**C语言编码规范**

**一、空行**

空行起着分隔程序段落的作用。空行得体将使程序的布局更加清晰。

1. 定义变量后要空行。尽可能在定义变量的同时初始化该变量，即遵循就近原则。如果变量的引用和定义相隔比较远，那么变量的初始化就很容易被忘记。若引用了未被初始化的变量，就会导致程序出错。
2. 每个函数定义结束之后都要加空行。
3. 两个相对独立的程序块、变量说明之后必须要加空行。比如上面几行代码完成的是一个功能，下面几行代码完成的是另一个功能，那么它们中间就要加空行。这样看起来更清晰。

**二、空格**

1. 关键字之后要留空格。像 const、case 等关键字之后至少要留一个空格，否则无法辨析关键字。像 if、for、while 等关键字之后应留一个空格再跟左括号(，以突出关键字。
2. 函数名之后不要留空格，应紧跟左括号(，以与关键字区别。
3. (向后紧跟；)、,、;这三个向前紧跟；紧跟处不留空格。
4. ,之后要留空格。如果;不是一行的结束符号，其后要留空格。
5. 赋值运算符、关系运算符、算术运算符、逻辑运算符、位运算符，如 =、==、！=、+=、－=、\*=、/=、%=、>>=、<<=、&=、^=、|=、>、<=、>、>=、+、－、\*、/、%、&、|、&&、||、<<、>>、^ 等双目运算符的前后应当加空格。

* 注意，运算符“%”是求余运算符，与 printf 中 %d 的“%”不同，所以 %d 中的“%”前后不用加空格。

1. 单目运算符 !、~、++、－－、－、\*、& 等前后不加空格。

* 注意：这里的“－”和规则5）里面的“－”不同。这里的“－”是负号运算符，规则5）里面的“－”是减法运算符。
* 这里的“\*”和规则5）里面的“\*”也不同。这里的“\*”是指针运算符，规则5）里面的“\*”是乘法运算符。
* 这里的“&”和规则5）里面的“&”也不同。这里的“&”是取地址运算符，规则5）里面的“&”是按位与运算符。

总之，规则6）中的是单目运算符，而规则5）中的是双目运算符，它们是不一样的。

1. 像数组符号[]、结构体成员运算符.、指向结构体成员运算符->，这类操作符前后不加空格。
2. 对于表达式比较长的 for 语句和 if 语句，为了紧凑起见，可以适当地去掉一些空格。但 for 和 if 后面紧跟的空格不可以删，其后面的语句可以根据语句的长度适当地去掉一些空格。例如：

for (i=0; i<10; i++)

for 和分号后面保留空格就可以了，=和<前后的空格可去掉。

**三、成对书写**

成对的符号一定要成对书写，如 ()、{}。不要写完左括号然后写内容最后再补右括号，这样很容易漏掉右括号，尤其是写嵌套程序的时候。

**四、缩进**

缩进是通过键盘上的 Tab 键实现的，缩进可以使程序更有层次感。原则是：如果地位相等，则不需要缩进；如果属于某一个代码的内部代码就需要缩进。

**五、对齐**

对齐主要是针对大括号{}说的：

1. {和}分别都要独占一行。互为一对的{和}要位于同一列，并且与引用它们的语句左对齐。
2. {}之内的代码要向内缩进一个 Tab，且同一地位的要左对齐，地位不同的继续缩进。

还有需要注意的是，很多编程软件是会“自动对齐”的，比如：

#include <stdio.h>  
int main(void)  
{  
    if (…)  
    return 0;  
}

写完 if 那一行后，按回车，此时光标在括号的右边，而此 if 下的大括号要写在与 if 左对齐的正下方，通常我们是按一下 Backspace 键使光标停在与 if 左对齐的正下方。但事实上我们不需要这样做，我们直接输入大括号即可，系统会自动对齐到与 if 左对齐的正下方。再接着看：

#include <stdio.h>  
int main(void)  
{  
    if (…)  
    {  
        while (…)  
    }  
    return 0;  
}

写完 while 那一行后，按回车，此时光标不是停在与 while 左对齐的正下方，同样，我们不需要按 Backspace，直接输入大括号即可，系统会自动对齐到与 while 左对齐的正下方的。

此外编程软件还有“对齐、缩进修正”功能。就是按 Ctrl+A 全选，然后按 Alt+F8，这时程序中所有成对的大括号都会自动对齐，未缩进的也会自动缩进。不管是在编程过程中，还是在编写结束完之后，都可以使用这个技巧。但如果完全按照规范写，那根本就不需要这个技巧，所以，这只是一个辅助功能。

**六、代码行**

1. 一行代码只做一件事情，如只定义一个变量，或只写一条语句。这样的代码容易阅读，并且便于写注释。
2. if、else、for、while、do 等语句自占一行，执行语句不得紧跟其后。

