

C++

- [!\[\]\(1207edb9a08751d3d55970560645ed23_img.jpg\)](#)
- [C & C++ !\[\]\(d7a34a706cfa4ef37c62a369101e1b36_img.jpg\)](#)



vscode

```
{
// Place your snippets for cpp here. Each snippet is defined under a snippet name and has a
prefix, body and
// description. The prefix is what is used to trigger the snippet and the body will be
expanded and inserted. Possible variables are:
// $1, $2 for tab stops, $0 for the final cursor position, and ${1:label}, ${2:another} for
placeholders. Placeholders with the
// same ids are connected.
// Example:
"copyright info":{
    "prefix": "//file",
    "body": [
        "//-----",
    ,
        "/// Copyright(C) 2020-2022, netflt.com",
        "/// All rights reserved.",
        "/// ",
        "/// name: $TM_FILENAME",
        "/// author: Danny.Zhu"
        "/// date: $CURRENT_YEAR-$CURRENT_MONTH-$CURRENT_DATE
$CURRENT_HOUR:$CURRENT_MINUTE:$CURRENT_SECOND",
        "///-----"
    ],
    "description": "add copyright information"
},
"simple comment":{
    "prefix": "//",
    "body": [
        "///<! ${1}"
    ],

```

```

        "description": "add simple comment"
    },
    "block comment":{
        "prefix": "///<",
        "body": [
            "/*",
            "/*<! ${1}",
            "*/"
        ],
        "description": "add block comments"
    },
    "TODO": {
        "prefix": "//td",
        "body": [
            "/*<! TODO (Danny.Zhu): $1"
        ],
        "description": "Log output to console"
    }
}

```

clang-format



.clang-format

```

---
Language:      Cpp
# BasedOnStyle: Chromium
SortIncludes:  true
AccessModifierOffset: -4
DerivePointerAlignment: false
PointerAlignment: Left
ConstructorInitializerIndentWidth: 4
AlignEscapedNewlines: Left

```

AlignAfterOpenBracket: Align
AlignConsecutiveMacros: Consecutive
AlignConsecutiveAssignments: None
AlignConsecutiveBitFields: Consecutive
AlignConsecutiveDeclarations: false
AlignOperands: Align
AlignTrailingComments: true
AllowShortBlocksOnASingleLine: Never
AllowShortFunctionsOnASingleLine: Empty
AllowAllArgumentsOnNextLine: true
AllowAllConstructorInitializersOnNextLine: false
AllowAllParametersOfDeclarationOnNextLine: false
AllowShortEnumsOnASingleLine: false
AllowShortLambdasOnASingleLine: Empty
AllowShortIfStatementsOnASingleLine: Never
AllowShortCaseLabelsOnASingleLine: false
AllowShortLoopsOnASingleLine: false
AlwaysBreakTemplateDeclarations: true
BinPackArguments: false
AlwaysBreakBeforeMultilineStrings: false
BreakBeforeBinaryOperators: NonAssignment
BreakBeforeTernaryOperators: true
BreakConstructorInitializersBeforeComma: false
BinPackParameters: false
ColumnLimit: 96
ConstructorInitializerAllOnOneLineOrOnePerLine: false
DerivePointerBinding: false
ExperimentalAutoDetectBinPacking: false
IndentCaseLabels: false
MaxEmptyLinesToKeep: 1
NamespaceIndentation: None
ObjCSpaceBeforeProtocolList: true
PenaltyBreakBeforeFirstCallParameter: 1
PenaltyBreakComment: 300
PenaltyBreakString: 1000
PenaltyBreakFirstLessLess: 120
PenaltyExcessCharacter: 1000000
PenaltyReturnTypeOnItsOwnLine: 1000
AlwaysBreakAfterDefinitionReturnType: None

AlwaysBreakAfterReturnType: None
PointerBindsToType: true
Cpp11BracedListStyle: false
Standard: Auto
IndentWidth: 4
TabWidth: 4
UseTab: Never
BreakBeforeBraces: Custom
AttributeMacros:
- __capability
BraceWrapping:
AfterCaseLabel: false
AfterClass: true
AfterControlStatement: false
AfterEnum: false
AfterFunction: true
AfterNamespace: true
AfterObjCDeclaration: true
AfterStruct: false
AfterUnion: false
AfterExternBlock: true
BeforeCatch: true
BeforeElse: true
BeforeLambdaBody: false
BeforeWhile: false
IndentBraces: false
SplitEmptyFunction: false
SplitEmptyRecord: false
SplitEmptyNamespace: false
SpacesInParentheses: false
SpacesInAngles: false
BreakBeforeConceptDeclarations: true
BreakBeforeInheritanceComma: false
BreakInheritanceList: BeforeComma
BreakConstructorInitializers: BeforeComma
BreakAfterJavaFieldAnnotations: false
BreakStringLiterals: true
CommentPragmas: '^ IWYU pragma:'
CompactNamespaces: false

