

# C++ | | | | | | | |--|--|--|--|--|--| | | | | | | | |--|--|--|--|--|--|

- |  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

  - |  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|--|--|--|--|--|--|
  - C & C++

--	--	--









# 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"
```



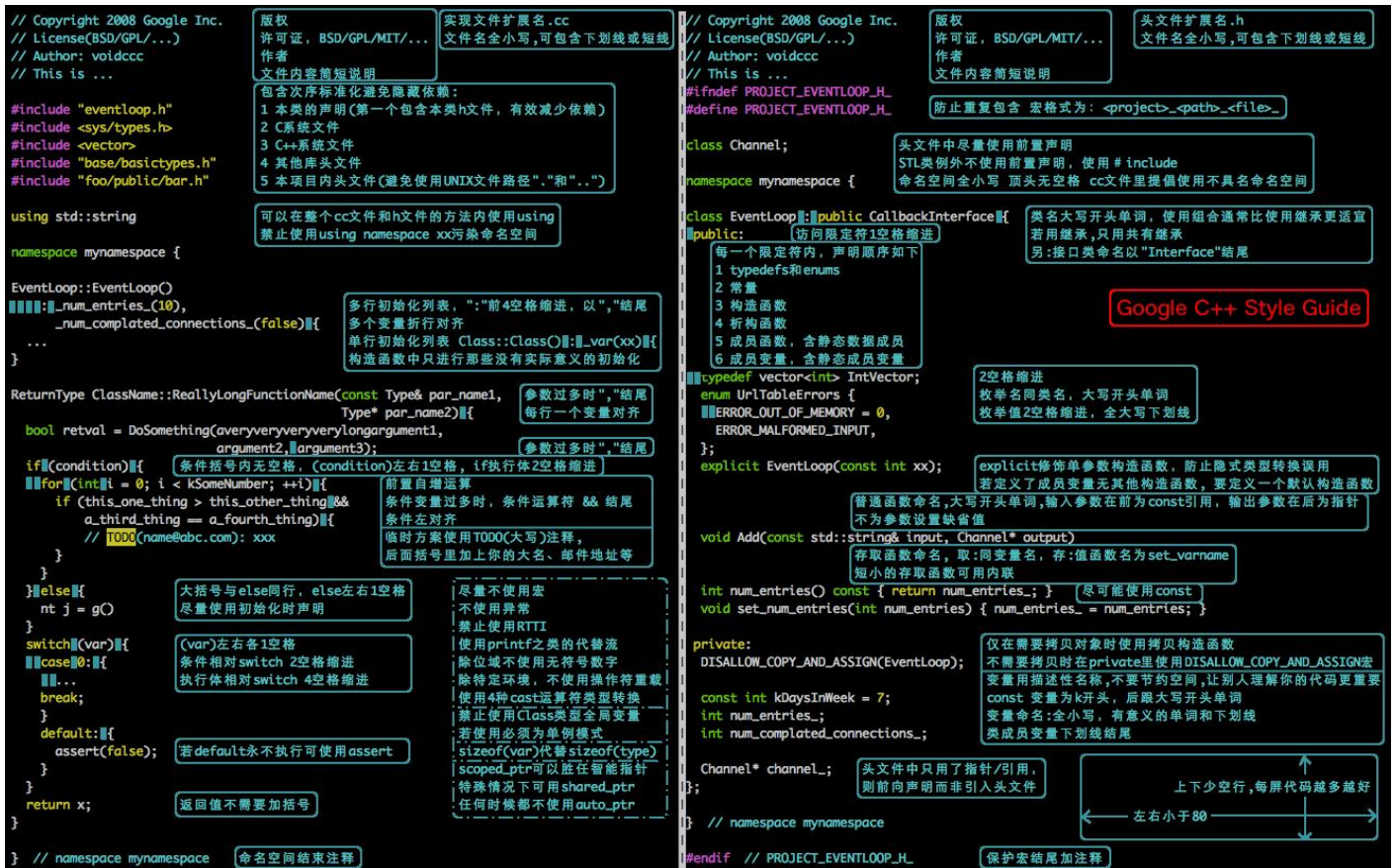
```

    },
    "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

IncludesMainRegex: '([-\_](test|unittest))?\$'

IncludesMainSourceRegex: ''

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 typenamees 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 }






Google C++ Style Guide C C++ [shibborage@gmail.com](mailto:shibborage@gmail.com)


/  C++

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

· [ struct   initialize()   reset()   validate() ] class

☐ C+ ☒ struct ☐ class ☐ struct ☒ C ☐ C

```
// 
struct Coordinate {
    int x;
    int y;
    int z;
};

// 
class Cat {
public:
    void meow();
private:
    ...
};
```



```
};
```

## 2.1.2. 显式转换操作符

·[1] 1. `DISALLOW_COPY_AND_ASSIGN` 宏定义 `init()` 函数 `Person::Person(const Person& p): _name(p._name)`, 初始化

1.

