

# C & C++ 入门

## 1. 环境

可以参考 Google C++ Style Guide 以及 C++ 社区的一些资源, 比如 [shiborag@gmail.com](mailto:shiborag@gmail.com), 等等。

## 2. 基础

本文主要介绍 C++ 的一些基础知识。

### 2.1. 数据类型

C++ 支持多种数据类型, 包括 Object-Oriented C++ 等等。

#### 2.1.1. struct 与 class 的区别

在 C++ 中, `struct` 和 `class` 都是用于定义数据结构的, 它们之间的区别主要在于:

在 C++ 中, `struct` 和 `class` 都是用于定义数据结构的, 它们之间的区别主要在于:

```
// 定义结构体
struct Coordinate {
    int x;
    int y;
    int z;
};

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

```
};
```

## 2.1.2. 静态成员函数

·[ ] 静态成员函数 `DISALLOW_COPY_AND_ASSIGN` 静态成员函数 `init()` 静态成员函数 `init()` 静态成员函数 `init()`, 静态成员函数

[ ]

[ ]

[ ]

[ ]

[ ]

[ ]

[ ] C++ [ ]

[ ]

[ ]

```
class Person {
public:
    // 静态成员函数 explicit
    explicit Person(const std::string& name) : _name(name) {} // 静态成员函数
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);
};
```

[ ]

Effective C++, item 04: Make sure that objects are initialized before they're used More Effective C++, Item 04: Avoid gratuitous default constructors [ ] 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

explicit constructor explicit MyClass::MyClass(MyArg arg)

[ explicit ] bug

Person( const std::string& ) [ explicit string ] Person

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

string C

```
class String {
public:
    String(const char* cs); //
private:
    ...
};
```

·

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

## 2.1.5. 用户自定义字面量



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. 点运算符

点运算符的函数原型如下：

点运算符

点运算符

assignment operator=[ MyClass& MyClass::operator=(const MyClass&) 点运算符 点运算符

点运算符

点运算符 点运算符

点运算符

点运算符

```
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) { //点运算符
        _x = other._x;
        _y = other._y;
        return *this;
    }
private:
    int _x;
    int _y;
}
```

点运算符

```
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;
```

[illegible]

- 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. $\square$

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

[illegible]





·[ ] [ ]

[ ] [ ]

[ ]

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();
    }

    // [ ]
```

```

void attach(void * p_object, DestructorFunc destructor); // C[buffer]
...
// attach

template <typename T>
T& create(); // 
...
// create
...
// [create_raw(),clone_cstring(),reset()]
private:
// private
// 
struct ObjectInfo; // 
...
// 
void push_info(); // [push_info()pop_info()]
void pop_info(); // [private]
...
// 
ObjectInfo *_p_object_info_list;
...
DISALLOW_COPY_AND_ASSIGN(ResourcePool); // 
};

```

## 2.1.12. [·[ ] ]

[ ] [ ]

[ ] [ ] operator ==( const Dict&, const Dict& ) [ ]

```

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

class Dictlterator {
    ...
};

```

·[ ] catch [ ]/[ ] [ ]

`int`: `initialize()` C++ Python, Java C+

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

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

[illegible]

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

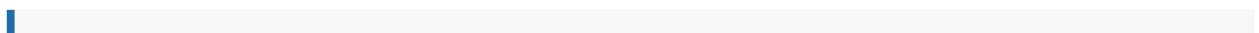
inline

```
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. 

--	--	--	--

[illegible][illegible]

The image shows two groups of base ten blocks. The first group consists of two vertical rods, each representing 10 units, for a total of 20 units. The second group consists of ten individual small cubes, each representing 1 unit, for a total of 10 units. Together, they represent 30 units.

```
class Iterator; // [ ] [ ] [ ] [ ]

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

11

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

2.3.8. //[illegible]

□□ □□□□□□□□□□□□□□□□ □□□□□□, □□□□□□□□□□□□□□ C++ □□□□□□, □□□□□□□□□□□□, [

2.3.9. 

--	--	--	--

```
·[ ] [RULE050] [ ]/[ ]·[ ] [RULE051] [ ] enum [ ] const [ ] define·[ ] [ ] (
```

`#define` `strlen("filename")` `file_name` `#define`

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

11

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

2.3.11. 

