

C++ ????????

- [\[1\]](#)
- [\[2\]](#)
- [C & C++ \[3\]](#)

????



????????

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": [
      "//-----"
    ],
    "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 ??

```

// Copyright 2008 Google Inc.
// License(BSD/GPL/...)
// Author: voidccc
// This is ...

#include "eventloop.h"
#include <sys/types.h>
#include <vector>
#include "base/basictypes.h"
#include "foo/public/bar.h"

using std::string

namespace mynamespace {

EventLoop::EventLoop()
    : _num_entries_(10),
      _num_complated_connections_(false) {}

...

ReturnType ClassName::ReallyLongFunctionName(const Type& par_name1,
                                               Type* par_name2) {
    bool retval = DoSomething(averyveryveryverylongargument1,
                              argument2, argument3);

    if(condition) {
        for(int i = 0; i < kSomeNumber; ++i) {
            if (this_one_thing > this_other_thing &&
                a_third_thing == a_fourth_thing) {
                // TODO(name@abc.com): xxx
            }
        }
    } else {
        int j = gO
    }
    switch(var) {
        case 0: {
            break;
        }
        default: {
            assert(false);
        }
    }
    return x;
}
// namespace mynamespace

```

```

// Copyright 2008 Google Inc.
// License(BSD/GPL/...)
// Author: voidccc
// This is ...

#ifndef PROJECT_EVENTLOOP_H_
#define PROJECT_EVENTLOOP_H_

class Channel;

namespace mynamespace {

class EventLoop : public CallbackInterface {
public:
    typedef vector<int> IntVector;
    enum UrLTableErrors {
        ERROR_OUT_OF_MEMORY = 0,
        ERROR_MALFORMED_INPUT,
    };
    explicit EventLoop(const int xx);

    void Add(const std::string& input, Channel* output);

    int num_entries() const { return num_entries_; }
    void set_num_entries(int num_entries) { num_entries_ = num_entries; }

private:
    DISALLOW_COPY_AND_ASSIGN(EventLoop);

    const int kDaysInWeek = 7;
    int num_entries_;
    int num_complated_connections_;

    Channel* channel_;
};
// namespace mynamespace
#endif // PROJECT_EVENTLOOP_H_

```

Google C++ Style Guide

- 版权许可证: BSD/GPL/MIT/... 文件内容简短说明
- 实现文件扩展名 .cc 文件名全小写, 可包含下划线或短横线
- 头文件扩展名 .h 文件名全小写, 可包含下划线或短横线
- 防止重复包含 宏格式为: <project>_<path>_<file>_
- 头文件中尽量使用前置声明 STL类例外不使用前置声明, 使用 #include 命名空间全小写 顶头无空格 cc文件里提倡使用不具命名空间
- 类名大写开头单词, 使用组合通常比使用继承更适宜 若用继承, 只用共有继承 另: 接口类命名以 "Interface" 结尾
- 每一个限定符内, 声明顺序如下: 1 typedefs和enums 2 常量 3 构造函数 4 析构函数 5 成员函数, 含静态数据成员 6 成员变量, 含静态成员变量
- 2空格缩进 枚举名同类名, 大写开头单词 枚举值2空格缩进, 全大写下划线
- explicit修饰单参数构造函数, 防止隐式类型转换调用 若定义了成员变量无其他构造函数, 要定义一个默认构造函数 不为参数设置缺省值
- 普通函数命名, 大写开头单词, 输入参数在前为const引用, 输出参数在后为指针 短小的存取函数可用内联
- 存取函数命名, 取:同变量名, 存:值函数名为set_varname
- 尽可能使用const
- 仅在需要拷贝对象时使用拷贝构造函数 不需要拷贝时在private里使用DISALLOW_COPY_AND_ASSIGN宏 变量用描述性名称, 不要节约空间, 让别人理解你的代码更重要 const 变量为k开头, 后跟大写开头单词 变量命名: 全小写, 有意义的单词和下划线 类成员变量下划线结尾
- 头文件中只用了指针/引用, 则前向声明而非引入头文件
- 上下少空行, 每屏代码越多越好
- 左右小于80
- 保护宏结尾加注释