2.

3. `Person::Person(const Person& p): _name(p._name)`

4.

5. `Person::Person(const Person& p): _name(p._name)`

6.

7. `Person::Person(const Person& p): _name(p._name)` C++ `Person::Person(const Person& p): _name(p._name)`

8.

9. `Person::Person(const Person& p): _name(p._name)`

```
class Person {
public:
    // 显式转换操作符
    explicit Person(const std::string& name) : _name(name) {} // 显式转换操作符
private:
    std::string _name;
};

DISALLOW_COPY_AND_ASSIGN(Person);

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

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

10.

Effective C++, item 04: Make sure that objects are initialized before they're used More Effective C++, Item 04: Avoid gratuitous default constructors 显式转换操作符non-



### 2.1.3.

```
·[ ]
[ ] private [ ]
```

```
[ ]
[ ]
default constructor[ MyClass::MyClass() [ ]
[ ]/[ ]
```

```
[ ]
[ ]C++ [ ] private [ ]
[ ]
```

```
[ ]
```

More Effective C++, Item 04: Avoid gratuitous default constructors Effective C++, item 06: Explicitly disallow the use of compiler-generated functions you do not want

```
[ ]
[ ]""[ ]
```

```
class String {
public:
    String() : // [ ]
        _str(_S_EMPTY_C_STR) {} // [ ]_S_EMPTY_C_STR[ ]

    ~String() {
        if (_str != _S_EMPTY_C_STR) {
            free(_str);
        }
    }

    const char* c_str() const {
        return _str;
    }
}
```



```
private:
    char* _str;
    static char _S_EMPTY_C_STR[1]; // = ""
};
```

## 2.1.4. 显式转换操作符

·[ ] 显式转换操作符 ·[ ] 显式转换操作符

·

·

implicit constructor MyClass::MyClass(MyArg arg) MyArg MyClass 显式转换操作符

explicit constructor explicit MyClass::MyClass(MyArg arg) 显式转换操作符

[ explicit ] 显式转换操作符 bug 显式转换操作符

Person( const std::string& ) [ explicit string ] Person 显式转换操作符

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

string 显式转换操作符 C 显式转换操作符

```
class String {
public:
    String(const char* cs); // 显式转换操作符

private:
    ...
};
```

·

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

## 2.1.5. 显式转换操作符







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. 点

·[ ]

点

点

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

点

点

点

点

```
class Point {
public:
    Point(int x, int y) : _x(x), _y(y) {}
    Point(const Point& other) :
        _x(other._x), _y(other._y) {}
    Point& operator=(const Point& other) { //点
        _x = other._x;
        _y = other._y;
        return *this;
    }
private:
    int _x;
    int _y;
}
```

点

```
class Person {
public:
    explicit Person(const std::string& name) : _name(name) {}
private:
    std::string _name;
    DISALLOW_COPY_AND_ASSIGN(Person);
}
```





- “ 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

2.1.7. 

--	--	--	--

RULE010

```
❏ destruct MyClass::~~MyClass() ❏❏
```

```
delete pBase;
```

```
class Connection {  
public:  
  
    void close(); // [0][1][2][3][4][5][6][7]"[8][9][10][11]"->"[12][13][14][15]  
  
    ~Connection() {  
  
        if (_connection_state == CONNECTED) {  
  
            close(); // [0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15]  
  
        }  
  
    }  
  
};
```



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

2.1.8. 

--	--

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

[illegible]



❌ ❌❌❌"❌❌"❌❌❌

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

    // 初始化成员变量
    virtual ~Chinese() {}

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

    // 初始化成员变量
    virtual void greet() { say("❌❌❌"); }

private:
    std::string _name;
};

// 初始化成员变量

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

    // 初始化成员变量
    virtual void greet() { say("❌❌❌?"); }
};
```

❌

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. ❌❌❌



·[ ]

