AOCL Crypto Coding Style

Coding Style Reference Document

Prem Mallappa pmallapp@amd.com

Contents

List of Figures

Part I

Naming

Allowed Cases:

- lower_case
- UPPER_CASE
- camelBack
- CamelCase
- camel_Snake_Back
- aNy_CasE
- Camel_Snake_Case

Chapter 1

Abstract Class

- AbstractClassCase - CamelCase
- AbstractClassPrefix - Abstract
- AbstractClassSuffix - NA

```
class model {
public:
  model();
};
```

After:

```
class AbstractModel {
public:
    AbstractModel();
}
```

Chapter 2

Class

- ClassCase - CamelCase
- ClassPrefix - None
- ClassSuffix - None

```
class model_blah {
public:
  model();
};
```

After:

```
class ModelBlah {
public:
    ModelBlah();
}
```

Chapter 3

Class Constants

- ClassConstantCase - CamelCase
- ClassConstantPrefix - c
- ClassConstantSuffix - NA

```
class FOO {
public:
  static const int CLASS_CONSTANT;
};
```

After:

```
class FOO {
public:
  static const int cClassConstant;
};
```

Chapter 4

Class Member

- ClassMemberCase - lower_case
- ClassMemberPrefix - m_
- ClassMemberSuffix - NA

```
class FOO {
public:
  static int    CLASS_MEMBER;
  int           another_class_member;
};
```

After:

```
class FOO {
public:
  static int m_class_member;
  int        m_another_class_member;
};
```

Chapter 5

Class Method (member function)

Class methods should use ‘camelBack’ case with no prefix or suffix.

- ClassMethodCase - camelBack
- ClassMethodPrefix - NA
- ClassMethodSuffix - NA

```
class FOO {
public:
  int CLASS_MEMBER();
};
```

After:

```
class Foo {
public:
  int classMember();
};
```

Chapter 6

Constants

- ConstantCase - CamelCase
- ConstantPrefix - c
- ConstantSuffix - NA

```
void function() { unsigned const MyConst_array[] = {1, 2, 3}; }
```

After:

```
void function() { unsigned const cMyconstArray[] = {1, 2, 3}; }
```

Chapter 7

Constant Members of a class

- ConstantMemberCase - CamelCase
- ConstantMemberPrefix - c
- ConstantMemberSuffix - NA

```
class Foo {
  char const MY_ConstMember_string[4] = "123";
}
```

After:

```
class Foo {
  char const cMyConstmemberString[4] = "123";
}
```

Chapter 8

Constant Parameter

- ConstantParameterCase - CamelCase
- ConstantParameterPrefix - c
- ConstantParameterSuffix - NA

```
void GLOBAL_FUNCTION(int PARAMETER_1, int const CONST_parameter);
```

After:

```
void GLOBAL_FUNCTION(int PARAMETER_1, int const cConstParameter);
```

Chapter 9

Constant Pointer Parameter

- ConstantPointerParameterCase - CamelCase
- ConstantPointerParameterPrefix - pc
- ConstantPointerParameterSuffix - NA

```
void GLOBAL_FUNCTION(int const *CONST_parameter);
```

After:

```
void GlobalFunction(int const *pcConstParameter);
```

Chapter 10

Constexpr Function

Follow same naming conventions for GlobalFunctionCase and no prefix or suffix is needed for Global constexpr function. and MethodCase with no prefix or suffix for Method constexpr functions

- ConstexprFunctionCase - GlobalFunctionCase
- ConstexprFunctionPrefix - NA
- ConstexprFunctionSuffix - NA

```
constexpr int CE_function() { return 3; }
```

After:

```
constexpr int Function() { return 3; }
```

Chapter 11

Constexpr Method

Follow same naming conventions of MethodCase with no prefix or suffix for Method constexpr functions.