ContinuationIndentWidth: 4
DeriveLineEnding: true
DisableFormat: false
EmptyLineBeforeAccessModifier: LogicalBlock
FixNamespaceComments: true
ForEachMacros:
 - foreach
 - Q_FOREACH
 - BOOST_FOREACH
StatementAttributeLikeMacros:
 - Q_EMIT
IncludeBlocks: Regroup
IncludeCategories:
 - Regex: '^<ext/.*\\.h>'
 Priority: 2
 SortPriority: 0
 CaseSensitive: false
 - Regex: '^<.*\\.h>'
 Priority: 1
 SortPriority: 0
 CaseSensitive: false
 - Regex: '^<.*'
 Priority: 2
 SortPriority: 0
 CaseSensitive: false
 - Regex: ',.*'
 Priority: 3
 SortPriority: 0
 CaseSensitive: false
IncludeIsMainRegex: '([-_](test|unittest))?\$'
IncludeIsMainSourceRegex: ''
IndentCaseBlocks: false
IndentGotoLabels: true
IndentPPDirectives: BeforeHash
IndentExternBlock: AfterExternBlock
IndentRequires: false
IndentWrappedFunctionNames: true
InsertTrailingCommas: None
JavaScriptQuotes: Leave

```
JavaScriptWrapImports: true
KeepEmptyLinesAtTheStartOfBlocks: true
MacroBlockBegin: ''
MacroBlockEnd: ''
ObjCBinPackProtocolList: Auto
ObjCBlockIndentWidth: 2
ObjCBreakBeforeNestedBlockParam: true
ObjCSpaceAfterProperty: false
PenaltyBreakAssignment: 2
PenaltyBreakTemplateDeclaration: 10
PenaltyIndentedWhitespace: 0
RawStringFormats:
  - Language:      Cpp
    Delimiters:
      - cc
      - CC
      - cpp
      - Cpp
      - CPP
      - 'c++'
      - 'C++'
    CanonicalDelimiter: ''
    BasedOnStyle:    google
  - Language:      TextProto
    Delimiters:
      - pb
      - PB
      - proto
      - PROTO
    EnclosingFunctions:
      - EqualsProto
      - EquivToProto
      - PARSE_PARTIAL_TEXT_PROTO
      - PARSE_TEST_PROTO
      - PARSE_TEXT_PROTO
      - ParseTextOrDie
      - ParseTextProtoOrDie
      - ParseTestProto
      - ParsePartialTestProto
```

```
CanonicalDelimiter: ''
BasedOnStyle:      google
ReflowComments:   true
SortJavaStaticImport: Before
SortUsingDeclarations: true
SpaceAfterCStyleCast: false
SpaceAfterLogicalNot: false
SpaceAfterTemplateKeyword: true
SpaceBeforeAssignmentOperators: true
SpaceBeforeCaseColon: false
SpaceBeforeCpp11BracedList: true
SpaceBeforeCtorInitializerColon: true
SpaceBeforeInheritanceColon: true
SpaceBeforeParens: ControlStatements
SpaceAroundPointerQualifiers: Default
SpaceBeforeRangeBasedForLoopColon: true
SpaceInEmptyBlock: false
SpaceInEmptyParentheses: false
SpacesBeforeTrailingComments: 1
SpacesInConditionalStatement: false
SpacesInContainerLiterals: false
SpacesInCStyleCastParentheses: false
SpacesInSquareBrackets: false
SpaceBeforeSquareBrackets: false
BitFieldColonSpacing: Both
StatementMacros:
  - Q_UNUSED
  - QT_REQUIRE_VERSION
UseCRLF:          false
WhitespaceSensitiveMacros:
  - STRINGIZE
  - PP_STRINGIZE
  - BOOST_PP_STRINGIZE
  - NS_SWIFT_NAME
  - CF_SWIFT_NAME
...
```

clang-tidy

.clang-tidy

```
---
# Configure clang-tidy for this project.

# Here is an explanation for why some of the checks are disabled:
#
# -google-readability-namespace-comments: the *_CLIENT_NS is a macro, and
#     clang-tidy fails to match it against the initial value.
#
# -modernize-use-trailing-return-type: clang-tidy recommends using
#     `auto Foo() -> std::string { return ...; }`, we think the code is less
#     readable in this form.
#
# -modernize-return-braced-init-list: We think removing typenames and using
#     only braced-init can hurt readability.
#
# -modernize-avoid-c-arrays: We only use C arrays when they seem to be the
#     right tool for the job, such as `char foo[] = "hello"`. In these cases,
#     avoiding C arrays often makes the code less readable, and std::array is
#     not a drop-in replacement because it doesn't deduce the size.
#
# -performance-move-const-arg: This warning requires the developer to
#     know/care more about the implementation details of types/functions than
#     should be necessary. For example, `A a; F(std::move(a));` will trigger a
#     warning IFF `A` is a trivial type (and therefore the move is
#     meaningless). It would also warn if `F` accepts by `const&`, which is
#     another detail that the caller need not care about.
#
# -readability-redundant-declaration: A friend declaration inside a class
```

```

# counts as a declaration, so if we also declare that friend outside the
# class in order to document it as part of the public API, that will
# trigger a redundant declaration warning from this check.
#
# -readability-function-cognitive-complexity: too many false positives with
# clang-tidy-12. We need to disable this check in macros, and that setting
# only appears in clang-tidy-13.
#
# -bugprone-narrowing-conversions: too many false positives around
# `std::size_t` vs. `*::difference_type`.
#
# -bugprone-easily-swappable-parameters: too many false positives.
#
# -bugprone-implicit-widening-of-multiplication-result: too many false positives.
# Almost any expression of the form `2 * variable` or `long x = a_int * b_int;`
# generates an error.
#
# -bugprone-unchecked-optional-access: too many false positives in tests.
# Despite what the documentation says, this warning appears after
# `ASSERT_TRUE(variable)` or `ASSERT_TRUE(variable.has_value())`.
#

```

Checks: >

```

-*,
abseil-*,
bugprone-*,
google-*,
misc-*,
modernize-*,
performance-*,
portability-*,
readability-*,
-google-readability-braces-around-statements,
-google-readability-namespace-comments,
-google-runtime-references,
-misc-non-private-member-variables-in-classes,
-misc-const-correctness,
-misc-use-anonymous-namespace,
-modernize-return-braced-init-list,
-modernize-use-trailing-return-type,
-modernize-use-nodiscard,