Effective C++, Item 40: Use multiple inheritance judiciously

## 2.1.10.

·[ ] ·[ ] public protected ·[ ] publi

public/protected private public/protected getter/setter const getter/setter

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

## 2.1.11.

·[ ] public protected private DISALLOW\_COPY\_AND\_ASSIGN private

C C++

```
//  bs::ResourcePool
// 
class ResourcePool {
public:
    // 
    typedef void(*DestructorFunc)(void * p_object); // 
    // 
    ResourcePool();
    ~ResourcePool() {
        //  inline
        reset();
    }

    // 
```







`[ ] catch`

例: 以下代码在 C++ 中, 调用 `initialize()` 函数, 初始化 C++ 环境, 并调用 `Python, Java` 等 C++ 函数。

`throw`, `f()`, `g()`, `h()`, `h`, `f`, `g`, `" "`, `:`, `,`, `.`, `.`

·[00] 0000 int 0000, 0000"0000, 000000" ·[00] 00000000/000000000000

2.2.1. 

--	--	--	--

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

[illegible]

```
// [ ] [ ] [ ] [ ] [ ] [ ]
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; // ~.cpp~~~~using namespace

int main() {
    string str; // ~~~~~bsl::string[]std::string

    A a;
    cout << a.name() << endl; // ~~~~std~~~~~
}
```

~~~~~

```
// a.h
#include <string>

class A {
public:
    std::string name() const; // []std::string[]string~~~~~
};

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



```
using std::cout; // using class
using std::endl;

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

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

## 2.2.2.

·[ ] [RULE025] ·[ ] (10)

10

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;
}
```

10

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

## 2.2.3.







·[ ] [ ] ·[ ] C++ [ ]

[ ] C++ [ ] static\_cast dynamic\_cast const\_cast reinterpret\_cast [ ]

[ ] [ ] \_cast [ ] C++ [ ]

[ ] static\_cast void \* [ ]

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

[ ] const\_cast [ ]

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

[ ]

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

## 2.2.8. ++/--

·[ ] [ ] ( ).

[ ] ++/-- [ ]

[ ]

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



## 2.3. C/C++ 関数

### 2.3.1. 関数

・[関数] 関数名 [引数] [戻り値] [関数体] == [関数] [関数名] [引数] [戻り値] [関数体] [RULE033] 関数

関数 if 関数 == 関数 == 関数 != 関数 関数/関数

関数

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

### 2.3.2. NULL, nullptr, 0

・[関数] 0 0.0 NULL '\0' 関数 nullptr 関数 C++11 関数

関数 関数

関数

```
int i = 0;
double i = 0.0;
```



```
void* p = NULL;
char ch = '\0';
```

### 2.3.3. sizeof

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

```
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

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++에서 C 스타일
```

## 2.3.5. goto

·[ ] [RULE037] ] goto

goto

## 2.3.6.

·[ DISALLOW\_COPY\_AND\_ASSIGN CFATAL\_LOG etc.) . [ c ## W # break continue return

inline

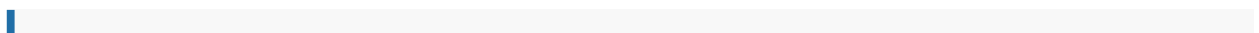
inline

```
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

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





## 2.3.7. 友元函数

·[友元] 友元函数/友元类

友元函数/友元类

友元函数

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

友元

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

## 2.3.8. 静态成员函数/静态成员变量

·[友元] 友元函数/友元类 single main() 友元函数/友元类

友元函数/友元类 友元函数/友元类 C++ 友元函数/友元类, 友元函数/友元类, [

## 2.3.9. 常量表达式

·[友元] [RULE050] 友元函数/友元类 ·[友元] [RULE051] 友元函数/友元类 enum 友元函数/友元类 define ·[友元] 友元函数/友元类

友元函数/友元类 define strlen("filename") file\_name define 友元函数/友元类 友元函数/友元类

友元函数/友元类

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

友元函数/友元类 const 友元函数/友元类



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

--	--

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

### 2.3.10. const

·[ ] [ ]const [ ] [ ]const [ ]const [ ]/[ ]const [ ]

```

const [bug, const_cast] = C.C

```

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

const MyClass my_obj; // const
const MyClass* p1;    //
const MyClass* const p2 = &my_obj; //
```