.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 typedefs 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: "*"
```

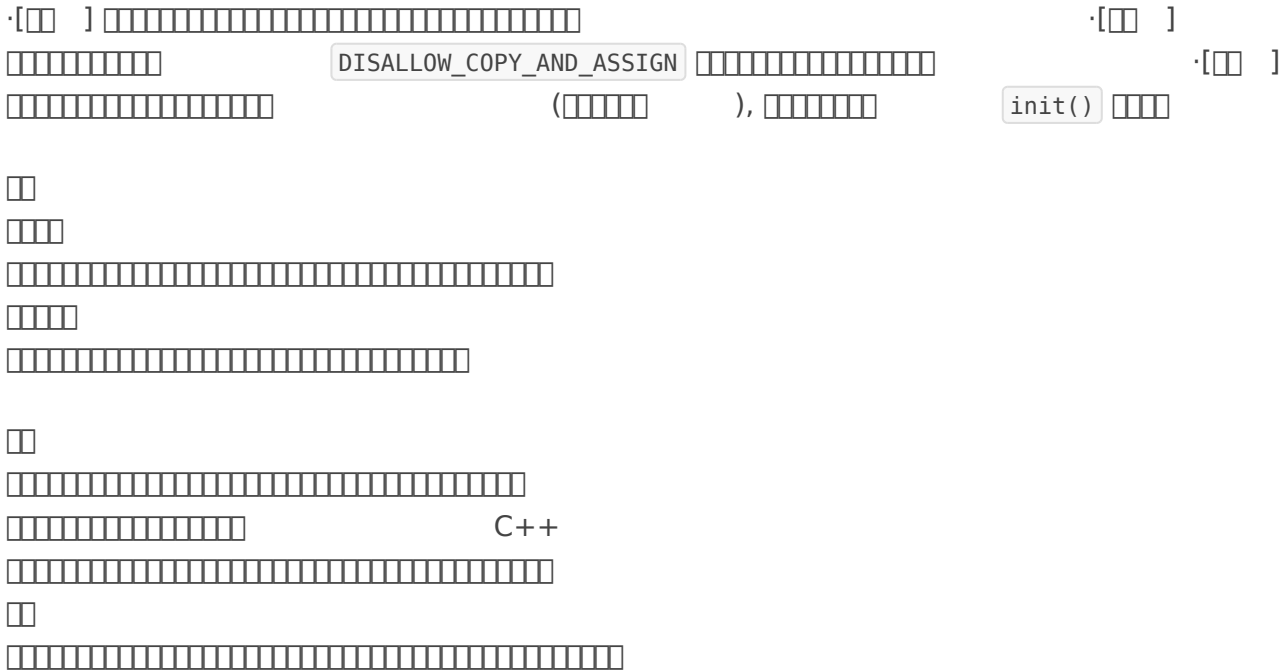
```
CheckOptions:
```

```
- { 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 }  
- { key: readability-function-cognitive-complexity.IgnoreMacros, value: 1 }
```



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

2.1.2. ???????



```
class Person {
public:
    // 显式转换
    explicit Person(const std::string& name) : _name(name) {} // 防止隐式转换
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);
    ~Foo();
```



```

    _str(_S_EMPTY_C_STR) {} // []_S_EMPTY_C_STR[]