```

```
-modernize-avoid-c-arrays,  
-performance-move-const-arg,  
-readability-braces-around-statements,  
-readability-identifier-length,  
-readability-magic-numbers,  
-readability-named-parameter,  
-readability-redundant-declaration,  
-readability-function-cognitive-complexity,  
-readability-convert-member-functions-to-static,  
-readability-qualified-auto,  
-readability-implicit-bool-conversion,  
-bugprone-narrowing-conversions,  
-bugprone-easily-swappable-parameters,  
-bugprone-implicit-widening-of-multiplication-result,  
-bugprone-unchecked-optional-access,  
-bugprone-branch-clone
```

```
# Turn all the warnings from the checks above into errors.
```

```
WarningsAsErrors: ""
```

```
CheckOptions:
```

```
- { key: readability-identifier-naming.ClassCase,          value: CamelCase }  
- { key: readability-identifier-naming.StructCase,         value: CamelCase }  
- { key: readability-identifier-naming.TemplateParameterCase, value: CamelCase }  
- { key: readability-identifier-naming.FunctionCase,       value: camel_back }  
- { key: readability-identifier-naming.PrivateMemberPrefix, value: _          }  
- { key: readability-identifier-naming.ProtectedMemberPrefix, value: _          }  
- { key: readability-identifier-naming.EnumConstantCase,   value: CamelCase }  
- { key: readability-identifier-naming.EnumConstantPrefix, value: k          }  
- { key: readability-identifier-naming.ConstexprVariableCase, value: CamelCase }  
- { key: readability-identifier-naming.ConstexprVariablePrefix, value: k          }  
- { key: readability-identifier-naming.GlobalConstantCase, value: CamelCase }  
- { key: readability-identifier-naming.GlobalConstantPrefix, value: k          }  
- { key: readability-identifier-naming.MemberConstantCase, value: CamelCase }  
- { key: readability-identifier-naming.MemberConstantPrefix, value: k          }  
- { key: readability-identifier-naming.StaticConstantCase, value: CamelCase }  
- { key: readability-identifier-naming.StaticConstantPrefix, value: k          }  
- { key: readability-implicit-bool-conversion.AllowIntegerConditions, value: 1 }  
- { key: readability-implicit-bool-conversion.AllowPointerConditions, value: 1 }  
- { key: readability-function-cognitive-complexity.IgnoreMacros, value: 1 }
```


C & C++

1.

--	--

Google C++ Style Guide C++
 ,
 ;
 , C++
 ;
 : shjborage@gmail.com

2.

 /  C++ 

2.1.

--	--	--	--	--	--

□□□□ C++ □□□□□□□□□□□□□□□□ Object-Oriented C++ □□□□□□

2.1.1. struct vs class

```

[ ] struct [ ] ( [ ] [ ] /
[ ] , initialize() , reset() , validate() ) [ ] [ ] class
[ ]

```

Diagram illustrating memory layout for C++ and C:

- C++ side:**
 - `struct` and `class` labels point to a single long array of 16 slots.
- C side:**
 - `struct` label points to a 4-slot array.
 - `C` label points to an 8-slot array.

```
// 座標構造体宣言
struct Coordinate {
    int x;
    int y;
    int z;
};
```

```
// 猫のクラス
class Cat {
public:
    void meow();
private:
    ...
};
```

2.1.2.

```
.[ ] [ ]  
[ ] [ ] DISALLOW_COPY_AND_ASSIGN [ ]  
[ ] ( ) , [ ] init() [ ]  
  
[ ]  
[ ]  
[ ]  
[ ]  
[ ]  
  
[ ]  
[ ]  
[ ] C++  
[ ]  
[ ]  
[ ]
```

```
class Person {
public:
    // 强制使用explicit
    explicit Person(const std::string& name) : _name(name) {} // 强制使用explicit
private:
    std::string _name;
};

DISALLOW_COPY_AND_ASSIGN(Person)

#define DISALLOW_COPY_AND_ASSIGN(ClassName) \
    ClassName(const ClassName&); \
    ClassName& operator=(const ClassName&)

class Foo {
public:
    explicit Foo(int f);
};
```

11

11

11

11

11

11

11

11

11

11

11


```
std::string _name;
};
```

string 类型 C 风格字符串

```
class String {
public:
    String(const char* cs);           // 构造函数

private:
    ...
};
```

□

“ More Effective C++, Item 05: Be wary of user-defined conversion functions

2.1.5. 类型转换

·[] 类型转换

□

类型转换

copy constructor MyClass::MyClass(const MyClass&) 类型转换
/

□

类型转换

类型转换

□

类型转换

```
class Point {
public:
    Point(int x, int y) : _x(x), _y(y) {}
    Point(const Point& other) : // 类型转换
        _x(other._x), _y(other._y) {}
private:
    int _x;
    int _y;
}
```

RAII (RAII)

```
class File {
public:
    explicit File(const char* file_name) :
        _fp(NULL) {
        ... // open file, and set _fp
    }
    ~File() {
        fclose(_fp);
    }
private:
    FILE *_fp;
    DISALLOW_COPY_AND_ASSIGN(File);
};
```

Effective C++, item 06: Explicitly disallow the use of compiler-generated functions you do not want Effective C++, item 14: Think carefully about copying behavior in resource-managing classes

2.1.6. RAII

[]

assignment operator= MyClass& MyClass::operator=(const MyClass&)

```
class Point {
public:
    Point(int x, int y) : _x(x), _y(y) {}
```