- ConstexprMethodCase - MethodCase (camelBack)
- ConstexprMethodPrefix - NA
- ConstexprMethodSuffix - NA

```
class Foo {
public:
  constexpr int CONST_expr_Method() { return 2; }
}
```

After:

```
class Foo {
public:
  constexpr int constExprMethod() { return 2; }
}
```

Chapter 12

Constexpr Variable

- ConstexprVariableCase
- ConstexprVariablePrefix
- ConstexprVariableSuffix

```
constexpr int ConstExpr_variable = MyConstant;
```

After:

```
constexpr int pre_constexpr_variable_post = MyConstant;
```

Chapter 13

Enumerations

- EnumCase - CamelCase
- EnumPrefix - NA
- EnumSuffix - NA

```
enum FOO { One, Two, Three };
     ^^^
```

After:

```
enum Foo { One, Two, Three };
     ^^^
```

Chapter 14

Enumeration Constant

Follow Enumeration style.

- EnumConstantCase - CamelCase
- EnumConstantPrefix - e
- EnumConstantSuffix - NA

```
enum FOO { One, Two, Three };
```

After:

```
enum Foo { eOne, eTwo, eThree };
          ^^    ^^    ^^
```

Chapter 15

Function

Functions are global entities (others are Methods). Both Static and non-static global functions use the same style.

- FunctionCase - CamelCase
- FunctionPrefix - NA
- FunctionSuffix - NA

```
char MY_Function_string();
```

After:

```
char MyFunctionString();
```

Chapter 16

Global Constant

- GlobalConstantCase - CamelCase
- GlobalConstantPrefix - gc
- GlobalConstantSuffix - NA

```
unsigned const MyConstGlobal_array[] = {1, 2, 3};
```

After:

```
unsigned const gcMyConstGlobalArray[] = {1, 2, 3};
```

Chapter 17

Global Constant Pointer

- GlobalConstantPointerCase - gp
- GlobalConstantPointerPrefix - NA
- GlobalConstantPointerSuffix - NA

```
int *const MyConstantGlobalPointer = nullptr;
```

After:

```
int *const gpcMyConstantGlobalPointer = nullptr;
```

Chapter 18

Global Constant Function

- GlobalFunctionCase - CamelCase
- GlobalFunctionPrefix - NA
- GlobalFunctionSuffix - NA

```
void GLOBAL_FUNCTION(int PARAMETER_1, int const CONST_parameter);
```

After:

```
void GlobalFunction(int PARAMETER_1, int const CONST_parameter);
```

Chapter 19

Global Pointer

- GlobalPointerCase
- GlobalPointerPrefix
- GlobalPointerSuffix

```
int *GLOBAL3;
```

After:

```
int *pre_global3_post;
```

Chapter 20

Global Variable

- GlobalVariableCase
- GlobalVariablePrefix
- GlobalVariableSuffix

```
int GLOBAL3;
```

After:

```
int pre_global3_post;
```

Chapter 21

Inline Namespace

- InlineNamespaceCase
- InlineNamespacePrefix
- InlineNamespaceSuffix

```
namespace FOO_NS {
inline namespace InlineNamespace {
...
}
} // namespace FOO_NS
```

After:

```
namespace FOO_NS {
inline namespace pre_inlinenamespace_post {
...
}
} // namespace FOO_NS
```

Chapter 22

Local Constant

- LocalConstantCase - lower_case
- LocalConstantPrefix - c_
- LocalConstantSuffix - NA

```
void foo() { int const local_Constant = 3; }
```

After:

```
void foo() { int const c_local_constant = 3; }
```

Chapter 23

Local Constant

- LocalConstantPointerCase - lower_case
- LocalConstantPointerPrefix - p_
- LocalConstantPointerSuffix - NA

```
void foo() { int const *local_var = 3; }
```

After:

```
void foo() { int const *cp_local_var = 3; }
```

Chapter 24

Local Pointer

- LocalPointerCase
- LocalPointerPrefix
- LocalPointerSuffix

```
void foo() { int *local_Variable; }
```