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

[illegible]

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

crash crash

3. ☐ ☐ ☒ ☐

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

### 3.1.1. `#include "foo.h"`

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

[1] `foo.cpp` [1] [1]; [1] [1];

[1] [1], `some-project/foo/internal/fooserver.cc` [1]:

```
#include "foo/public/fooserver.h" // [1]
#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 <foo.h>`

·[1] [1]`#include <foo.h>`

### 3.1.3. `#include <foo.h>`

·[1] [1]`#include <foo.h>`

[1] `#include <foo.h>`

[1]

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

### 3.1.4. `#include guards`

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

[1] `#include guards` [1] `#ifndef` [1] `#define` [1] `#endif` [1]

[1] [1] `#include guards` [1] svn [1] `#include guards` [1]; [1]

// [1], svn `https://svn.saick.net/ps/se/trunk/ac/strategy/ StrategyQueue/default/time_open_cluster_data.h` [1]:

```
#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. 头文件

### 3.2.1. 头文件结构

·[头文件] 头文件结构

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

头文件头文件头文件头文件

头文件头文件

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

### 3.2.2. 头文件

·[头文件] [RULE064] 头文件头文件头文件 ·[头文件] 头文件头文件头文件 ·[头文件] [RULE066] 头文件头文件 ·[头文件] 头文件头文件头文件头文件

头文件头文件头文件头文件头文件头文件头文件头文件头文件头文件头文件头文件

头文件

```
// 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. 访问控制

·[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. 初始化

·[命名空间] 初始化函数，命名空间，命名空间：命名空间。命名空间 4命名空间 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 8命名空间 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";
}
```

--	--

“1TBS”

### 3.2.3.8. switch

```
[ ] switch case [ ] break [ ] return [ ] [ ] [ ] [ ] [ ] [ ] default { //pass } //do nothing switch switch case [ ] [ ]  
switch [ ] [ ]
```

[illegible]

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

```
switch (cur_char) {
    case '\0':
    case '\': {
        return "it's a string!";
    }
    default:
        printf("not found yet");
        break;
}
```

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

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

3.2.3.9. 

--	--	--	--

[ ] [RULE083] do-while while E084

--	--	--	--

```
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) // 1
if(is_good && is_powerful) // 1 if(111111)
if ( is_good && is_powerful ) // 1, 11111111
switch (type) // 1
while (condition) // 1
catch (std::exception& ex) // 1
for (int i = 0; i < 10; ++i) // 1
for (int i = 0;i<10;++i) // 1, 111111

```

```
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 : 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, 异常 .hpp 异常

异常 异常信息

## 3.2.7. 异常

·[文件] 异常return异常

异常 return 异常 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 const 异常

异常 异常 异常 异常 异常

异常

```
/*
C/C++ 异常, 异常, 异常/异常.
异常 ``const`` 异常. 异常+异常const异常
异常(异常), 异常 ``const`` 异常
异常, 异常: 异常, 异常. 异常/异常 (异常/异常) 异常, 异常, 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] [☐] <☐
```

### 3.3.3.

$\cdot[\square] [\text{RULE102}] \begin{array}{|c|} \hline \square \\ \hline \end{array}$

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. 

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

[illegible]

Diagram illustrating a 1D array structure. The array is divided into segments, with the first segment labeled `cur_user_index` and subsequent segments labeled `i,j,k`.

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>
```

[illegible]

### 3.4.

3.4.1. 

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

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

[illegible]3.4.2. 

--	--	--	--





### 4.3.2.1. 花括号

·[C] [WCPP006] { } 花括号成对出现 ·[C] [WCPP005] 花括号成对出现

花括号

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

·[C] 在 int x = 3 & (4 < 32) 花括号, 花括号

### 4.3.3. 花括号WINAPI VS 花括号IDE

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

花括号

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

## 4.4. 花括号"花括号"花括号

·[C] 花括号"花括号"花括号 花括号

Revision #5

Created 17 August 2024 14:48:26 by Danny

Updated 15 September 2024 12:13:05 by Danny