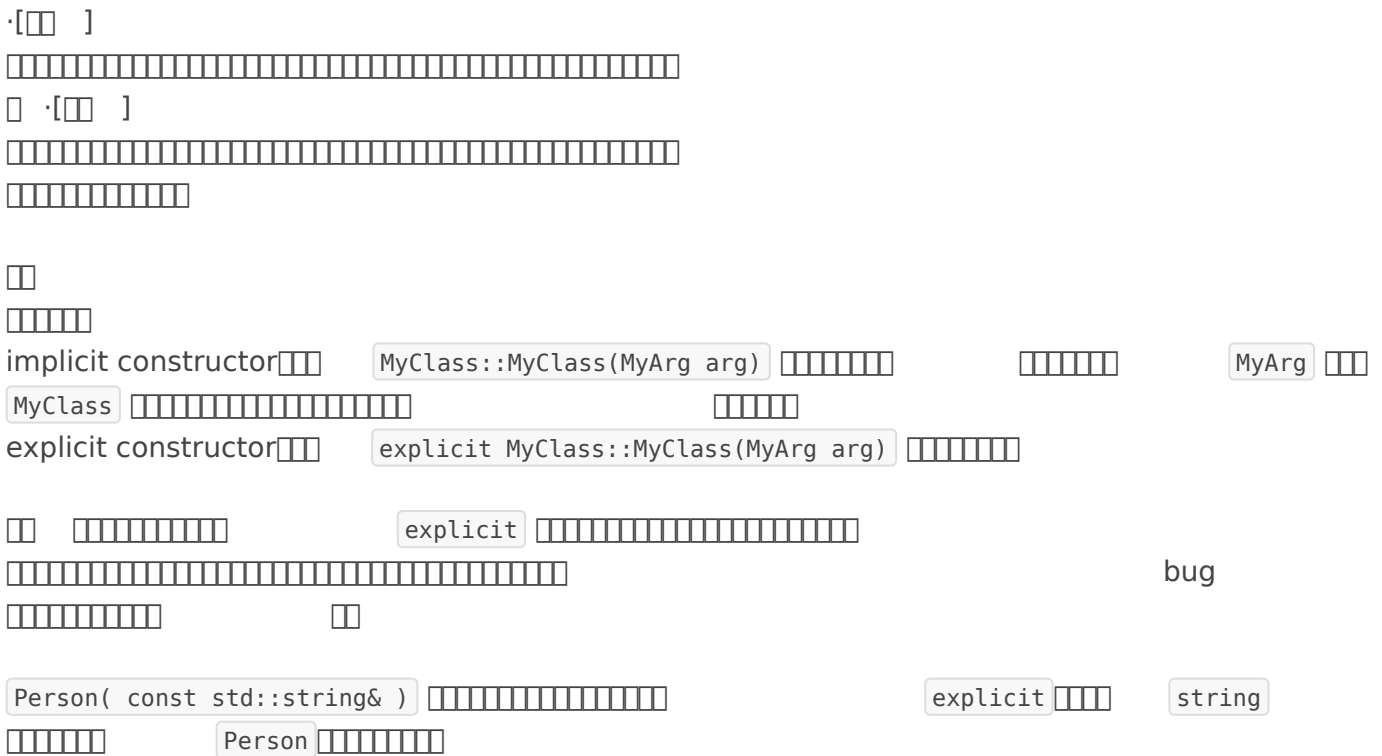
~String() {
    if (_str != _S_EMPTY_C_STR) {
        free(_str);
    }
}

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

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

```

2.1.4. ???????



```

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

```


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. ????????

·[]

□

```

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

```

□

```


```

□

```


```

```

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

```


2.1.10. ???????

```

class Base {
public:
    Base() {}
};

class Derived {
protected:
    Base base;
public:
    Derived() {}
};

class Base {
public/protected:
    Base() {}
};

class Derived {
private:
    Base base;
public/protected:
    Base base;
getter/setter:
    Base base;
const:
    Base base;
getter/setter:
    Base base;
};
    
```

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

2.1.11. ???????

```

class Base {
public/protected/private:
    Base() {}
};

class Derived {
private:
    Base base;
};
    
```

C++

```

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

```

// 初始化
void attach(void * p_object, DestructorFunc destructor); // 初始化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

```

class Dict {
public:
    Dict();
    Dict(const Dict&);
    Dict& operator=(const Dict&);
    ~Dict();
};

```

```

class DictIterator {
public:
    DictIterator(const Dict&);
    DictIterator(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:
    ...
};

```



```

·[ ] 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; // a.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;
}

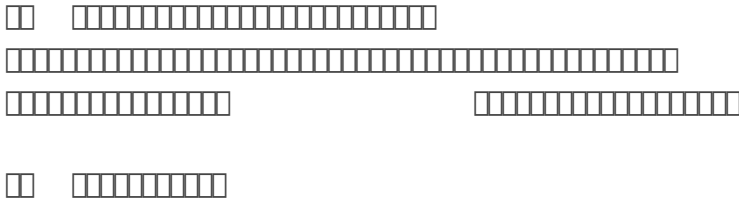
```

2.2.2. ????

·[] [RULE025] □□□□□

·[] □□□□□□□□□□□□

([] 10 [])



```

class Person {
public:
    std::string get_name() const { // 内存地址
        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++() {
            ...

```


□□ □□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□

□□ /
if □□□□ == □□□□ = □□□□
== □□ != □□□□□□□□□□

□□

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

2.3.2. NULL, nullptr?

·[□□]□□□□ 0□ 0.0□ NULL□ '\0'□□□□□□□□□□□□□□□□□□ 0□ ·[□□]□□□□ C++11
□□□□□□□□□□□□□□□□□□ nullptr □□□□□□□□

□□ □□□□□□□□□□□□□□□□

□□

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

2.3.3. sizeof ???

·[] sizeof() sizeof()
 sizeof() 32/64
 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++ C
```

2.3.5. goto

goto

goto

2.3.6. ?

```
DISALLOW_COPY_AND_ASSIGN, CFATAL_LOG etc.)  
do...while(0)  
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


```
enum { ... }
```

2.3.9. ????

