### CO-DING

* Classnames Pascal-Case → ClassName
* Non-Member-Variables Camelcase → myVariable
* Non-Static Member-Variables → \_variableName
* Static Member Variable → classicVariableName
* Non-Member static Variable → gVariable
* Global Constants, enums upper case → MAX\_ITERATIONS
* functions, methods camelCase → myMethod(void)
* Generic variables should have the same name as their type → void setMethod(Topic topic)
* setter/getter camelCase → setName(name) / NameType getName(void)
* The compute prefix should be used in methods where something (complex) is computed

→ controller.computeDistance()

* preferred: iterWhat → iterRow, iterCol, + i,j,k
* boolean prefixes: is, has, can, should → isFound, hasErrors
* Complement names must be used for complement operations → init/cleanup, create/destroy
* Indentations are defined as two white-spaces (“ “)
* The body of a function should not exceed one page
* Standard init: NULL for pointers, and '\0' for chars.
* Logical units within a block should be separated by one blank line.
* Assignment in conditionals must be avoided

// GOOD

File \*fileHandle = open(fileName, "w");

if (!fileHandle) {

:

}

// BAD

if (!(fileHandle = open(fileName, "w"))) {

:

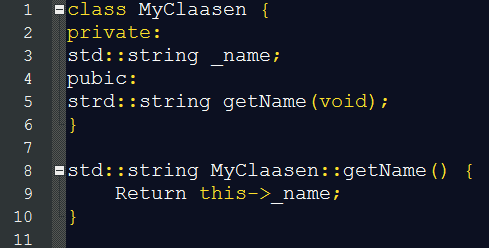
}

### 

### FILETS

Filenames Pascalcase → MyClass.cpp

If only one class is defined inside a cpp file, the corresponding cpp and hpp filename should match the classname

Else use a name that describes all contained classes

Simple methods→ inline code → Value value() const { return mValue; }

### Include Filets and Include Statementos

Header files must contain an include guard

#ifndef CLASS\_NAME\_HPP

#define CLASS\_NAME\_HPP

:

#endif

Variables should get initialized when declared → GOOD: int i = 123; BAD: int i; i = 123;

For further references see++: <https://github.com/omichel/webots/wiki/CPP-Coding-Style>