```cpp
// 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{
    nt j = g()
  }
  switch(var){
  case 0:{
    ...
    break;
    }
    default:{
      assert(false);
    }
  }
  return x;
}

} // namespace mynamespace
```

版权
许可证，BSD/GPL/MIT/...
作者
文件内容简短说明

实现文件扩展名.cc
文件名全小写,可包含下划线或短线

包含次序标准化避免隐藏依赖：
1 本类的声明(第一个包含本类h文件，有效减少依赖)
2 C系统文件
3 C++系统文件
4 其他库头文件
5 本项目内头文件(避免使用UNIX文件路径"."和"..")

可以在整个cc文件和h文件的方法内使用using
禁止使用using namespace xx污染命名空间

多行初始化列表，":"前4空格缩进，以","结尾
多个变量折行对齐
单行初始化列表 Class::Class():_var(xx){
构造函数中只进行那些没有实际意义的初始化

参数过多时","结尾
每行一个变量对齐

参数过多时","结尾

条件括号内无空格，(condition)左右1空格，if执行体2空格缩进

前置自增运算
条件变量过多时，条件运算符 && 结尾
条件左对齐

临时方案使用TODO(大写)注释，
后面括号里加上你的大名、邮件地址等

大括号与else同行，else左右1空格
尽量使用初始化时声明

(var)左右各1空格
条件相对switch 2空格缩进
执行体相对switch 4空格缩进

若default永不执行可使用assert

返回值不需要加括号

尽量不使用宏
不使用异常
禁止使用RTTI
使用printf之类的代替流
除位域不使用无符号数字
除特定环境，不使用操作符重载
使用4种cast运算符类型转换
禁止使用Class类型全局变量
若使用必须为单例模式
sizeof(var)代替sizeof(type)
scoped_ptr可以胜任智能指针
特殊情况下可用shared_ptr
任何时候都不使用auto_ptr

命名空间结束注释

```cpp
// 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_
```

版权
许可证，BSD/GPL/MIT/...
作者
文件内容简短说明

头文件扩展名.h
文件名全小写,可包含下划线或短线

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名命名空间

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

访问限定符1空格缩进

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，防止隐式类型转换误用
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

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
不为参数设置缺省值

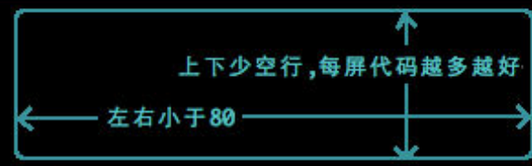
存取函数命名，取:同变量名，存:值函数名为set_varname
短小的存取函数可用内联

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



保护宏结尾加注释

Google C++ Style Guide