[illegible]

--	--

2.1.7.

delete pBase;

[illegible]

```
class Connection {
public:
    void close();    // 閉鎖された接続を解放する
    ~Connection() {
        if (_connection_state == CONNECTED) {
            close(); // 閉鎖された接続を解放する
        }
    }
};
```

“ Effective C++, item 07: Declare destructors virtual in polymorphic base classes
Effective C++, item 08: Prevent exceptions from leaving destructors More
Effective C++, item 11: Prevent exceptions from leaving destructors

```
.[ ] [ ] " " [ ] .[ ]  
[ ] .[ ] [RULE014] [ ]
```

```
class Chinese {
public:
    explicit Chinese(const std::string& name) : _name(name) {}

    // 析构函数
    virtual ~Chinese() {}
}
```

```

// 初始化成员函数
void say(const std::string& word);

// 成员函数
virtual void greet() { say("hello"); }

private:
    std::string _name;
};

// 初始化成员函数

class Cantonese : public Chinese {
public:
    explicit Cantonese(const std::string& name) : Chinese(name) {}
    virtual ~Cantonese() {}

    // 成员函数
    virtual void greet() { say("hello?"); }
};

```

□□

“Effective C++, Item 36: Never redefine an inherited non-virtual function
 Effective C++, Item 32: Make sure public inheritance models “is-a”
 Effective C++, Item 38: Model “has-a” or “is-implemented-in-terms-of” through
 composition Effective C++, Item 39: Use private inheritance judiciously
 C++ Coding Standards, § 37
 C++ Coding Standards, § 34

2.1.9. 成员函数

·[] 成员函数

□□

成员函数
 成员函数
 成员函数

□□

2.1.10. 友元函数

```

友元函数 [友元] 可以访问类的所有成员函数和成员变量。
public 友元函数 protected 友元函数 [友元] 友元函数 public
友元函数 const 友元函数 getter/setter 友元函数
[友元] 友元函数 public

```

```

友元函数 public/protected 友元函数 /
友元函数
友元函数
友元函数 private 友元函数 public/protected 友元函数 getter/setter 友元函数 const
友元函数 getter/setter 友元函数

```

```

友元函数

```

Effective C++, Item 22, Declare data members private

2.1.11. 友元类

```

友元类 public protected private 友元类
DISALLOW_COPY_AND_ASSIGN 友元类 private 友元类

```

```

友元类 友元类

```

```

友元类 C 友元类 C++ 友元类

```

```

// 友元类 bsl::ResourcePool 友元类
// 友元类
class ResourcePool {
public:
    // 友元类
    typedef void(*DestructorFunc)(void * p_object); // 友元类
    // 友元类/友元类
    ResourcePool();
    ~ResourcePool() {
        // 友元类 inline
        reset();
    }
}

```

```

// 初始化
void attach(void * p_object, DestructorFunc destructor); // 设置C++对象的buffer
...
// attach

template <typename T>
T& create(); // 初始化
... // create
... // 初始化

create_raw(),clone_cstring(),reset()
private:
// private
// 初始化
struct ObjectInfo; // 初始化
...
// 初始化
void push_info(); // 初始化push_info()pop_info()
void pop_info(); // 初始化private
...
// 初始化
ObjectInfo *_p_object_info_list;
...
DISALLOW_COPY_AND_ASSIGN(ResourcePool); // 初始化
};

```

2.1.12. 字典的迭代器

字典的迭代器

字典的迭代器 DictIterator 的定义

```
operator ==( const Dict&, const Dict& )
```

```

class DictIterator;
class Dict {
public:
    typedef DictIterator iterator;
    friend class DictIterator; // 友元
    friend bool operator ==(const Dict&, const Dict&); // 友元
private:
    ...

```

- “ Effective C++, Item 23, Prefer non-member non-friend functions to member functions

2.1.13.

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
/[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

```
catch [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

[illegible]

```

try {
    throw new RuntimeException("h()");
} catch (Exception e) {
    // ...
}

```

2.1.14. ☐ ☐ ☐ ☐ ☐ ☐

```

·[ ] int , " , "·[ ] /
[ ]

[ ] " "
[ ]
[ ]

```

2.2. C++

2.2.1.

```

·[ ] [RULE022] namespace C/C++ namespace
, [ main() ] [RULE023] using
·[ ] [RULE024] using namespace using class

[ ]
[ ]
[ ] using [ ]
< >::< > root using class using
[ ] cpp or

[ ]

```

```

// 
namespace saick {
...
// 
namespace my_module {
...
} // namespace my_module
} // namespace saick

```

BadCase:

```

// a.h
#include <string>
using namespace std; // using namespace

class A {
public:
    string name() const;
}

```

```

// a.cpp
#include "bsl/bsl_string.h"
#include "a.h"

using namespace bsl; // 在a.cpp中使用namespace

int main() {
    string str; // 使用bsl::string或std::string

    A a;
    cout << a.name() << endl; // 使用std::cout
}

```

编译

```

// a.h
#include <string>

class A {
public:
    std::string name() const; // 返回std::string类型的字符串
};

// a.cpp
#include "bsl/bsl_string.h"
#include "a.h"

using std::cout; // 使用std::cout
using std::endl;

int main() {
    bsl::string str; // 使用bsl::string

    A a;
    cout << a.name() << endl;
}

```

2.2.2. 编译

·[] [RULE025] (10)

```
class Person {  
public:  
    std::string get_name() const { //   
        return _name;  
    }  
    int get_age() const {  
        return _age;  
    }  
private:  
    std::string _name;  
    int _age;  
};  
  
inline void greet(const Person& person) { //inline  
    std::cout << "Hello, " << person.get_name() << "!" <<std::endl;  
}
```

Effective C++, item 30: understand the ins and outs of inlining

2.2.3.

·[]

++

