**FACULDADE INTEGRADAS DE CAMÕES**

**VINICIUS GUSTAVO DE OLIVEIRA**

**VINICIUS ROCCO FILHO**

Faculdade Integradas Camões (FICA)

Caixa Postal 80060-110-970 – Curitiba – PR – Brasil

Tecnologia em Analise e desenvolvimento de Sistemas –Faculdade Integradas de Camões

Faculdade Integradas de Camões (FICA) –Curitiba, PR – Brazil

{Vinicius [Gustavo}vinispei2017@gmail.com](mailto:Gustavo%7dvinispei2017@gmail.com)

{Vinicius Rocco} vinicius.rocco.filho@hotmail.com

**Resumo.** O uso de boas práticas escrita em C++ é um fator fundamental para o desenvolvimento de softwares que atende requisitos de programa. Mas suas práticas têm sido mostradas por desenvolvedores que usa ele, porém seu uso pode não ser específico à linguagem C. Suas informações sobre normas na linguagem C++ são feitas como um manual mostrando vários tópicos que pode ser usado em seu software. Este artigo tem como objetivo ajudar e mostrar um conjunto de normas de programação em C++, que permita a sua aplicação prática imediata tanto num contexto educacional e de treinamento, quanto num contexto profissional.

**Abstract** The use of good practices written in C ++ is a key factor in the development of software that meets program requirements. But its practices have been shown by developers who use it, but its use may not be specific to C language. Its information on C ++ standards is made as a manual showing various topics that can be used in its software. This article aims to help and show a set of programming rules in C ++ that allow its immediate practical application in an educational and training context, as well as in a professional context.

1. **Guia de Boas Práticas em C++:**

Um guia para normatizar a escrita possibilita a criação de um código padronizado de melhor entendimento por parte dos programadores que o utilizem.

1. **Estilo do Código**

O programador pode ter seu estilo de código que pode acarretar em problemas de legibilidade, por isso é importante sempre revisar para que possa ser o mais visível possível.

**2.1 NOMES DE VARIÁVEIS**

Uma boa pratica é deixar sempre variáveis com letra minúscula, quando você rever ou outro programador ver, vai saber que aquilo é uma variável.

Exemplo:

Bom

string minhaVariavel;

// ou

string minha\_variavel;

Ruim

string Minha\_Varivel209;

// ou

string Minha\_Variavel\_1997;

**2.2 NOMES DE CONSTANTES**

Diferente das variáveis uma boa pratica de uma constante é deixar sempre com letra maiúscula.

Exemplo:

Bom

const double PI = 3.14159;

Ruim

const double pi = 3.14159;

**2.3 NOMES DE FUNÇÕES**

Nome de funções devem começar com a primeira letra minúscula, assim como as variáveis:

Exemplo:

Bom

void minhaFuncao();

Ruim

void MinhaFuncao();

Pior ainda

void Minha\_Funcao();

**2.4 NOMES DE CLASSES**

Quanto a classes é recomendado começar com a primeira letra maiúscula.

Bom

class LinkedList();

Ruim

class linkedList();

**3. Comentários:**

Para comentários utiliza // para comentários separados por linhas, por exemplo:

bool equal( int value1, int value2 )

{

// Compara dois valores e retorna

// verdadeiro se os valores são iguais

if( value1 == value2 )

{

return true;

}

return false;

}

Para arrumar um erro e tirar uma parte do algoritmo usa-se o /\*, por exemplo:

bool equal( int value1, int value2 )

{

/\*

// Compara dois valores e retorna

// verdadeiro se os valores são iguais

if( value1 == value2 )

{

return true;

}

\*/

return false;

}

**4- INDENTAÇÃO**

A indentação é muito importante para um algoritmo mais claro, isso pode mudar de projeto para projeto mas os padrões é 4 espaços, 2 espaços ou 1 tab.

**4.1- NÃO UTILIZE NÚMEROS MÁGICOS**

Não utilize números mágicos como o exemplo abaixo:

Ruim

double calc( double value )

{

return value \* 3.14159;

}

Nesse caso seria melhor usa uma constante:

Bom

const double PI = 3.14159;

double calc( double value )

{

return value \* PI;

}

É bom usar números em alguns casos, quando fazer sentido, por exemplo:

Bom

double calc( double value )

{

return value \* 2;

}

Ruim

#define TWO 2

double calc( double value )

{

return value \* TWO;

}

**4.2- USO DAS CHAVES**

Uma excelente boa pratica é sempre usar as chaves mesmo que exista só uma linha de código e seu programa, o não uso das chaves pode acarretar em erro em seu algoritmo.

