金山研究院文件

C / C++ 源 代 码

书 写 规 范

金山软件股份有限公司
2004 年 4 月

目录

前言

本规范规定了 C/C++ 源代码的书写规范。

制定本规范的目的是统一研发部的 C/C++ 语言源代码的书写风格，以便于代码的阅读、维护、管理及修订。

本规范于 2000 年 4 月由研发部各部门代表（董波、沈家正、王羽、许彬、万里、林剑锋、李洪义、卢新冬、陈飞舟、王炜、赵青、张宏资）讨论制定，并于同月实施。从实施之日起，研发部所有新增及修订的 C/C++ 语言源代码都必要遵从本规范进行书写。在条件允许的情况下，研发部各成员应尽量遵从本规范修订在该日期之前已有的源代码。

需要对本规范进行修订增删时，须由研发部经理组织相关人员集体讨论，先达成共识，再进行修改。

本规范起草人：董波、沈家正、王羽、许彬、万里、林剑锋、李洪义、卢新冬、陈飞舟、王炜、赵青、张宏资。

本规范执笔人：袁岚

修订记录：

2004 年 4 月。参与人：万里、朱传靖、陈飞舟、孙国军、胡翌、李宇雄、吴越、董波

执笔：李宇雄

C/C++源代码书写规范

1．文件起始处的说明

在.h/.cpp 的开头应有一段格式统一的说明，内容包括：

- 文件名 (FileName)；
- 创建人 (Creator)；
- 文件创建时间 (Date)；
- 简短说明文件功能、用途 (Comment)。

例：

```
//////////////////////////////////////////////////////////////
//
// FileName  :  KSample.h
// Creator   :  John Doe
// Date    :  2004-2-4 21:42:54
// Comment   :
//
//////////////////////////////////////////////////////////////
```

2．关于注释

除非极其简单，否则对函数应有注释说明。内容包括：功能、入口/出口参数，必要时还可有备注或补充说明。

2.1 推荐采取特定的格式(见第 14 条),以方便用软件自动生成帮助文档。

2.2 对于 if、while、do 等其大括号内嵌代码块比较长(需拖动滚动条来看)或者多层嵌套时，在结尾的“}”后要加上其所对应的语句的注释说明。

例：

```
if ((isWindow == TRUE) && (isVisible == true))
{
  代码段
}// if ((isWindow == TRUE)...
```

2.3 函数入口参数有缺省值时，应注释说明。

例：

```
BOOL KSSaveToFile(
  const char cszFileName[],
  BOOL bCanReplace /* = TRUE */
);
```

或者：

```
BOOL KSSaveToFile(
  const char cszFileName[],
  BOOL bCanReplace // = TRUE
);
```

2.3 描述某个代码段的注释，可以给注释描述的代码段外围加上{}，帮助阅读。

例：

```
//代码段实现功能或意图描述
{
```

```
  代码段
}
```

3．每行代码长度

每行代码的长度推荐为 80 个 ASCII 字符，最长不得超过 120；折行以对齐为准。

例：

```
HANDLE KSOpenFile(const char cszFileName[],
        int nMode);
```

或者：

```
BOOL KSReadFile(
  HANDLE hFile,
  void *pvBuffer,
  int nReadSize,
  int *pnReadSize
);
```

4．合并行的问题

循环、分支代码，判断条件与执行代码不得在同一行上。

例：

正确：

```
 if (n == -2)
   n = 1;
 else
   n = 2;
```

不得写做：

```
 if (n == -2) n = 1;
 else n = 2;
```

5．指针中*号的位置

指针的定义，* 号既可以紧接类型，也可以在变量名之前。

例：

```
可写做：      int*  pnsize;
也可写做：    int  *pnsize;
但不得写做：  int * pnsize;
```

6．全局函数的调用

在类的成员函数内调用全局函数时，在全局函数名前必须加上“::”。

7．关于 if...else if

else if 必须写在一行。

8．与“{”、“}”有关的各项规定

8.1 在“{”之前不允许有除空格与 Tab 之外的其他任何字符；在它之后可有注释，但不允许有代码。与“{”对应的“}”必须在同一列上。

例：

正确：

```
for (i = 0; i < cbLine; i++)
```

```
{  // .....
  printf("Line %d:", i);
  printf("%s\n", pFileLines[i]);
}
```

不得写做：

```
for (i = 0; i < cb; i++)
{  printf("Line %d:", i);
  printf("%s\n", pFileLines[i]);
}
```

也不得写做：

```
for (i = 0; i < cb; i++){
  printf("Line %d:", i);
  printf("%s\n", pFileLines[i]);
}
```

8.2 在循环、分支之后若只有一行代码，在无歧义的情况下可省略“{”、“}”。

例：

```
if (n == -2)
  n = 1;
else
  n = 2;
```

9．与空格有关的各项规定

9.1 在所有两目、三目运算符的两边都必须有空格。在单目运算符两端不必空格。但在“->”、“::”、“.”、“[”、“]”等运算符前后，及“&”（取地址）、“*”（取值）等运算符之后不得有空格。

