**Директивы #if и #endif**

Обе директивы, #if и #endif, допускают условную компиляцию последовательности

кода в зависимости от истинного результата вычисления выражения, включающего

в себя один или несколько идентификаторов. Идентификатор считается истинным,

если он определен, а иначе — ложным. Так, если идентификатор определен директивой

#define, то он будет оценен как истинный. Ниже приведена общая форма директивы #if.

#if идентификаторное\_выражение

последовательность операторов

#endif

Если *идентификаторное\_выражение,* следующее после директивы #if, истинно,

то компилируется код (*последовательность операторов*), указываемый между ним

и директивой #endif. В противном случае этот промежуточный код пропускается.

Директива #endif обозначает конец блока директивы #if.

Идентификаторное выражение может быть простым, как наименование идентификатора.

В то же время в нем разрешается применение следующих операторов: !, ==,

!=, && и ||, а также круглых скобок.

Ниже приведен пример применения упомянутых выше директив.

(***glava16\_9***)

#define EXPERIMENTAL

using System;

class Test

{

static void Main()

{

#if EXPERIMENTAL

Console.WriteLine("Compiles for experimental version.");

#endif

Console.WriteLine("In all versies.");

}

}

В приведенном выше коде определяется идентификатор EXPERIMENTAL. Поэтому

когда в этом коде встречается директива #if, идентификаторное выражение вычисляется

как истинное и затем компилируется первый оператор, содержащий вызов

метода WriteLine(). Если же удалить определение идентификатора EXPERIMENTAL

и перекомпилировать данный код, то первый оператор, содержащий вызов метода

WriteLine(), не будет скомпилирован, поскольку идентификаторное выражение директивы

#if вычисляется как ложное. Но второй оператор, содержащий вызов метода

WriteLine(), компилируется в любом случае, потому что он не входит в блок директивы

#if.

Как пояснялось выше, в директиве #if допускается указывать идентификаторное

выражение. В качестве примера рассмотрим следующую программу.

(***glava16\_9.1***)

#define EXPERIMENTAL

#define TRIAL

using System;

class Test

{

static void Main()

{

#if EXPERIMENTAL

Console.WriteLine("Compiles for experimental version.");

#endif

#if EXPERIMENTAL && TRIAL

Console.WriteLine("Check of experimental version.");

#endif

Console.WriteLine("In all versies.");

}

}

В данном примере определены два идентификатора: EXPERIMENTAL и TRIAL. Второй

оператор, содержащий вызов метода WriteLine(), компилируется лишь в том

случае, если определены оба идентификатора.

Для компилирования кода в том случае, если идентификатор не определен, можно

воспользоваться оператором !, как в приведенном ниже примере.

#if !EXPERIMENTAL

Console.WriteLine("Этот код не экспериментальный!");

#endif

Вызов метода будет скомпилирован только в том случае, если идентификатор

EXPERIMENTAL ***не***определен.

**Директивы #else и #elif**

Директива #else действует аналогично условному оператору else языка С#, определяя

альтернативный ход выполнения программы, если этого не может сделать директива

#if. С учетом директивы #else предыдущий пример программы может быть

расширен следующим образом.

(***glava16\_10***)

#define EXPERIMENTAL

using System;

class Test

{

static void Main()

{

#if EXPERIMENTAL

Console.WriteLine("Compiles for experimental version.");

#else

Console.WriteLine("For ended version.");

#endif

#if EXPERIMENTAL && TRIAL

Console.WriteLine("Check of experimental version.");

#else//потому что нет Trial, выполнится этот код, а не тот что выше

Console.WriteLine("Thi is not experimental version");

#endif

Console.WriteLine("In all versies.");

}

}

Обратите внимание на то, что директива #else обозначает конец блока директивы

#if и в то же время — начало блока самой директивы #else. Это необходимо

потому, что с любой директивой #if может быть связана только одна директива

#endif. Более того, с любой директивой #if может быть связана только одна директива

#else.

Обозначение #elif означает *"*иначе если*"*, а сама директива #elif определяет последовательность

условных операций if-else-if для многовариантной компиляции.

После директивы #elif указывается идентификаторное выражение. Если это выражение

истинно, то компилируется следующий далее кодовый блок, а остальные выражения

директивы #elif не проверяются. В противном случае проверяется следующий

по порядку блок. Если же ни одну из директив #elif не удается выполнить, то при

наличии директивы #else выполняется последовательность кода, связанная с этой директивой,

а иначе не компилируется ни один из кодовых блоков директивы #if.