Exemplo:

Bom

int sum = 0;

for (int i = 0; i < 15; ++i)

{

++sum;

std::cout << i << std::endl;

}

Ruim

for (int i = 0; i < 15; ++i)

std::cout << i << std::endl;

Erro semântico

int sum = 0;

for (int i = 0; i < 15; ++i)

++sum;

std::cout << i << std::endl;

**4.3- COMPRIMENTO DE LINHAS**

Tente sempre manter a linha em um tamanho médio, caso seja muito extensa a linha do algoritmo vale a pena quebrá-la em múltiplas linhas, exemplo:

Bom

if( (x == 1 && y == 2 && myFunction() == true) ||

(x == 0 && y == 0 && myFunction() == false) )

{

}

Ruim

if( (x == 1 && y == 2 && myFunction() == true) || (x == 0 && y == 0 && myFunction() == false) )

{

}

**4.4- ASPAS EM ARQUIVOS LOCAIS**

Utilize aspas duplas ("") para incluir arquivos locais.

Bom

#include <string>

#include "MyHeader.hpp"

Ruim

#include <string>

#include <MyHeader.hpp>

**4.5- CONSTANTES**

Utilize os “const” sempre que possível, ele ajuda o compilador avisando que é uma variável imutável. Isto otimiza o código e ajuda o programador a saber se uma função pode dar erro e também previne o compilador não compilar dados desnecessariamente

Bom

class MyClass

{

public:

void do\_something(const int i);

void do\_something(const std::string &str);

};

Ruim

class MyClass

{

public:

void do\_something(int i);

void do\_something(std::string str);

};

Se necessário quebrar uma linha:

* Quebre antes de um commando
* Quebre depois de um operador
* Alinhar a nova linha com o inicio da expressão, no mesmo nível, da linha anterior
* Se as regras acima levar a um código confuso ou muito próximo da margem direita deve-se indentar 8 espaços.

As linhas quebradas para na declaração, das condições do if deve utilizar a regra do 8 espaços para indentação. Exemplo:

if ((condition1 && condition2)

|| (condition3 && condition4) || (condition5 && condition6)) { dosomething;

}

**LINHAS EM BRANCO**

Deve-se deixar uma linha em branco entre:

* Métodos
* Variáveis locais de um método e sua primeira declaração  Depois de um bloco ou uma linha de comentário.

Deve-se deixar 2 linhas em branco entre:

* Definições das classes e interfaces  Entre secções do source file.

**6.** ESPAÇOS EM BRANCO

Deve ser usado:

* Entre uma palavra chave seguido de parênteses
* Em todos os operadores envolvendo operações com dois ou mais termos, o que não se aplica as operações envolvendo um só termo. Exemplo: incremento, decremento
* Entre as declarações em um for
* Entre os operadores de coerção (casts) e o termo que esta sendo modelado.

**ESTRUTURAS DE CONTROLE:**

Declaração do if, if-else, if else-if else:

Tem a seguinte forma:

if (condição)

{

Declaração;

}

if (condição)

{

Declaração;

} else

{

Declaração;

}

if (condição)

{

Declaração;

} else if (condição)

{

Declaração;

} else {

Declaração;

}

**Declaração do for** Tem a seguinte forma: for (inicio; condição; incremento/decremento)

{

Declaração;

}

**Caso o bloco do for seja vazio tem a seguinte forma**:

for (inicio; condição; incremento/decremento);

**Declaração do while:**

Tem a seguinte forma:

while (condição)

{

Declaração;

}

Caso o bloco do while seja vazio tem a seguinte forma:

while (condição);

**Declaração do do-while:**

Tem a seguinte forma:

Do

{

Declaração;

} while (condição);

**Declaração do switch:**

Tem a seguinte forma:

switch (condição)

{ case a: Declaração; break; case b: Declaração; break;

}

**Declaração try-catch:**

Tem a seguinte forma:

try {

Declaração;

} catch (Exceptionclass e)

{

Declaração;

}

Pode-se utilizar também o finally:

try {

Declaração; } catch (Exceptionclass e) {

Declaração;

} finally

{

Declaração;

}

**7. References**

**C++BestPractices**: <https://www.gitbook.com/book/lefticus/cpp-best-practices/details>

**Google C++ Style Guide**: <https://google.github.io/styleguide/cppguide.html>

**10 most voted C++ best practices**: <http://codergears.com/Blog/?p=1957>

# ****Guia Rápido de Boas Práticas em C++:**** https://github.com/kelvins/Boas-Praticas-Cplusplus#estilo-de-c%C3%B3digo