которые решают обратную задачу — перевод программы с низкоуровневого языка на высокоуровневый. Этот процесс называют декомпиляцией, а программы — [декомпиляторами](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Но, поскольку компиляция — это процесс с потерями, точно восстановить исходный код, скажем, на C++ в общем случае невозможно. Более эффективно декомпилируются программы в байт-кодах — например, существует довольно надёжный декомпилятор для [Flash](http://ru.wikipedia.org/wiki/Adobe_Flash). Сходным процессом является [дизассемблирование](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B7%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) машинного кода в код на [ассемблере](http://ru.wikipedia.org/w/index.php?title=%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B5&action=edit&redlink=1), который всегда выполняется успешно. Связано это с тем, что между кодами машинных команд и командами ассемблера — однозначные соответствия.

## 4 Структура компилятора

Процесс компиляции состоит из следующих этапов:

1. [Лексический анализ](http://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7). На этом этапе последовательность символов исходного файла преобразуется в последовательность [лексем](http://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B5%D0%BC%D0%B0).
2. [Синтаксический (грамматический) анализ](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7). Последовательность лексем преобразуется в дерево разбора.
3. [Семантический анализ](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7&action=edit&redlink=1). Дерево разбора обрабатывается с целью установления его семантики (смысла) — напр. привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом», и может быть дополненным деревом разбора, новым деревом, абстрактным набором команд или чем-то ещё, удобным для дальнейшей обработки.
4. [Оптимизация](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80%D0%B0). Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах, напр. над промежуточным кодом или над конечным машинным кодом.
5. [Генерация кода](http://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%B4%D0%B0&action=edit&redlink=1). Из промежуточного представления выдается код на целевом языке.

В конкретных реализациях компиляторов, эти этапы могут быть раздельны или совмещены в том или ином виде.

## 

## 5 Трансляция и компоновка

Важной исторической особенностью компилятора, отраженной в его названии ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *compile* — собирать вместе, составлять), являлось то, что он мог производить и [*компоновку*](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA) (то есть содержал две части — транслятор и компоновщик). Это связано с тем, что [раздельная компиляция](http://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F&action=edit&redlink=1) и компоновка как отдельная стадия сборки выделились значительно позже появления компиляторов, и многие популярные компиляторы (например, [GCC](http://ru.wikipedia.org/wiki/GCC)) до сих пор физически объединены со своими компоновщиками. В связи с этим, **вместо термина «компилятор» иногда используют термин «транслятор» как его синоним**: либо в старой литературе, либо когда хотят подчеркнуть его способность переводить программу в машинный код (и наоборот, используют термин «компилятор» для подчеркивания способности собирать из многих файлов один).

# Компиляция

Компиляция и тестирование справочной службы не составляют проблемы: с помощью MS HW создайте (или загрузите) проектный файл и щелкните по кнопке save and compile или по инструментальной кнопке*.* После этого появляется окно запуска компилятора.

При компиляции MS HW сворачивает свое окно и вновь раскрывает его после завершения работы компилятора, если отмечен переключатель Minimize window while compiling. После окончания компиляции в нем будут показаны сообщения компилятора. Если компилятор обнаружил ошибки, он сообщает о них, причем некритические ошибки сопровождаются предупреждениями *(warnings)* и замечаниями *(notes),* а критические - прерывают компиляцию.

Для тестирования скомпилированного справочного файла используется одна из опций меню Test:

**contents File** - тестирует файл содержания;

**close All Help** - закрывает все ранее открытые нlр-файлы;

**send a macro** - посылает в winHelp нужную макрокоманду;

**WinHelp api** - вызывает нужный раздел справочной службы по присвоенному ему в секции map числовому идентификатору.

Тестирование файла содержания заключается в автоматическом вызове всех указанных в содержании разделов. Если какой-либо раздел не вызывается, выдается сообщение, позволяющее найти и устранить ошибку.

Связь с программой реализуется с помощью свойств HelpContext видимых компонентов, в которые следует поместить числовые идентификаторы нужных разделов справочной службы так, как они определены. В секции MAP. Кроме того, в свойство Application. HelpFile нужно поместить имя нlр-файла. Обычно эта связь устанавливается в обработчике события ОnCreat главной формы программы. После такой настройки пользователь программы сможет с помощью клавиши F1 получить контекстно-чувствительную справку, т. к. при нажатии F1 автоматически вызывается раздел, числовой идентификатор которого помещен в свойство HeipContext компонента с фокусом ввода. Если Heipcontext компонента с фокусом ввода содержит 0, вызывается раздел, указанный в HelpContext его владельца, а если и у того это свойство не определено, используется HelpContext активной формы (если во всей цепочке владельцев, включая активную форму, свойство HelpContext не определено, нажатие F1 игнорируется).

**Определение условий компиляции**

Подобно директивам условной компиляции Delphi в RTF-файл можно вставлять указания Help-компилятору помещать или не помещать в результирующий файл тот или иной раздел. Такие указания могут понадобиться на этапе отладки справочной службы, а также при создании нескольких версий приложения (например, справочная служба демонстрационной версии может не содержать некоторые разделы, определенные для основной версии).Они вставляются в текст с помощью сноски, помеченной символом “\*”. Чтобы включить условный раздел в Help-файл или исключить его из файла, ключевое слово, определяемое сноской “\*”, должно указываться в секции соответственно include или Exclude файла проекта справочной службы.

В тексте сноски “\*” можно указать одно или несколько управляющих слов (следующие друг за другом слова в тексте сноски разделяются символом “;”). Раздел будет включен в Help-файл, если хотя бы одно из связанных с ним управляющих слов указано в секции include файла проекта (если раздел не имеет сноски “\*”, он всегда включается в результирующий файл). Управляющие слова могут содержать любые символы, кроме “;” и пробелов. Например:

Test\_Build; AppVersionI; DebuggVer

**Контрольные вопросы:**

* 1. Что называют компилятором?
  2. Перечислите виды компиляторов.
  3. Перечислите виды компиляции.
  4. Поясните стркутуру компилятора (т.е. из каких этопов состоит процесс компиляции).
  5. Поясните назначение терминов трансляция и компановка.