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

const MyClass my_obj; // const
const MyClass* p1;    //
const MyClass* const p2 = &my_obj; //
```

--	--

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

2.3.11. 

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

.[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] .[ ] [ ] [ ] [ ] [ ] 4KB [ ] [ ] [ ] [ ]

□□ □□□□(variable-length array):□□□□□□□□, □□ GCC □□□□

crash crash

3. ☐ ☐ ☒ ☐

### 3.1. | | | | | | |--|--|--|--|--| | | | | | | |--|--|--|--|--|



### 3.1.1. `#include`

·[1] `foo.cpp`(`foo.h`), C(`foo.h`)C++(`foo.h`)

`foo.cpp` `foo.h`; `foo.h` `foo.h`;

`foo.h`, `some-project/foo/internal/fooserver.cc` `foo.h`:

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

### 3.1.2. `#include <>` `#include ""`

·[1] `#include <>` `#include ""`

### 3.1.3. `#include`

·[1] `#include`

`#include` `foo.h`

`foo.h`

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

### 3.1.4. `#include guards`

·[1] [RULE060] `#include guards` ·[1] [RULE061] `include <SVNPATH>_<FILE>_H` `SVNPATH` `trunk`, `branch-xxx`;

`include guards` `#ifndef` `#define` `#endif` `foo.h`

`include guards` `svn` `include guards` `foo.h`; `foo.h`

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

```
// 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)  
//  
// <[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]>
```

--	--



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

} //namespace mylib

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

} // namespace mylib
```

### 3.2.3. []

·[[]] [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. 命名空间

·[命名空间] [RULE071] 命名空间

### 3.2.3.3. 命名空间

·[命名空间] [RULE072] public/protected/private命名空间

命名空间

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

### 3.2.3.4. 命名空间

·[命名空间] 命名空间命名规则，命名空间，命名空间：命名空间。命名空间命名规则4命名空间命名规则8命名空间

```
// 命名空间
MyClass::MyClass(int var) : _some_var(var), _some_other_var(var + 1) {}

// 命名空间 命名8命名空间
// 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. 命名空间







```
return "you come from solar system";
}
```

11

“1TBS”

### 3.2.3.8. switch□□

The diagram illustrates the mapping of C switch-case statements to Rust match expressions. It shows two lines of code side-by-side, with boxes highlighting corresponding keywords and structures.

**Left side (C code):**

```
switch (i) {
  case 1:
    break;
  case 2:
    return 1;
  default:
    //pass
    //do nothing
}
```

**Right side (Rust code):**

```
match i {
  1 => {},
  2 => { return 1; },
  _ => {}
}
```

**Mapping:**

- `switch` maps to `match`.
- `(i)` maps to the variable `i`.
- `case` maps to `=>`.
- `break` maps to `{}`.
- `return 1` maps to `{ return 1; }`.
- `default` maps to `_ => {}`.
- `//pass` and `//do nothing` map to `{}`.

[ case [ break [ return [ case [ ] ] ] default [ ] ] ]

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

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

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

```
switch (cur_char) {
case '\0':
    printf("wow!"); // break return
case '\n':
    return "it's a string!";
} // default
```

3.2.3.9. 

--	--	--	--

[ ] [RULE083] [ ] do-while [ ] while [E084]

Construct	Frequency
do-while	10
while	9
do{...}	8
	7
	6
	5
	4
	3
	2
	1
	0



--	--	--	--

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

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

```
for (size_t i = 0; i < N; ++i) {  
    sum += arr[i];  
}  
  
// do-while  
  
do {  
    ...  
} while (++it != end); // □  
  
do {  
    ...  
}  
  
while (++it != end); // □□□□□□□□□□□□□□□□□□□□□□
```

--	--

“1TBS”

3.2.4. 

--	--	--	--	--

[illegible]

--	--

```
if (is_good && is_powerful) // []
if(is_good && is_powerful) // [] if[[]]
if ( is_good && is_powerful ) // [], [[]]
switch (type) // []
while (condition) // []
catch (std::exception& ex) // []
for (int i = 0; i < 10; ++i) // []
for (int i = 0;i<10;+i) // [], [[]]
```



```
call_some_func(arg1, arg2, arg3); // []
call_some_func (arg1, arg2, arg3); // [], []
call_some_func( arg1, arg2, arg3 ); // [], []
call_some_func(arg1,arg2,arg3); // [], []
call some func(arg1 , arg2 , arg3); // [], []
```

```
class Derived : public Base // 1
class Derived:public Base // 1, 11111111
```

```
++i; !i; ~i; *i; &i;      // □
++ i; ! i; ~ i; * i; & i;  // □□□□□□□□□□□□□□□□
```

```
a + b; a ~ b; a || b; a << b; a = b; a %= b; // □
```

a ? b : c    // □

a?b:c        // □, □□□□□□□□□□

[illegible]

```
xx_comparator(a, b); xx_map["key"]; // []
xx_comparator (a, b); xx_map ["key"]; // [ ]
xx comparator (a, b ); xx map[ "key" ]; // [ ]
```

[illegible]

12 laptop



