Оглавление

[Список терминов 1](#_Toc485225389)

[Правила именования переменных 1](#_Toc485225390)

[Общие правила именования идентификаторов 2](#_Toc485225391)

[Правила именования классов, интерфейсов 2](#_Toc485225392)

[Правила именования полей класса 2](#_Toc485225393)

[Правила именования методов класса 3](#_Toc485225394)

[Правила именования свойств класса 3](#_Toc485225395)

[Правила именования локальных переменных 3](#_Toc485225396)

[Правила форматирования, размещения кода 3](#_Toc485225397)

[Комментарии 4](#_Toc485225398)

[Другое 5](#_Toc485225399)

[Список использованных материалов 5](#_Toc485225400)

[Приложение 1: Примеры префиксов, используемых для именования элементов управления. 6](#_Toc485225401)

[Приложение 2: Сводная таблица правил именования 6](#_Toc485225402)

# Список терминов

В русском языке есть некоторые разночтения в названии некоторых частей программы. Во избежание путаницы в этом разделе будут приведены часто встречающиеся термины, их английские аналоги и краткое пояснение.

* operator – оператор (например, +, -, \*, !);
* statement – инструкция (например, «a = b;», «if (a) {}»);
* expression – выражение (например, «a – b + c», «a == 1»).

**Стили именования:**

* *Pascal case (паскаль)* - указание этого стиля оформления идентификатора обозначает, что первая буква заглавная и все последующие первые буквы слов тоже заглавные, например, BackColor, LastModified, DateTime;
* *Camel case (Кэмел)*  – первая буква строчная, а остальные первые буквы слов заглавные, например: backColor, lastModified, dateTime;
* *UpperCase* – стиль используется только для сокращений, все буквы в имени идентификатора в верхнем регистре, например,  ID;
* *Hungarian notation (венгерская нотация)* – перед именем идентификатора пишется его тип в сокращенной форме, например, strFirstName, iCurrentYear.

# Правила именования переменных

## Общие правила именования идентификаторов

* Не используйте аббревиатуры или неполные слова, акронимы в идентификаторах, если только они не являются общепринятыми. Например, пишите GetWindow, а не GetWin; MyDogName вместо MDN;
* Широко распространенные акронимы используйте для замены длинных фраз. Например, UI вместо User Interface или Olap вместо On-line Analytical Processing.
* Если имеется идентификатор длиной менее трех букв, являющийся сокращением, то его записывают заглавными буквами, например System.IO, System.Web.UI. Имена длиннее двух букв записывайте в стиле Паскаль или Кэмел, например Guid, Xml, xmlDocument.
* Необходимо использовать имена, которые ясно и четко описывают предназначение и/или смысл сущности, Используйте названия, которые будут без искажений передавать ваши идеи в коде. Примеры плохих идентификаторов:
  + od: Что значит od? Очередной день? Оператор доставки? Опция доступа?
  + Button1\_Click(): Что за Button1? И чем отличается от Button2?
  + thisAs: А это что вообще такое?!

## Правила именования классов, интерфейсов

* Следует избегать имен классов, совпадающих с именами классов .NET Framework;
* Для классов используется стиль именования pascal case;
* В качестве имен классов используются имена существительные;

public class Employee

{

}

public class BusinessLocation

{

}

public class DocumentCollection

{

}

## Правила именования полей класса

* Непубличные поля (private, protected и protected internal) именуются в стиле Кэмел и могут начинаться с префикса \_, например, \_backColor, \_lastModified, \_dateTime.
* Имена публичных (internal и public) переменных должны быть в стиле Паскаль и не иметь подчеркивания в качестве префикса.
* Если класс представляет собой сущность, хранимую в базе данных – имя класса соответствует имени таблицы. В этом случае имя класса – это название сущности в единственном числе, имя таблицы – во множественном числе. Пример: таблица Users, класс User;
* Имена файлов, в которых находятся классы, совпадают с именами классов. Для именования файлов используется стиль pascal case.
* Имена интерфейсов начинаются с буквы I, после которой следует название интерфейса в pascal case. Пример: IDisposable.
* Для именования элементов управления можно использовать венгерскую нотация (стиль Hungarian notation). См. Приложение 1: Примеры префиксов, используемых для именования элементов управления.

public class Employee

{

private int \_id;

private DateTime \_birthDate;

internal int DepartmentId;

}

public interface IGroupable

{

}

## Правила именования методов класса

* Используйте глаголы или комбинацию глагола и существительных и прилагательных для имен методов.
* Используйте стиль Паскаль для именования метода (вне зависимости от области видимости метода).
* Для именования параметров используется стиль camel case;

|  |
| --- |
| Type GetType(string typeName);  string Format(string format, params object[] args); |

## Правила именования свойств класса

* Используйте существительное или одно или несколько прилагательных и существительное для имени свойства.
* Используйте стиль Паскаль для регистра букв.
* Название свойства типа bool должно представлять из себя вопрос, требующий ответа да или нет. Примеры названий: “CanDownload”, “HasKeywords”, “IsChecked”, “NeedsUpdate”.