```
class MyContainer {  
public:  
    class iterator {  
public:
```

```

        iterator& operator++() {
            ...
            return *this;
        }
};
};

```

2.2.4. 空 / 空

·[] 空
空
空

·[]
·[]

空 空 C++ 空 NULL 空 0 空 NULL
空

NULL

2.2.5. 空

·[] [RULE028] 空

空 空 bug

空

Effective C++, Item 37: Never redefine a function's inherited default parameter value

2.2.6. 空 RTTI

·[] 空 RTTI 空
空 dynamic_cast

·[]

空 空 RTTI 空
空

dynamic_cast

2.2.7. 空

·[] 空 ·[] 空 C++ 空

空 C++ 空
reinterpret_cast 空

C++ 空

static_cast 空 dynamic_cast 空 const_cast 空

A diagram showing a 16-bit register (represented by 16 small squares) with a 'cast' button below it. To the right of the 'cast' button is a 5-bit output (represented by 5 small squares).

C++

```

[] [] static_cast [] [] void * [] []

```

```
void foo(const void* network_buf) {
    const MyBuf * buf = static_cast<const MyBuf*>(network_buf); //XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
}
```

 `const_cast`

```
// 测试void old_copy(void *dest, const void *src, size_t n);  
// 测试  
void old_copy(void* dest, void* src, size_t n);  
  
// 测试  
void new_copy(void* dest, const void* src, size_t n) {  
    old_copy(dest, const_cast<void *>(src), n);    // 测试const  
}  
}
```

--	--

“ Effective C++, item 27: Minimize casting More Effective C++, item 02: Prefer C++ style casts

2.2.8. $++/--$

[illegible][illegible]

--	--

“ More Effective C++, item 06: Distinguish between prefix and postfix forms of increment and decrement operators

2.3. C/C++

2.3.1. 比較演算子

・[] 比較演算子
比較演算子
比較演算子
== 比較演算子
・[] [RULE033]
・[]

if (is_valid) {
...
}
if (!is_finished) {
...
}
// 比較演算子
if (my_value == 0) {
...
}
// 比較演算子
if (p_value == NULL) {
...
}
// 比較演算子
const double EPSILON = 1e-9; // 比較演算子
if (fabs(my_value) < EPSILON) {
...
}

if

```
// 比較演算子
if (is_valid) {
    ...
}
if (!is_finished) {
    ...
}
// 比較演算子
if (my_value == 0) {
    ...
}
// 比較演算子
if (p_value == NULL) {
    ...
}
// 比較演算子
const double EPSILON = 1e-9; // 比較演算子
if (fabs(my_value) < EPSILON) {
    ...
}
```

2.3.2. NULL, nullptr 0

・[] 比較演算子
0 比較演算子
NULL 比較演算子
'\0' 比較演算子
nullptr 比較演算子
C++11

if (is_valid) {

if

```
int i = 0;
double i = 0.0;
void* p = NULL;
char ch = '\0';
```

2.3.3. sizeof

·[] sizeof() ·[] sizeof()
 sizeof() 32/64
 sizeof() sizeof()
 bug

```
int g_my_id;

void func() {
    size_t size_of_my_id = sizeof(g_my_id);    // OK
}

int g_my_id;

void func() {
    size_t size_of_my_id = sizeof(int);        // NO! g_my_id long long size_t
}
```

2.3.4. typedef

·[] type tag typedef ·[
] struct typedef gcc

typedef typedef

YES:

```
class MyContainer {
public:
    typedef char value_type;    // type tag

    void my_method() {
        typedef std::vector<int>::iterator iterator_type;    //
        ...
    }
}
```

```
}
```

NO:

```
typedef tagMyStruct {  
    ...  
} MyStruct; // C++가 C를 모방한 것
```

2.3.5. goto

·[] [[RULE037]] goto

goto

2.3.6.

·[]

DISALLOW_COPY_AND_ASSIGN, CFATAL_LOG etc.)

·[]

·[]

return [] / /

inline

inline

```
inline int abs(int i) {  
    return (i >= 0 ? i : -i);  
}  
  
//  
template <typename T>  
inline T abs( T value ) {  
    return (value >= 0 ? value : -value);  
}
```

```
enum Color {
    RED,
    BLUE,
    GREEN,
    COLOR_COUNT
};
```

```
const int BUFFER_SIZE = 1048576;
```

```
const size_t BUFFER_SIZE = 1048576;
```

Effective C++, item 02: Prefer consts, enums, and inlines to #define

2.3.7. 命名空间

```
namespace std {
```

```
    int& at(int i) {
```

```
        return a[i];
```

```
    }
}
```

```
class Iterator; // 声明
class Container {
    Iterator* begin();
};
```

Effective C++, item 31: Minimize compilation dependencies between files

2.3.8. 单例模式 / 静态成员函数 / 静态成员变量

```
namespace std {
```

```
    int& at(int i) {
```

```
        return a[i];
```

```
    }
}
```

```
singleton<int>
```

```
/singleton<int>
```

```
namespace std {
```

```
    int& at(int i) {
```

```
        return a[i];
```

```
    }
}
```

```
main()
```

[illegible][illegible]

```

1  define [ ] / [ ] define
2  [ ]
3  [ ]
4  [ ]
5  [ ]
6  [ ]
7  [ ] file_name
8  [ ] strlen("filename")
9  [ ]

```