```
UB_LOG_WARNING(
    "An exception[%s] was thrown from[%s:%d:%s] " // 异常信息, 文件
    "with a message[%s]! stack_trace:%s%s",
    e.name(), e.file(), int(e.line()), e.function(), // 异常信息
    e.what(), e.get_line_delimiter(), e.stack()
);
```

## 3.2.6. 异常

·[文件] 异常信息100字节 ·[文件] .h 异常信息, 异常信息, 文件 .hpp 异常信息

异常信息

## 3.2.7. 异常

·[文件] 异常return异常信息

异常 return 异常信息 异常()

异常

```
return (ret_val);
```

## 3.2.8. 异常

·[文件] 异常信息 ·[文件] 异常 const 异常信息 const 异常

异常 异常信息 异常信息 异常 异常

异常

```
/*
C/C++ 异常信息, 异常, 异常/异常.
异常信息 ``const`` 异常. 异常+异常const异常
异常(异常), 异常 ``const`` 异常
异常, 异常信息: 异常, 异常. 异常/异常 (异常/异常) 异常, 异常信息, 异常.
*/
// 异常const异常
void foo(const std::string& input1, const MyClass& input2);

// 异常
void foo(int input1, float input2);
```



```

// [ ]const[ ]
void foo(const MyClass* input1);

// [ ]
void foo(int* output1, MyClass* output2);

// [ ]std::string*[ ]char*[ ]char*[ ]buffer[ ]
void foo(std::string* out);

// [ ]
void foo(const InClass& input1, OutClass* out1);

// [ ]
void foo(const InClass& input1, // [ ]const[ ]
        int input2, // [ ]POD[ ]
        OutClass* outpu1); // [ ]

// [ ]
void foo(const InClass& input1,
        int* inout, // [ ]
        OutClass* output1)

[ ]:
// 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. [ ]/[ ]



3.3.1. 

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

`. [ ] creat [ ] usr [ ]`

VSriptEvaluator vse \_evaluator ~~WHEN BACK~~ extern "C"

--	--

```
// ██████████ ██████████ ████████████████████;
// ████████
int num_errors;           // Good.
int num_completed_connections; // Good.
// ████████
int n;                    // Bad - ████
int nerr;                 // Bad - ██████
int n_comp_conns;        // Bad - ambiguous abbreviation.
// ██: ████████████;
// █████:
int num_dns_connections; // █████ "DNS" █████
int price_count_reader;  // OK, price count. Makes sense.
// █████
int wgc_connections;     // wgc ███?
int pc_reader;           // pc█████!
// ████████████████████
int error_count;         // Good!
int error cnt;          // Bad! ████████, ████████;
```

### 3.3.2.

```
[...] .h [...] .hpp [...] C++ [...] .cpp [...] C [...] .c [...] [RULE100] [...] <...>_test.h(...) :t
```

<> <...>ap mylib.h,mylib utils.h,mylib api.h lib

3.3.3. 

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

[illegible]



using namespace

### 3.3.4.

[RULE103] getter: my\_member\_variable(), setter: set\_my\_member\_variable(), mutable: mutable\_my\_member\_variable()

accessor Google protobuf

```
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.

[RULE105] [RULE106]

enum struct class typedef template

```
enum Color {
    RED,
    BLUE,
    GREEN,
    COLOR_COUNT
};
struct Point {
    int x;
    int y;
};
```



3.3.6. 

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

·[ ] [RULE107] [ ]g[ ]·[ ] [RULE108] [ ]

3.3.7. 

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

·[ ] [RULE109] [ ] ·[ ] [ ]

3.3.8. 

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

·[ ] [RULE110\_s\_ ] ·[ ] [RULE\_s\_ ]

```
[s_  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. 

--	--	--	--

```
#include <stdint.h>
```

unsigned long long

### 3.4.

3.4.1. 

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

·[ ] [ ]: [ ], [ ] UTF-8 [ ] ·[ ] [ ] ·[ ] [ ]

[illegible]3.4.2. 

--	--	--	--











### 4.3.2.1. 条件

・[C] [WCPP006] { } 条件文  
・[C] [WCPP005] 条件文

例

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

・[C] 例 int x = 3 & (4<<32);, 例

### 4.3.3. WINAPIとVSとIDE

・[C] 例 WINAPI 例 DWORD, HANDLE 例 C++ 例 include guard, #pragma once

例

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

## 4.4. 文字列

・[C] 例 " " 例