After:

```
void foo() { int *p_local_var; }
```

Chapter 25

Local Variable

- LocalVariableCase
- LocalVariablePrefix
- LocalVariableSuffix

```
void foo() { int local_Variable; }
```

After:

```
void foo() { int local_var; }
```

Chapter 26

Member Variable

- MemberCase
- MemberPrefix
- MemberSuffix

After:

Chapter 27

Methods

- MethodCase
- MethodPrefix
- MethodSuffix

After:

Chapter 28

Namespace

- NamespaceCase
- NamespacePrefix
- NamespaceSuffix

After:

Chapter 29

Function Parameters

- ParameterCase
- ParameterPrefix
- ParameterSuffix

After:

Chapter 30

Parameter Packs

- ParameterPackCase
- ParameterPackPrefix
- ParameterPackSuffix

After:

Chapter 31

Pointer Parameter

- PointerParameterCase
- PointerParameterPrefix
- PointerParameterSuffix

After:

Chapter 32

Private members

- PrivateMemberCase
- PrivateMemberPrefix
- PrivateMemberSuffix

After:

Chapter 33

Private Methods

- PrivateMethodCase
- PrivateMethodPrefix
- PrivateMethodSuffix

After:

Chapter 34

Protected Members

- ProtectedMemberCase
- ProtectedMemberPrefix
- ProtectedMemberSuffix

After:

Chapter 35

Protected Methods

- ProtectedMethodCase
- ProtectedMethodPrefix
- ProtectedMethodSuffix

After:

Chapter 36

Public Members

- PublicMemberCase
- PublicMemberPrefix
- PublicMemberSuffix

After:

Chapter 37

Public Methods

- PublicMethodCase
- PublicMethodPrefix
- PublicMethodSuffix

After:

Chapter 38

Static Constants

- StaticConstantCase
- StaticConstantPrefix
- StaticConstantSuffix

After:

Chapter 39

Static Variables

- StaticVariableCase
- StaticVariablePrefix
- StaticVariableSuffix

After:

Chapter 40

Structures

- StructCase
- StructPrefix
- StructSuffix

After:

Chapter 41

Template Parameters

- TemplateParameterCase
- TemplateParameterPrefix
- TemplateParameterSuffix

After:

Chapter 42

Template Template Parameters

- TemplateTemplateParameterCase
- TemplateTemplateParameterPrefix
- TemplateTemplateParameterSuffix

After:

Chapter 43

Type Aliases

- TypeAliasCase
- TypeAliasPrefix
- TypeAliasSuffix

After:

Chapter 44

Typedefs

- TypedefCase
- TypedefPrefix
- TypedefSuffix

After:

Chapter 45

Type Template Parameters

- TypeTemplateParameterCase
- TypeTemplateParameterPrefix
- TypeTemplateParameterSuffix

After:

Chapter 46

Union

- UnionCase
- UnionPrefix
- UnionSuffix

After:

Chapter 47

Value Template

- ValueTemplateParameterCase
- ValueTemplateParameterPrefix
- ValueTemplateParameterSuffix

After:

Chapter 48

Variable

- VariableCase
- VariablePrefix
- VariableSuffix

After:

Chapter 49

Virtual Method

- VirtualMethodCase
- VirtualMethodPrefix
- VirtualMethodSuffix

After:

Part II

Example .clang-tidy file

```
Checks: '-*,readability-identifier-naming'
CheckOptions:
  - { key: readability-identifier-naming.NamespaceCase,       value: lower_case }
  - { key: readability-identifier-naming.ClassCase,           value: CamelCase  }
  - { key: readability-identifier-naming.PrivateMemberPrefix, value: m_          }
  - { key: readability-identifier-naming.StructCase,          value: CamelCase  }
  - { key: readability-identifier-naming.FunctionCase,        value: lower_case }
  - { key: readability-identifier-naming.VariableCase,        value: lower_case }
  - { key: readability-identifier-naming.GlobalConstantCase,  value: UPPER_CASE }
```