Ниже приведена общая форма директивы #elif.

#if идентификаторное\_выражение

последовательность операторов

#elif идентификаторное\_выражение

последовательность операторов

#elif идентификаторное\_выражение

последовательность операторов

// ...

#endif

В приведенном ниже примере демонстрируется применение директивы #elif.

(***glava16\_11***)

#define EXPERIMENTAL

//#define RELEASE

//#define TRIAL

using System;

class Test

{

static void Main()

{

#if EXPERIMENTAL

Console.WriteLine("Compiles for experimental version.");

#elif RELEASE

Console.WriteLine("For ended version.");

#else

Console.WriteLine("Check of experimental version.");

#endif

#if !RELEASE && TRIAL

Console.WriteLine("Check of experimental version.");

#endif

Console.WriteLine("In all versies.");

}

}

**Директива #undef**

С помощью директивы #undef удаляется определенный ранее идентификатор.

Это, по существу, означает, что он становится *"*неопределенным*"*. Ниже приведена общая

форма директивы #undef.

#undef идентификатор

Рассмотрим следующий пример кода.

#define SMALL

#if SMALL

// ...

#undef SMALL

// теперь идентификатор SMALL не определен.

После директивы #undef идентификатор SMALL уже оказывается неопределенным.

Директива #undef применяется главным образом для локализации идентификаторов

только в тех фрагментах кода, в которых они действительно требуются.

**Директива #error**

Директива #error вынуждает компилятор прервать компиляцию. Она служит в

основном для отладки. Ниже приведена общая форма директивы #error.

#error сообщение\_об\_ошибке

Когда в коде встречается директива #error, выводится сообщение об ошибке. Например,

когда компилятору встречается строка кода

#error Это тестовая ошибка!

компиляция прерывается и выводится сообщение "Это тестовая ошибка!".

**Директива #warning**

Директива #warning действует аналогично директиве #error, за исключением

того, что она выводит предупреждение, а не ошибку. Следовательно, компиляция не

прерывается. Ниже приведена общая форма директивы #warning.

#warning предупреждающее\_сообщение

**Директива #line**

Директива #line задает номер строки и имя файла, содержащего эту директиву.

Номер строки и имя файла используются при выводе ошибок или предупреждений

во время компиляции. Ниже приведена общая форма директивы #line.

#line номер "имя\_файла"

Имеются еще два варианта директивы #line. В первом из них она указывается с

ключевым словом default, обозначающим возврат нумерации строк в исходное состояние,

как в приведенном ниже примере.

#line default

А во втором варианте директива #line указывается с ключевым словом hidden.

При пошаговой отладке программы строки кода, находящиеся между директивой

#line hidden

и следующей директивой #line без ключевого слова hidden, пропускаются отладчиком.

**Директивы #region и #endregion**

С помощью директив #region и #endregion определяется область, которая разворачивается

или сворачивается при структурировании исходного кода в интегрированной

среде разработки Visual Studio. Ниже приведена общая форма этих директив:

#region текст

// последовательность кода

#endregion текст

где *текст* обозначает необязательную символьную строку.

**Директива #pragma**

С помощью директивы #pragma инструкции задаются компилятору в виде опций.

Ниже приведена общая форма этой директивы:

#pragma опция

где *опция* обозначает инструкцию, передаваемую компилятору.

В текущей версии C# предусмотрены две опции для директивы #pragma. Первая

из них, warning, служит для разрешения или запрета отдельных предупреждений со

стороны компилятора. Она принимает две формы:

#pragma warning disable предупреждения

#pragma warning restore предупреждения

где *предупреждения* обозначает разделяемый запятыми список номеров предупреждений.

Для отмены предупреждения используется опция disable, а для его разрешения

— опция restore.

Например, в приведенной ниже директиве #pragma запрещается выдача предупреждения

№168, уведомляющего о том, что переменная объявлена, но не используется.

#pragma warning disable 168

Второй для директивы #pragma является опция checksum. Она служит для формирования

контрольной суммы в проектах ASP.NET. Ниже приведена ее общая форма:

#pragma checksum "имя\_файла" "{GUID}" "контрольная\_сумма"

где *имя\_файла* обозначает конкретное имя файла; *GUID* — глобально уникальный

идентификатор, с которым связано *имя\_файла*; *контрольная\_сумма* — шестнадцатеричное

число, представляющее контрольную сумму. У этой контрольной суммы должно

быть четное число цифр.