例：

正确：

```
int n = 0, nTemp;
for (int i = nMinLine; i <= nMaxLine; i++)
```

不得写做：

```
int n=0, nTemp;
for ( int i=nMinLine; i<=nMaxLine; i++ )
```

9.2 for、while、if 等关键字之后应有 1 个空格，再接“(”。

例：

正确：

```
if (-2 == n)
```

不得写做：

```
if(-2 == n)
```

9.3 调用函数、宏时，“(”前不得有空格。

例：

正确：

```
printf("%d\n", nIndex);
```

不得写做：

```
printf ("%d\n", nIndex);
```

9.4 类型强制转换时，“(”“)”前后不得有空格

例：

可写做：

```
(KSFile*)pFile;
```

也可写做：

```
(KSFile *)pFile
```

不得写做：

```
( KSFile* )pFile
( KSFile * ) pFile
```

10．与缩进有关的各项规定

10.1 缩进以 Tab 为单位。要求在编辑器中将 1 个 Tab 设置为 4 个空格宽度。

10.2 下列情况，代码缩进一个 Tab：

1．函数体相对函数名及“{”、“}”。

例：

```
int Power(int x)
{
  return (x * x);
}
```

2．if、else、for、while、do 等之后的代码。

3．一行之内写不下，折行之后的代码，应在合理的位置进行折行。若有 + - * / 等运算符，则运算符应在上一行末尾，而不应在下一行的行首。

10.3 下列情况，不必缩进：switch 之后的 case、default。

例：

```
switch (nID)
{
case ID_PLAY:
  ......
  break;
case ID_STOP:
  ......
  break;
default:
  ......
  break;
}
```

11．关于出错处理

本规范对此不做特别规定。但要求项目组在开始编码前必须为项目定义明确的出错处理规范。

12．与类相关的.h 文件与.cpp 文件

12.1 原则上，应该在一个单独的.h 文件中定义一个类，在一个单独的.cpp 文件中实现这个类。.h 与.cpp 文件的文件名必须与类名相同。

12.2 若几个类的规模都不大，关系又很密切，则可在一个.h 文件中定义这些类，在一个.cpp 文件中实现。对于附属于较大规模类的一个很小规模的类，可以写在那个大规模类的.h 和.cpp 里。

13．注释书写与自动生成帮助文档规范

13.1 以下的注释规范用于自动生成帮助文档，自动生成文档工具为 Doxygen。对于那些不希望提取为帮助文档的注释，不在本规范规定的范围内。

13.2 块注释

[格式]

开头：/**

结尾：*/

例：

```
/**
* ... This is a block comment ...
*/
```

13.3 行注释

[格式]

开头：///

例：

```
/// This is a line comment...
```

13.4 函数的摘要

[格式]

在函数声明主体开头的注释块中加上控制符：@brief

例：

```
/**
* @brief 用于从输入流读取一个字符，不回显
*/
int getch(void);
```

13.5 函数的形参

[格式]

在函数声明主体开头的注释块中加上控制符：@param

例：

```
/**
* @param buf1 缓冲区 1
* @param buf2 缓冲区 2
* @param count 字符的个数
*/
int memcmp(const void *buf1, const void *buf2, size_t count);
```

13.6 函数的返回值

[格式]

在函数声明主体开头的注释块中加上控制符：@return

例 1：

```
/**
* @return 读取的字符
*/
int getchar(void);
```

例 2：

```
/**
* @return None
*/
void rewind(FILE *stream);
```

13.7 函数的注意事项（Remark）

[格式]

在函数声明主体开头的注释块中加上控制符：@remark

例：

```
/**
* @remark 拷贝 strSource 到 strDestination（包含 strSource 中的'\0'），
 要注意在拷贝的过程中并没有溢出检查
*/
char *strcpy(char *strDestination, const char *strSource);
```

13.8 补充说明

更多的格式请参考 Doxygen 文档，因为有些高级的选项比较繁琐。编写注释的目的是为了方便自动生成文档，而在此基础上又不能使程序员过于分散精力。Doxygen 的文档自动生成功能非常强大，很多事情都不需要手工干预，我们只要用到最基本的几项功能也可以满足需求了。

附录一　命名规范

本规范给出了工程、文件、函数、变量、类/结构、宏、常量、枚举、联合等的命名规范。

通则：

1．所有命名都应使用标准的英文单词或缩写，不得使用拼音或拼音缩写，除非该名字描述的是中文特有的内容，如半角、全角，声母、韵母等。
2．所有命名都应遵循达意原则，即名称应含义清晰、明确。
3．所有命名都不易过长，应控制在规定的最大长度以内。
4．所有命名都应尽量使用全称。
5．如果命名使用缩写，则应该使用《通用缩写表》(见附录二)中的缩写；原则上不推荐使用《通用缩写表》以外的缩写，如果使用，则必须对其进行注释和说明。