```
enum Color {
    RED,
    BLUE,
    GREEN,
    COLOR_COUNT
};
```

```
const size_t BUFFER_SIZE = 1048576;
```

2.3.10. const

·[] 常量表达式 const 常量表达式 常量表达式
 常量表达式 const 常量表达式 /常量表达式
 常量表达式 const_cast 常量 const [] const
 常量表达式

常量 常量 const 常量表达式 bug
 const_cast 常量 C 常量 C 常量 const
 const 常量

```
class MyClass {
public:
    void my_const_method(const OtherClass& arg) const;
};

const MyClass my_obj;    // const
const MyClass* p1;       // 常量指针
const MyClass* const p2 = &my_obj; // 常量指针常量
```

Effective C++, item 03: Use const whenever possible

2.3.11. 常量表达式

·[] 常量表达式 ·[] 常量表达式 4KB 常量表达式

常量 常量 (variable-length array): 常量表达式, GCC 常量

常量 常量表达式 crash 常量 crash
 常量表达式

3. 常量 / 常量

3.1. 目录

3.1.1. 文件扩展名

·[头文件] 通常使用 .h 扩展名。 .cpp 文件 (C++ 文件), C(头文件) 使用 C++(头文件) 扩展名。

头文件通常使用 .h 扩展名。 cpp 文件通常使用 .cpp 扩展名。 ; 头文件通常使用 ; 扩展名。

头文件通常使用 , some-project/foo/internal/fooserver.cc 扩展名。

```
#include "foo/public/fooserver.h" // 头文件
#include <sys/types.h>
#include <unistd.h>
#include <hash_map>
#include <vector>
#include "base/basictypes.h"
#include "base/commandlineflags.h"
#include "foo/public/bar.h"
```

3.1.2. #include<> 与 #include""

·[头文件] 通常使用 #include <> 扩展名 #include ""

3.1.3. #include 与 #include

·[头文件] 通常使用 include 扩展名

头文件 #include 扩展名

头文件

```
#include "bsl/ResourcePool.h"
#include "bsl/var/IVar.h"
#include "mylib/MyClass.h"
```

3.1.4. 头文件 include guards

·[头文件] [RULE060] 头文件 include guards ·[头文件] [RULE061] include guards 头文件
<SVNPATH>_<FILE>_H 头文件 SVNPATH 头文件 / 头文件 : trunk, branch-
xxx;

```
#include guards [ ] include [ ] #ifndef, #define, #endif
[ ]
```

```
[ ] [ ] include guards [ ] svn [ ]
include guards [ ] ; [ ]
```

```
// [ ] , svn [ ] https://svn.saick.net/ps/se/trunk/ac/strategy/
StrategyQueue/default/time_open_cluster_data.h [ ] :
```

```
#ifndef PS_SE_AC_STRATEGY_STRATEGYQUEUE_DEFAULT_TIME_OPEN_CLUSTER_DATA_H
#define PS_SE_AC_STRATEGY_STRATEGYQUEUE_DEFAULT_TIME_OPEN_CLUSTER_DATA_H
...
#endif // PS_SE_AC_STRATEGY_STRATEGYQUEUE_DEFAULT_TIME_OPEN_CLUSTER_DATA_H
```

3.2. []

3.2.1. []

·[] []

1. []
2. include guards
3. []
4. [] , [] ([] , []) , [] / [] ·[] CPP []
5. []
6. []
7. [] [] ·[]
[]

[] []

[] []

```
// Copyright 2017 Saick net. All Rights Reserved.
// Author: LastName FirstName (eric@saick.net)
//
// <[ ]>
[ ]Author[ ]
// Copyright 2017 Saick net. All Rights Reserved.
// Author: LastName FirstName (eric@saick.net)
//
// LastName2 FirstName2 (x@saick.net)
//
```

```
// <aaaaaaaaaaaaaaaaaaaa>
```

3.2.2. 条件

·[] [RULE064] 条件
[RULE066] 条件 ·[] 条件
: 条件 ; : 条件 if , 条件 else/else if
, else/else if .

条件
条件

条件

```
// mylib/AwesomeClass.h
namespace mylib {
class AwesomeClass {
    void foo() { //条件
        ...
    }
    void bar();
}; // MyClass

} //namespace mylib

// mylib/AwesomeClass.cpp
namespace mylib {
void AwesomeClass::bar() {
    while (...) {
        if (...) {
            ...
        } else {
            ...
        }
    } // while loop
}

} // namespace mylib
```

3.2.3. 循环

·[] [RULE068] 5
·[] [RULE069] 4

```
terminal
4
```

3.2.3.1.

·[] [RULE070]

namespace (100 ,); namespace

;

```
namespace bsl {
namespace var {
class IVar {
};

} // namespace var
} // namespace bsl
```

3.2.3.2.

·[] [RULE071]

3.2.3.3.

·[] [RULE072] public/protected/private

```
class MyClass {
public:
    void foo();
protected:
    void bar();
private:
    void meow();
    int _haha;
};
```

3.2.3.4. 初始化器

·[] 初始化器, 初始化器, 初始化器. 初始化器
初始化器 4初始化器 8初始化器

```
// 初始化器  
MyClass::MyClass(int var) : _some_var(var), _some_other_var(var + 1) {}  
  
// 初始化器 8初始化器 初始化器  
// the first initializer line:  
MyClass::MyClass(int var) :  
    _some_var(var),           // 8 space indent  
    _some_other_var(var + 1) { // lined up  
    do_something();  
    ...  
}
```

3.2.3.5. 初始化器

·[] 初始化器, 初始化器
初始化器 8初始化器

初始化器

