

Культура програмування у .NET Framework

Техніка написання коду включає багато аспектів розробки програмного забезпечення. Вони не впливають на функціональність, зате покращують розуміння вихідного тексту.

ІМЕНА

- Назва має говорити «Що», а не «Як». Рекомендується уникати імен, які пояснюють внутрішню реалізацію метода (яка може змінитись). Наприклад `GetNextStudent()` замість `GetNextArrayElement()`.
- Робіть назви достатньо довгими, щоб вони були значущими, але й достатньо короткими.

Підпрограми, процедури:

- Уникайте назв, що можуть бути інтерпретовані суб'єктивно (`AnalyzeThis()`, `xxK8`).
- В об'єктно-орієнтованих мовах надлишково включати назву класу в назву властивості. Вживайте `Book.Title` замість `Book.BookTitle`.
- Називайте методи, які виконують деяку дію над об'єктом відповідним дієсловом. Наприклад, `CalculateInvoiceTotal()`.
- При перевантаженні процедур, усі варіанти мають виконувати однакову функцію.

Змінні

- Додавайте кваліфікатори обчислення (Avg, Sum, Min, Max, Index) до назви змінної.
- Використовуйте парні назви для змінних-антонімів (min/max, begin/end, open/close).
- Для складних назв внутрішні слова починайте з великої літери. Для назв функцій – CalculateInvoiceTotal, для змінних – documentFormatType.
- В бульових змінних варто використовувати приставку Is (fileIsFound).
- Уникайте використання абстрактних назв статусів як Flag для не булевих змінних. Замість documentFlag -> documentFormatType
- Давайте зрозумілі назви навіть для змінних життєвий цикл яких короткий. Змінні i, j використовуйте тільки в коротких циклах (for i=1 to totalNumber).
- Не використовуйте літеральні числа і рядки. Використовуйте константи. Замість for i= 1 to 7 => for i=1 to NUM_DAYS_OF_WEEK

Таблиці

- ⦿ Давайте назви таблиць в одиничній формі (Student замість Students).
- ⦿ Називаючи стовпчики, уникайте повторення назви таблиці. Наприклад, не створюйте колонку StudentName в таблиці Student
- ⦿ Не включайте у назву колонки тип даних. Тип описує інша сутність.

Різне

- ⦿ Мінімізуйте використання абревіатур. Ті, що використовуєте – використовуйте всюди в одному значенні. Наприклад, якщо `min` – це мінімум, то не використовуйте її для позначення `minute`.
- ⦿ При наданні імені функції включайте опис значення, що повертається: `GetCurrentWindowName()`.
- ⦿ Назви файлів та директорій мають описувати їх призначення.
- ⦿ Не використовуйте ім'я повторно для різних елементів. Наприклад функція `ProcessSales()` та змінна `iProcessSales`.
- ⦿ Уникайте омонімів (наприклад, `write and right`) для полегшення подальшого перегляду коду.
- ⦿ По можливості уникайте орфографічних помилок.

Документація програми існує у двох формах. *Зовнішня* документація – специфікації, файли допомоги, документи дизайну підтримуються поза кодом. *Внутрішня* складається з коментарів, що лишають програмісти під час розробки.

Внутрішня документація має модифікуватись паралельно із написанням коду. Вона не дає користі під час виконання і використання програми але є безцінною для програміста, який буде підтримувати цей код далі.

Рекомендовані техніки коментування коду:

- При розробці в C# використовуйте утиліту XML Documentation.
- При редагуванні коду – модифікуйте і коментарі
- На початку кожної функції рекомендується описати її призначення, параметри, що приймає, повертає та обмеження.
- Не використовуйте коментарі в кінці рядка. Виключення – коментування оголошень змінних (в такому разі вирівнюйте усі коментарі табом).
- Не використовуйте коментарі без смислового навантаження (наприклад рядок плюсів) Натомість – відділіть код від коментарів пустим рядком.
- Не виділяйте коментар. Це виглядає привабливо, але заважає підтримці коду.
- Перед інсталяцією видаліть усі тимчасові коментарі.

ПРОЦЕСИ ТА СИСТЕМИ ПІДТРИМКИ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

- Якщо це можливо – не коментуйте поганий код – перепишіть його. Це важливо для підтримки. Але не жертвуйте продуктивністю заради читабельності. Необхідно знайти золоту середину між продуктивністю і читабельністю.
- Використовуйте завершені речення.
- Коментуйте під час написання коду. Ви можете не мати часу зробити це пізніше.
- Не використовуйте недоречні коментарі (жарти, анекдоти).
- Використовуйте коментарі для пояснення призначення коду (а не перекладу).
- Коментуйте усе, що не очевидно з коду.
- Для запобігання повторної появи проблеми, завжди коментуйте правки помилок або *work-around-code*, особливо в при командній розробці.
- Коментуйте цикли та логічні переходи – це ключові місця коду
- Використовуйте стандартизовані коментарі по усьому коду програми.
- Відділяйте коментарі від сепараторів пробілом. Це полегшує читання без кольорових схем.

Форматування.

- ◎ Використовуйте стандартний відступ в блоках
- ◎ Розташовуйте вертикально скобки відкриття/закриття

```
for (i = 0; i < 100; i++)
```

```
{
```

```
...
```

```
}
```

- ◎ Можна використати також інший стиль:

```
for (i = 0; i < 100; i++){
```

```
...
```

```
}
```

```
Do {
```

```
...
```

```
} while ();
```


ПРОЦЕСИ ТА СИСТЕМИ ПІДТРИМКИ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

- ⦿ Використовуйте один стиль в коді усієї програми.
- ⦿ Використовуйте відступи в логічних конструкціях.

```
If ... Then
```

```
If ... Then
```

```
...
```

```
Else
```

```
End If
```

```
Else
```

```
...
```

```
End If
```

```
If ... Then
```

```
    If ... Then
```

```
        ...
```

```
    Else
```

```
        ...
```

```
    End If
```

```
Else
```

```
...
```

```
End If
```

ПРОЦЕСИ ТА СИСТЕМИ ПІДТРИМКИ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

- ⦿ Визначте максимальну довжину рядка коментаря, для того щоб не використовувати прокрутку потім в презентаціях.
- ⦿ Використовуйте пробіли перед і після більшості операторів, якщо це не вплине на інтерпретацію коду.
- ⦿ Коли код розбитий на декілька рядків ставте оператори в кінці рядка, а не на початку.
- ⦿ Всюди де можливо не розташовуйте більше однієї команду в одному рядку. Виключення – конструкції циклу.
- ⦿ При написанні HTML коду застосовуйте стандартний формат: наприклад великі букви для тегів і маленькі для атрибутів.
- ⦿ При написанні SQL коду застосовуйте верхній регістр для ключових слів і змішаний регістр для власних назв.

ПРОЦЕСИ ТА СИСТЕМИ ПІДТРИМКИ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

- ◎ Розподіляйте код логічно між фізичними файлами.
- ◎ Виділяйте кожен логічний фрагмент (SQL clause) SQL коду в окремий рядок:

```
SELECT Id, FirstName, LastName  
FROM Customers  
WHERE State = 'WA'  
ORDER BY Id
```
- ◎ Розбивайте великі секції коду на менші модулі, які легше сприймати.