Regras e Convenções de Escrita de Código

for project mouse

December 26, 2024

Introdução

Este documento descreve regras e convenções para escrita de código em linguagens **C++**, **CSHARP** e **GDScript**. A adoção de padrões consistentes melhora a legibilidade e manutenção do código. Serão abordados estilos como **PEP8**, **PascalCase** e **snake_case**.

1 Regras Gerais

- Utilize indentação consistente (recomenda-se 4 espaços por nvel (TAB padrão do vscode).
- Evite linhas muito longas (entre 80-120 caracteres).
- Insira comentários para explicar lógica complexa ou decisões importantes.
- Nomeie variáveis e funções de forma descritiva e consistente.
- Separe código em módulos ou classes para organizar funcionalidades.
- Escreva TUDO em português exceto convenções decididas ao longo do projeto.
- Evite sempre o uso de váriaveis globias, principalmente em partes maiores e mais complexas da aplicação

2 C++

C++ utilizaremos convenções variadas. Recomenda-se seguir estas práticas gerais até segunda ordem:

Tipo

- Classes: **PascalCase** (e.g., `class MinhaClasse`).
- Funções: **camel_Case like** (e.g., `void minha_Funcao()`).
- Variáveis: **snake_case** (e.g., `int minha_var`).
- Constantes: **UPPER_SNAKE_CASE** (e.g., `const int MAX_VALUE = 100`).

Exemplo de Código

```cpp
#include <iostream>

class MinhaClasse {
public:
    void print_Mensagem() {
        std::cout << "Hello, World!" << std::endl;
    }
};

int main() {
    MinhaClasse minha_classe;
    minha_classe.print_Mensagem();
    return 0;
}
```

3 CSHARP

CSHARP seguiremos convenções muito próximas das práticas .NET, priorizando **PascalCase** para muitos elementos.

Tipo

- Classes e métodos: **PascalCase** (e.g., `public class MinhaClasse`, `void MeuMetodo()`).
- Variáveis locais e parâmetros: **camelCase** (e.g., `int minhaVar`).
- Constantes: **UPPER: PASCALCase** (e.g., `const int MAXValue = 100`).
- Campos privados: Prefixados com _ (e.g., `private int _meuPrivado`).

Exemplo de Código

```
using System;

namespace MyNamespace {
    public class MinhaClasse {
        private int _meuPrivado;

        public void MeuMetodo() {
            Console.WriteLine("Hello, World!");
        }
    }

    class Programa {
        static void Main(string[] args) {
            MinhaClasse minhaClasse = new MinhaClasse();
            MinhaClasse.MeuMetodo();
        }
    }
}
```

4 GDScript

GDScript, utilizado no Godot Engine, segue o estilo **snake_case** em geral, com algumas variações específicas para scripts.

Tipo

- Funções e variáveis: **snake_case** (e.g., `func minha_funcao():`, `var minha_var`).
- Classes: **PascalCase** (e.g., `class_name MinhaClasse`).
- Constantes: **UPPER_SNAKE_CASE** (e.g., `const MAX_SPEED = 100`).

Exemplo de Código

```
extends Node

class_name MinhaClasse

    const MAX_SPEED = 100

```

```
7       var minha_var = 10
8
9       func _comecar():
10          print("Hello, World!")
11
12      func minha_funcao():
13          return MAX_SPEED * minha_var
```

5 Comentários

Durante todo o código comente o necessário para que qualquer um seja capaz de entender o que foi feito. Siga o seguinte padrão para comentar:

- Funções: Comente acima da declaração da função o que e como ela faz. Explique sua entrada e seu retorno abaixo da declaração da função.
- Classes: Comente acima da declaração da classe para que ela serve. Comente abaixo da declaração da classe todos os seus campos.
- Constantes: Devem ter nomes descritivos o suficiente para que não seja necessário comentá-los, exceto caso esteja utilizando uma constante como flag, portanto deve-se explicar como e porque utiliza essa flag.
- Loops: Deve-se comentar ao lado da declaração do loop o que ele faz e como ele faz
- Sinalização para os outros devs: Deve-se sinalizar caso alguma parte do código não deva ser alterada e porque. Deve-se sinalizar caso alguma parte do código deva ser alterada, como e porque.

Conclusão

Seguir convenções de escrita facilita a colaboração e a leitura do código. Consistência é essencial para projetos profissionais e pessoais. Perceba que seguindo esse padrão de convenções durante o projeto, saberemos em qual linguagem e com o que estamos trabalhando mais facilmente.