Перечень запретов, рекомендаций и требований для языка С#

**Запреты:**

1. Избегай использования неявного типа данных (`var`) без необходимости:

Пример:

Не рекомендуется:

csharp

var number = 10; // Плохо

Рекомендуется:

csharp

int number = 10; // Хорошо

2. Не использовать пустые `catch` блоки для обработки исключений:

Пример:

Не рекомендуется:

csharp

try

{

// Код, который может вызвать исключение

}

catch

{

}

Рекомендуется:

csharp

try

{

// Код, который может вызвать исключение

}

catch (Exception ex)

{

// Обработка исключения

}

3. Избегай использования магических чисел в коде:

Магические числа - это значения, которые появляются напрямую в коде без объяснения их смысла. Их использование может сделать код менее читаемым, подверженным ошибкам и сложным для поддержки и изменений в будущем. Вместо использования этих чисел напрямую, рекомендуется объявлять их как константы с понятными именами.

Пример: Не рекомендуется:

csharp

if (age > 18 && age < 65) // 18 и 65 - магические числа

{

//код

}

```

Рекомендуется:

csharp

private const int AdultAge = 18;

private const int SeniorAge = 65;

if (age > AdultAge && age < SeniorAge)

{

// код

}

```

**Рекомендации:**

1. Используйте подходящие имена переменных и методов в соответствии с их назначением:

Пример:

csharp

// Плохо

int x = 5;

if (x > 0) { /код/ }

// Хорошо

int positiveNumber = 5;

if (positiveNumber > 0) { /код/ }

2. Используйте LINQ для выполнения запросов к данным вместо циклов:

Пример:

csharp

// Плохо

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };

int sum = 0;

foreach (int num in numbers)

{

sum += num;

}

// Хорошо

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };

int sum = numbers.Sum();

3. Используйте комментарии для пояснения сложного кода или там, где логика неочевидна:

Пример:

csharp

// Добавляем нового пользователя в базу данных

User newUser = new User("Alice", 25);

userService.Add(newUser);

**Требования:**

1. Используйте camelCase для именования переменных и методов:

Пример:

csharp

string fullName;

int userAge;

void calculateTotalSum() { /код/ }

2. Соблюдайте принципы SOLID и DRY в организации кода:

Пример:

csharp

// Принцип DRY: избегайте дублирования кода

private void CalculateTotal(int a, int b)

{

// Логика для расчета

}

// Принцип SOLID: принцип единственной ответственности

public class UserService

{

public void AddUser(User user)

{

// Логика добавления пользователя

}

3. Используйте понятные и информативные сообщения об ошибках и логирование для отслеживания событий:

Пример:

csharp

try

{

// Логика, которая может вызвать исключение

}

catch (Exception ex)

{

logger.LogError(ex, "Произошла ошибка при выполнении операции X");

throw;

}

**Чек-лист на основе ограничений и рекомендаций для языка C#:**

| 1 | Избегай использования неявного типа данных var без необходимости. Использование явного типа данных вместо var, если тип данных явно известен. |  |
| --- | --- | --- |
| 2 | Не использовать пустые catch блоки для обработки исключений. Обработка исключений с указанием типа исключения для более точной обработки. |  |
| 3 | Избегай использования магических чисел в коде. Определение констант для использования вместо магических чисел. |  |
| 4 | Используйте понятные и информативные имена переменных и методов. Использование имен переменных, методов и классов, понятных и описательных. |  |
| 5 | Используйте LINQ для выполнения запросов к данным вместо циклов. Применение LINQ для манипуляции данными, вместо явных циклов. |  |
| 6 | Используйте комментарии для пояснения сложного кода или там, где логика неочевидна. Добавление комментариев там, где код может быть непонятен или сложен. |  |
| 7 | Используйте camelCase для именования переменных и методов. Использование camelCase для именования переменных и методов. |  |
| 8 | Соблюдайте принципы SOLID и DRY в организации кода. Избегайте повторения кода (DRY) и поддерживайте принципы SOLID в организации кода. |  |
| 9 | Используйте понятные и информативные сообщения об ошибках и логирование. Обработка исключений с логированием и понятными сообщениями об ошибках. |  |
| 10 | Создание констант для значений, которые могут потенциально меняться в будущем. |  |