具体规范：

1．工程名：
不强制统一。
2．文件名：
- 只能由英文字母、数字和下划线组成文件名。
- 区分大小写,基于跨平台的考虑,建议全小写。
- 长度不限于 8.3 格式，建议不多于 30 个字符。
- 如果文件用于定义或实现类，则文件名与类名必须保持一致。

3．函数名：
- 参照 Windows API 的命名规范。
- 函数名最长不得超过 64 个字符。
- 推荐使用动宾结构。函数名应清晰反映函数的功能、用途。
- 推荐函数名第一个字母大写。

4．变量名：
原则上，变量的命名遵从匈牙利记法。即：前缀 + 类型 + 名称。最终的变量名总长不得超过 32 个英文字符。
1）格式：
```
[m_|s_|g_] type [class name|struct name] variable name
```
2）解释：
- m_ ：类的成员变量
- ms_：类的静态成员变量

- s_ ：静态全局变量
- g_ ：普通全局变量
- 类型缩写（type）
  - char, TCHAR： c/ch
  - 字符串： s/sz/str
  - bool, BOOL： b
  - int, __int16,__int32,__int64： n
  - unsigned： u
  - long： l
  - unsigned long: ul
  - double， float： f
  - BYTE： by
  - WORD： w
  - DWORD： dw
  - function： fn
  - pointer: p

  小范围局部变量与结构成员可省略类型缩写

5．类名：

- 必须以大写"K"开头，后面字母反映具体含义，以清晰表达类的用途和功能为原则。
- 接口必须以大写"I"开头，代表 Interface 。
- 当名称由多个单词构成时，每一个单词的第一个字母必须大写。

6．结构、宏、枚举以及联合的名字必须全部大写。

附录二 通用缩写表

说明：

1、本缩写表中列出的都是通用性缩写，不提供标准缩写，如：Win9x、COM 等。

2、使用本缩写表里的缩写时，请对其进行必要的注释说明。

3、除少数情况以外，大部分缩写与大小写无关。

通用缩写表

#	缩写	全称
1	addr	Address
2	adm	Administrator
3	app	Application
4	arg	Argument
5	asm	assemble
6	asyn	asynchronization
7	avg	average
8	DB	Database
9	bk	back
10	bmp	Bitmap
11	btn	Button
12	buf	Buffer
13	calc	Calculate
14	char	Character
15	chg	Change
16	clk	Click
17	clr	color
18	cmd	Command
19	cmp	Compare
20	col	Column
21	coord	coordinates
22	cpy	copy
23	ctl / ctrl	Control
24	cur	Current
25	cyl	Cylinder
26	dbg	Debug
27	dbl	Double
28	dec	Decrease
29	def	default
30	del	Delete
31	dest / dst	Destination
32	dev	Device
33	dict	dictionary
34	diff	different
35	dir	directory

#	缩写	全称
36	disp	Display
37	div	Divide
38	dlg	Dialog
39	doc	Document
40	drv	Driver
41	dyna	Dynamic
42	env	Environment
43	err	error
44	ex/ext	Extend
45	exec	execute
46	flg	flag
47	frm	Frame
48	func / fn	Function
49	grp	group
50	horz	Horizontal
51	idx / ndx	Index
52	img	Image
53	impl	Implement
54	inc	Increase
55	info	Information
56	init	Initial/Initialize/Initialization
57	ins	Insert
58	inst	Instance
59	INT / intr	Interrupt
60	len	Length
61	lib	Library
62	lnk	Link
63	log	logical
64	lst	List
65	max	maximum
66	mem	Memory
67	mgr / man	Manage / Manager
68	mid	middle
69	min	minimum
70	msg	Message
71	mul	Multiply
72	num	Number
73	obj	Object
74	ofs	Offset
75	org	Origin / Original
76	param	Parameter
77	pic	picture
78	pkg	package
79	pkt	packect
80	pnt / pt	Point
81	pos	Position
82	pre / prev	previous
83	prg	program

#	缩写	全称
84	prn	Print
85	proc	Process / Procedure
86	prop	Properties
87	psw	Password
88	ptr	Pointer
89	pub	Public
90	rc	rect
91	ref	Reference
92	reg	Register
93	req	request
94	res	Resource
95	ret	return
96	rgn	region
97	scr	screen
98	sec	Second
99	seg	Segment
100	sel	Select
101	src	Source
102	std	Standard
103	stg	Storage
104	stm	Stream
105	str	String
106	sub	Subtract
107	sum	summation
108	svr	Server
109	sync	Synchronization
110	sys	System
111	tbl	Table
112	temp / tmp	Temporary
113	tran / trans	translate/transation/transparent
114	tst	Test
115	txt	text
116	unk	Unknown
117	upd	Update
118	upg	Upgrade
119	util	Utility
120	var	Variable
121	ver	Version
122	vert	Vertical
123	vir	Virus
124	wnd	Window