## Правила именования локальных переменных

* Используйте стиль Кэмел для регистра букв в именах переменных.
* Объявляйте переменные непосредственно перед их использованием.
* Счетчики в циклах традиционно называют i, j, k, l, m, n.

int itemCount = 10;

for( int i = 0; i < itemCount; ++i ){

}

# Правила форматирования, размещения кода

Используйте форматирование, чтобы подчеркнуть структуру вашего кода и сделать код более удобным для чтения. Рекомендуется следовать следующим соглашениям:

* В одном файле не объявляется больше одного namespace’а и одного класса (исключение – небольшие вспомогательные private классы и вложенные типы, делегаты);
* Все подключения namespace’ов (using) размещаются в начале файла.
* Объявляйте все переменные-члены в верхней части класса, статические переменные – на самом верху.

public class Account

{

public static string BankName;

public string Number;

public DateTime DateOpened {get; set;}

// Constructor

public Account()

{

// ...

}

}

* Одно объявление должно содержать не более одного поля и должно располагаться на одной строке.

class A

{

// Так нельзя:

int \_var1, \_var2;

// Так тоже:

int \_var1,

\_var2;

// Надо так:

int \_var1;

int \_var2;

...

* Используйте круглые скобки, чтобы сделать выражения более очевидными, как показано в следующем коде:

if ((val1 > val2) && (val1 > val3))

* Код, расположенный внутри фигурных скобок, должен располагаться с отступом в один символ tab’а. Допустимые варианты расстановки фигурных скобок:

|  |  |
| --- | --- |
|  |  |

* Размер tab’а – 4 пробела.
* В условии if-else, while, for, и т.п. рекомендуется всегда использовать фигурные скобки. Это увеличивает читаемость кода и упрощает дальнейшую поддержку.
* Использование строк длиннее 100 символов не желательно. При необходимости инструкция переносится на другую строку. При переносе части кода на другую строку вторая и последующая строки сдвигаются вправо на один символ табуляции;

# Комментарии

* Все комментарии должны быть на русском языке;
* Там, где это возможно, располагайте комментарии над кодом, а не в конце строки.  Далее приведены несколько примеров:

// Create object to access Internet resources  
WebClient myClient = new WebClient();

Комментарии могут быть размещены в конце строки, если позволяет свободное пространство:

public class SomethingUseful   
{  
    private int          itemHash;            // instance member  
    private static bool  hasDoneSomething;    // static member  
}

* Для функций, классов, enum’ов, свойств и полей комментарии необходимо указывать в таком виде, чтобы по ним можно было автоматически сгенерировать документацию. Для этого используются стандартные tag’и такие как <summary>, <param> и <return>.
* Для функций создающих exception’ы – возможные исключения необходимо указывать в tag’ах <exception>.
* Для функций, выполняющих сложные алгоритмы, не очевидные для восприятия, необходимо указывать подробные комментарии не только к заголовку функции, но и самому алгоритму с пояснением каждого шага выполнения алгоритма.

# Другое

* В том случае если в get’е или set’е какого-либо свойства выполняются сложные вычисления, если операция, выполняемая в get или set является преобразованием, имеет побочный эффект или долго выполняется – свойство должно быть заменено функциями;
* Свойство не должно менять своего значения от вызова к вызову, если состояние объекта не изменяется. Если результат при новом вызове может быть другим при том же состоянии объекта, вместо свойства необходимо использовать функции;
* Вместо использования “магических чисел” (что это такое, см. [википедию](https://ru.wikipedia.org/wiki/Магическое_число_(программирование)) ) для идентификаторов статусов, состояний и т.п. необходимо указывать константы или enum’ы. Идентификаторы состояний в виде чисел использовать нельзя.
* Все настройки, влияющие на работу приложения, нельзя указывать жестко в коде, а необходимо выносить в config. Если  есть возможность прописать значение по умолчанию – они должны быть прописаны, если значение по умолчанию не может быть задано и соответствующий ключ не прописан в конфиге – необходимо создавать исключение.
* При обработке исключений пустые секции catch использовать нельзя;

# Список использованных материалов

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

<http://michaelsmirnov.blogspot.ru/2011/01/c.html>

<http://www.dofactory.com/reference/csharp-coding-standards>

<https://blogs.msdn.microsoft.com/brada/2005/01/26/internal-coding-guidelines/>

<http://rsdn.org/article/mag/200401/codestyle.XML>

# Приложение 1: Примеры префиксов, используемых для именования элементов управления.

* Button – btn;
* CheckBox – cb;
* Image – img;
* ImageButton – ibtn;
* Label – l;
* LinkButton – lbtn;
* ListBox – lb;
* Panel – pnl;
* RadioButton – rb;
* TextBox – tb;
* …

# Приложение 2: Сводная таблица правил именования

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Регистр** | **Пример** |
| Класс | Pascal | User |
| Локальная переменная | Camel | User |
| Интерфейс | Pascal | IDisposable |
| Public функция | Pascal | Authenticate |
| Private функция | Pascal | Authenticate |
| Параметр функции | Camel | userID |
| Public свойство | Pascal | FirstName |
| Private свойство | Pascal | FirstName |
| Public поле | Pascal | FirstName |
| Private поле | Camel | firstName или \_firstName |