```
enum { ... } [RULE050] ... / ... [RULE051] ...
enum { ... } const ... define { ... } enum ...
... [ ... ]
... [ ... ]
... XXX_COUNT( ... ) [ ... ]
... [ ... ]
```

```
define ... / ... define
...
... 8 ... strlen("filename")
... file_name
... [ ... ]
```

```
{ ... }
```

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

```
const { ... }
```

```
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_cast { ... } const { ... } const
... [ ... ]
```

const C C const
 const_cast C C const
 const 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; //
  
```

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

2.3.11. ???????

[] [] 4KB []

(variable-length array): , GCC

crash crash

3. ??/??

3.1. ??????

3.1.1. ????????

[] [] .cpp (), C() C++()
 [] .h

[] cpp [] [] ; [] [] ;

[] [] , some-project/foo/internal/fooserver.cc [] :

```
#include "foo/public/fooserver.h" // 0000
#include <sys/types.h>
#include <unistd.h>
#include <hash_map>
#include <vector>
#include "base/basicctypes.h"
#include "base/commandlineflags.h"
#include "foo/public/bar.h"
```

3.1.2. #include<>?#include""

·[] [] #include <>[] #include ""

3.1.3. #include????????????

·[] [] include[]

[] #include []

[]

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

3.1.4. ??include guards

·[] [] [RULE060] [] include guards ·[] [] [RULE061] include guards[]
 <SVNPATH>_<FILE>_H[HPP] [] SVNPATH [] /[] : 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)  
//  
// <>
```

3.2.2. ????

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

```
// mylib/AwesomeClass.h  
namespace mylib {
```



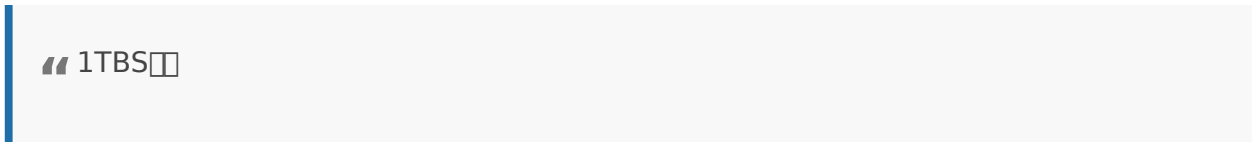
```

if (you.come_from(EARTH)) {
    return "hello";           // 000000{}000
} else if (you.come_from(MARS)) {
    return "@%!%#@!~&%$$";   // 000
} else {
    // 000000000000000000000000
    throw YouCheatMeException() << BSL_EARG << "your name:" << you.name(); // 00000
}

// Demo 00000000
if (you.com_from(EARTH)
    || you.com_from(MARS)
    || you.com_from(VENUS)) {
    return "you come from solar system";
}

```

00



3.2.3.8. switch??

```

.[[] ] switch []
[]
default []
switch []

[] case [] break [] return [] .[[] ]
[] case [] .[[] ] switch []
[] //pass [] //do nothing .[[] ] switch [] case
[] switch []

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

[]

```

```

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

```


3.2.4. ??????

```
·[ ] [RULE085] if/switch/while/for/catch
[ ] [RULE086] for
[ ] [RULE087]
·[ ] [RULE088]
·[ ] [RULE089]
·[ ] [RULE090]
·[ ] [RULE091]
a.b, a->b, a.*b, a->*b, a::b [ ] [RULE092]
```

[]

```
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) // [ ], [ ]
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); // [ ], [ ]

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
[]
[] []
[]
[] [] 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(), // []
    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. ???? ?

·[] const const
·[]

[] [] [] []

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

//
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);

//
```


_evaluator

[FALL 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] , _ ; :this_is_my_awesome_file.cpp ·[] <>_test.h(.cpp) ·[] .h <>_<>.h api include api mylib.h,mylib_utils.h,mylib_api.h

3.3.3. ???????

·[] [RULE102] []
[]

·[] [RULE103]
·[] []

[] []
namespace []
[]

using

3.3.4. ????

·[] [RULE103] [] ·[] [] ·[] []
[] accessor [] : 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]
·[] []

Interface []

[] [] enum [] struct [] class [] typedef [] template []

[]

```
enum Color {
    RED,
    BLUE,
    GREEN,
```


·[] [RULE122] []

[] []

[]

3.5.2. ???????

·[] []

[]

·[] []

[RULE123] []

[] []

[]

[]

Effective C++, item 26: Postpone variable definitions as long as possible

3.6. ????????

·[] [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 DWORD HANDLE [] C++ []
include guard #pragma once

[]

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

4.4. ??????"?"??

·[] " " []