Sistema de Notação de Nomes – Spark – v2.0 - Jul/2019

1. Variáveis e Constantes

ETI*VariavelNome* / **ETI***CONSTANTE_NOME*

Escopo

```
g  – global
f  – arquivo (static fora de função)
m  – membro de objeto de classe (não use em struct/union)
a  – argumento de Função
t  – argumento de Template
s  – estático (dentro de função ou classe)
▒  – outros (variável no corpo de uma função, membro de struct)
```

Tipo

```
b  – booleano
c  – char
d  – double
l  – long double
e  – enum
f  – float
in – inteiro signed [1]
un – inteiro unsigned [1]
s  – string (std::string, char*, etc)
o  – objeto (instância) de class/struct/union
h  – função/método (apenas ponteiros)
k  – complexo (std::complex<tipo> kFrequencia)
t  – template (Tipo_t tInicio = 1 )
v  – void
```

Indireção

```
v  – vetor (1 dimensão ou std::vector<>) [2]
a  – matriz (2+ dimensões ou std::vector<std::vector<> >) [2]
q  – container (std::map<>, std::set<> e std::list<>, exceções: std::vector<>)
p  – ponteiro[1]
r  – referência
▒  – sem indireção
```

2. Funções e Métodos

ERI*NomeDaFuncao()*

Escopo

```
m  – método de classe
v  – método virtual de classe
s  – método estático de classe
f  – função em escopo de arquivo (static)
▒  – função em escopo global
```

Retorno da função (tipo)

Seguir Tipos de **1. Variáveis e Constantes**

Indireção

Seguir Indireção de **1. Variáveis e Constantes**

3. Tipos

*NomeDoTipo*_**N**

Natureza

```
e  – enum
s  – struct
c  – classe sem herança (não é herdeira de outra classe pai)
d  – classe derivada
u  – union
n  – namespace
t  – template
f  – ponteiro para função (usando typedef)
x  – outros (a definir futuramente, por demanda)
```

Ex:

- *enum ErroArquivo**_e** {}* - um tipo enum
- *class Comunica**_b** {}* - classe base para ser herdada
- *int16_t **mi2v**Fast[10]* - vetor de inteiros, membro de classe
- *char* **gs**Mens* - string de escopo global
- *double** **ada**V1* - argumento de função, array de doubles
- *Tipo**_e**& **er**Erro* - referência para variavel enum em escopo de função
- *template<int8_t **ti1**SIZE>* - template com argumento de constante inteira
- *uint16_t **au2**Num* - argumento de função para inteiro de 16 bits.
- *double (***pd**Func)(void** **avp**Ptr)* - Ponteiro p/ função com argumento void**, retornando double

Dicas:

- *Escrever nome de constantes com maiúsculas: const static double **gd**PI = 3.141d;*
- *Usar apenas tipos inteiros de <inttypes.h>: uint64_t **mu8**HoraAtiva;*
- *Constantes enum iniciam com **E_***