```
void foo();  
void this_is_a_very_long_function_name(  
    VeryLongLongDogType *p_my_dog,  
    const std::map<std::string, int>& a_wierd_dictionary);
```

3.2.3.6. 初始化器

·[] 初始化器 100
初始化器 8初始化器

初始化器

```
grey_wolf.eat(white_sheep);  
hungry_lion.eat(  
    big_grey_wolf,  
    little_white_sheep,  
    quick_brown_fox,  
    lazy_dog);
```

3.2.3.7. if/while

·[] [RULE076] if
8
if ... else if ... else
//pass //do nothing

if
if ... else if ...
else

```
if (you.come_from(EARTH)) {  
    return "hello";  
} else if (you.come_from(MARS)) {  
    return "@%!%#@!~&$%$";  
} else {  
    throw YouCheatMeException() << BSL_EARG << "your name:" << you.name();  
}  
  
// Demo  
if (you.com_from(EARTH)  
    || you.com_from(MARS)  
    || you.com_from(VENUS)) {  
    return "you come from solar system";  
}
```

“ 1TBS

3.2.3.8. switch

·[] switch
case
break
return
case
default
//pass //do nothing
switch
case
break
return
case
switch
default

```
switch (cur_char) {
case '\"':
case '\\'': {
    return "it's a string!";
}
default:
    printf("not found yet");
    break;
}
```

```
switch (cur_char) {
    case '"':
        printf("wow!"); // break return
    case '\':
        return "it's a string!";
} // default
```

```
[ ] [RULE083] 
while 
```

`do{...}` ☐ `do-`

```
while (iter.next()) {  
    // pass  
}
```

```
for (size_t i = 0; i < N; ++i) {
    sum += arr[i];
}

// do-while
do {
```



```

call_some_func(arg1 , arg2 , arg3); // [], []

template <typename T>          // []
template<typename T>          // [], template<[]

class Derived : public Base // []
class Derived:public Base    // [], []

MyClass::MyClass() : _member_var(0) // []
MyClass::MyClass():_member_var(0)   // [], []

++i; !i; ~i; *i; &i;          // []
++ i; ! i; ~ i; * i; & i;     // []

a + b; a ~ b; a || b; a << b; a = b; a %= b;    // []
a+b; a~b; a||b; a<<b; a=b; a%=b;                // [], []

a ? b : c      // []
a?b:c          // [], []

a.b; a->b; a.*b; a->*b; a::b;                  // []
a . b; a -> b; a .* b; a ->* b; a :: b;       // [], []

xx_comparator(a, b); xx_map["key"];           // []
xx_comparator (a, b); xx_map ["key"];         // []
xx_comparator( a, b ); xx_map[ "key" ];       // []

```

3.2.5. []

```

·[[] ][RULE093] [] 100
[]

[] []
[]
[] 12[] laptop[]
[] ,

[] []

```

```

UB_LOG_WARNING(
    "An exception[%s] was thrown from[%s:%d:%s] " // [], []

```

```

    "with a message[%s]! stack_trace:%s%s",
    e.name(), e.file(), int(e.line()), e.function(),           // 関数名とファイル名
    e.what(), e.get_line_delimiter(), e.stack()
);

```

3.2.6.

--	--	--	--

```

.[ ] [ ] 100 [ ] .[ ] .h [ ]
[ ] , [ ] .hpp [ ]

[ ]
[ ]
[ ]

```

3.2.7.

--	--	--	--	--

```
.[[] ] [] return[[]][[]][[]][[]][[]][[]][[]][[]]
[] return [][[]][[]][[]][[]][[]][[]][[]] ()
[]
```

```
return (ret_val);
```

3.2.8.

--	--	--	--

The diagram shows a 4x4 grid of memory blocks. The first row contains four blocks, each with a label to its right: `·[ ]`, `·[ ]`, `const`, and `const`. The second row contains four blocks, each with a label to its right: `·[ ]`, `·[ ]`, `const`, and `const`. The third row contains four blocks, each with a label to its right: `·[ ]`, `·[ ]`, `const`, and `const`. The fourth row contains four blocks, each with a label to its right: `·[ ]`, `·[ ]`, `const`, and `const`. The labels are: `·[ ]`, `·[ ]`, `const`, `const`, `·[ ]`, `·[ ]`, `const`, `const`, `·[ ]`, `·[ ]`, `const`, `const`, `·[ ]`, `·[ ]`, `const`, `const`.

```

/*
C/C++ 0000000000, 0000, 000/000000.

00000000000000000000 ``const`` 00. 00000000+00000000000000000000const00000000000000000000
000000000000(00000000000000000000), 000000 ``const`` 00

00000, 0000000000: 0000, 0000000. 00/000000 (0000/00000) 0000000, 00000000000000, 00000.

*/

// 0000000000const00

```

```

void foo(const std::string& input1, const MyClass& input2);

// 常量的非静态成员函数
void foo(int input1, float input2);

// 常量的静态成员函数
void foo(const MyClass* input1);

// 常量的静态成员函数
void foo(int* output1, MyClass* output2);

// 常量的静态成员函数
void foo(std::string* out);

// 常量的静态成员函数
void foo(const InClass& input1, OutClass* out1);

// 常量的静态成员函数
void foo(const InClass& input1, // 常量const
        int input2, // 常量POD
        OutClass* output1); // 常量const

// 常量的静态成员函数
void foo(const InClass& input1,
        int* inout, // 常量const
        OutClass* output1)
常量成员函数:
// Always have named parameters in interfaces.
class Shape {
public:
    virtual void rotate(double radians) = 0;
}

// Always have named parameters in the declaration.
class Circle : public Shape {
public:
    virtual void rotate(double radians);
}

```

```
// Comment out unused named parameters in definitions.
void Circle::rotate(double /*radians*/) {}
```

3.3. $\frac{1}{2}$ / $\frac{1}{2}$ 3.3.1.

--	--	--	--	--	--

```
.[ ] [ ]  
[ ]  
[ ]  
[ ]  
[ ]  
[ ]  
[ ]  
  
[ ] [ ]  
[ ]  
[ ]  
vse [ ]  
_evaluator [ ] [FALL BACK] [ ] extern "C" [ ] /  
[ ]  
[ ]
```

```
// 000000000000 00000000000 000000000000000000;

// 00000000

int num_errors;           // Good.

int num_completed_connections; // Good.


// 00000000

int n;                    // Bad - 000

int nerr;                 // Bad - 0000000

int n_comp_conns;        // Bad - ambiguous abbreviation.


// 00: 000000000000;

// 0000:

int num_dns_connections;  // 00000"DNS"00000

int price_count_reader;   // OK, price count. Makes sense.


// 000000

int wgc_connections;      // wgc 00??

int pc_reader;            // pc000000!
```

```
// 测试测试测试测试测试
int error_count;           // Good!
int error_cnt;             // Bad! 测试测试测试, 测试测试测试;
```

3.3.2. 测试

·[] 测试 .h 测试 .hpp 测试 C++ 测试 .cpp 测试 C 测试 .c 测试 ·[] [RULE100] 测试 , 测试 _测试 ; 测试 :this_is_my_awesome_file.cpp ·[] 测试 <测试>_test.h(.cpp) 测试 ·[] 测试 .h 测试 .hpp 测试 ·[] 测试 lib测试 <测试>_<测试>.h 测试 api 测试 include 测试 api 测试 mylib.h,mylib_utils.h,mylib_api.h

测试 测试

3.3.3. 测试

·[] [RULE102] 测试 ·[] [RULE103] 测试 ·[] 测试

测试 using namespace 测试

3.3.4. 测试

·[] [RULE103] 测试 ·[] 测试 ·[] 测试 accessor 测试 : getter: my_member_variable(), setter: set_my_member_variable(), mutable: mutable_my_member_variable()

测试 accessor 测试 Google protobuf 测试

测试

```
class Person {
public:
    ...
    const IdCard& id_card() const { return _id_card; } // getter
    IdCard* mutable_id_card() { return &_id_card; }
    void set_id_card(const IdCard& id_card) { _id_card = id_card; } // setter

private:
    IdCard _id_card;
```

```
int _age;

int get_age() const { return _age; }

};
```

3.3.5. 枚举类型

·[] [RULE105] 枚举类型

Interface

·[] [RULE106]

·[]

enum struct class typedef template

```
enum Color {
    RED,
    BLUE,
    GREEN,
    COLOR_COUNT
};

struct Point {
    int x;
    int y;
};
```

3.3.6. 全局变量

·[] [RULE107]

·[] [RULE108]

g

3.3.7. 静态变量

·[] [RULE109]

·[]

i,j,k

i,j,k

cur_user_index

3.3.8. 静态常量

·[] [RULE110]

s_

·[] [RULE111]

s

```
static, ;  
_s_
```

3.3.9. /

```
[RULE112],  
MyClass::_my_attr [RULE113]  
MyStruct::my_attr
```

3.3.10.

```
[ ] const
```

3.3.11. /

```
[ ]  
MY_LIB_MY_MACRO_THAT_SCARES_SMALL_CHILDREN
```

3.3.12.

```
[ ] int [ ]<stdint.h>  
int64_t, int32_t  
int  
unsigned long long
```

3.4.

3.4.1.

```
[ ] : , UTF-8 [ ] [ ]  
[ ]  
[ ] [ ]  
[ ]  
[ ]  
[ ]  
[ ]  
[ ]
```

3.4.2.

```
[ ] [ ] TODO  
; [ ] TODO : ,
```



```
·[ ] [RULE124] auto_ptr ·[ ] [RULE125]
strcpy()/strcat()/strdup()/sprintf()
snprintf()/strncat()/strndup()/snprintf() ·[ ] [RULE126] strncpy
ullib ul_strncpy
auto_ptr auto_ptr
strncpy() \0
snprintf
```

4.

4.1. /

```
·[ ] , ·[ ]
,
. ,
-- gtest. ,
.
:
```

```
class FooEnvironment : public testing::Environment {
public:
    virtual void SetUp() {
        std::cout << "Foo FooEnvironment SetUP" << std::endl;
    }
    virtual void TearDown() {
        std::cout << "Foo FooEnvironment TearDown" << std::endl;
    }
};

// Example:
// template
class FooTest : public testing::Test {
public:
    ...
}
```

```
typedef std::list List;
static T shared_;
T value_;
};
```

4.2.

```
·[ ] 
;
;
```

4.3. Windows

Windows C++ C++

4.3.1.

```
·[ ] [WCPP001] [WCPP002] [WCPP003] ,·[ ] 
[WCPP004] ,m_·[ ] C
·[ ] C++ 
m_xxxx g_xxx ·[ ] I 
: , EatApple :
, eatApple
```

4.3.2.

4.3.2.1.

```
·[ ] [WCPP006] { } 
[WCPP005] 

```

```
if (i == j)
{
    //
    //i for do while switch case
    i++;
}
```

·[] () , int x = 3 & (4<<32);

4.3.3. WINAPI

VS IDE

·[] WINAPI
include guard #pragma once

DWORD HANDLE ·[] C++

```
#ifndef ABC_H  
#define ABC_H  
#endif
```

4.4.

" " "

·[] " " "