* 此外，非常重要的一点是，不论执行语句有多少行，就算只有一行也要加{}，并且遵循对齐的原则，这样可以防止书写失误。

**七、注释**

C语言中一行注释一般采用//…，多行注释必须采用/\*…\*/。注释通常用于重要的代码行或段落提示。在一般情况下，源程序有效注释量必须在 20% 以上。虽然注释有助于理解代码，但注意不可过多地使用注释。

1. 注释是对代码的“提示”，而不是文档。程序中的注释不可喧宾夺主，注释太多会让人眼花缭乱。
2. 如果代码本来就是清楚的，则不必加注释。

例如：

i++;  //i加1

这个就是多余的注释。

1. 边写代码边注释，修改代码的同时要修改相应的注释，以保证注释与代码的一致性，不再有用的注释要删除。
2. 当代码比较长，特别是有多重嵌套的时候，应当在段落的结束处加注释，这样便于阅读。
3. 每一条宏定义的右边必须要有注释，说明其作用。

**八、排版规则**

1. 程序应采用缩进风格编写，每层缩进使用一个制表位（TAB），类定义、方法都应顶格书写；
2. 源程序建议使用英文书写，尽量不含有中文。每行不超过80字符。对于较长的语句（>80字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读；循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分；
3. 左花括号要另起一行，不能跟在上一行的行末；
4. 一个变量定义占一行，一个语句占一行；
5. 对独立的程序块之间、变量说明之后必须加空行；
6. 在结构成员赋值等情况，等号对齐，最少留一个空格；
7. 若函数或过程中的参数较长，则要进行适当的划分。
8. 形参的排序风格：

Ø 最常使用的参数放在第一位；

Ø 输入参数列表应放在输出参数列表的左边；

Ø 将通用的参数放在特殊的参数的左边。

**九、命名**

1. **应用程序的命名**

“系统简称”+模块名称

1. **子模块的命名**

每个子模块的名字应该由描述模块功能的1-3以单词组成。每个单词的首字母应大写。在这些单词中可以使用一些较通用的缩写。

1. **变量命名**

变量的命名的基本原则是使得变量的含义能够从名字中直接理解。可以用多个英文单词拼写而成，每个英文单词的首字母要大写，其中英文单词有缩写的可用缩写；变量的前缀表示该变量的类型；对于作用域跨越10行以上的变量名称不能少于4个字符，除循环变量，累加变量外不得使用I、j、k等名称的变量。变量分为取全局变量和局部变量，对于全局变量以加前缀“g\_”来区分。

使用有意义的英语单词,使用大小写分隔,每个单词的第一个字母为大写



考虑到习惯性和简洁性，对于按常规使用的局部变量允许采用极短的名字，如用n、i作为循环变量，p、q作为指针等。另外，要注意的是：全局变量在程序中不要定义太多，能用局部变量的就用局部变量。如果要使用相关的变量，建议采用类的方式或者结构的方式存放，以减少具体变量的个数。

1. **常量命名**

常量所有的字母均为大写。并且单词之间使用下划线”\_”隔开。

1. **函数/过程的命名**

函数/过程名称应该尽量使用能够表达函数功能的英文名称，函数名称中应该禁止使用如同function1,function2等含义不清的名称。单词间应该使用大小写分隔。全局函数/过程名称以“g\_”前缀开始。

1. **接口命名**

接口名称要以大写字母开头。如果接口包含多个单词，每个单词的首字母大写，其他字母小写，如果，这些单词是缩略语（例如XML），也要首字母大写，其他字母小写(写为Xml)。

1. **类的命名**

* 类名称要以大写字母开头；
* 类名称如果包含多个单词，每个单词的首字母要大写，其他字母小写；如果这些单词是缩略语（例如XML），也要首字母大写,其他字母小写（写作Xml）；
* 类名称应该是一个名词或名词短语；
* 类成员变量的命名规则与上述规则相同，但是要以“m\_”开始，表示其为成员变量（Member）；
* 类名称不能出现下划线。

1. **方法的命名**

* 方法名称以小写字母开头。
* 方法名称如果包含多个单词，除了第一个单词外，每个单词的首字母大写，其它字
* 母小写。如果这些单词是缩略语（例如XML），也要首字母大写，其它字母小写（写作Xml）。
* 方法名称应该是一个动词或动名词短语，意思是“完成什么功能”，“执行什么操作”。

**八、实际案例**

1、解一个一元二次方程

# include <stdio.h>

# include <math.h> /\*因为要用到求平方函数sqrt()，所以要包含头文件 math.h\*/

int main(void)

{

//把三个系数保存到计算机中

int a = 1; // “=”不表示相等，而是表示赋值

int b = 2;

int c = 1;

double delta; //delta存放的是b\*b - 4\*a\*c的值

double x1, x2; //分别用于存放一元二次方程的两个解

delta = b\*b - 4\*a\*c;

if (delta > 0)

{

x1 = (-b + sqrt(delta)) / (2\*a);

x2 = (-b - sqrt(delta)) / (2\*a);

printf("该一元二次方程有两个解，x1 = %f, x2 = %f\n", x1, x2);

}

else if (0 == delta)

{

x1 = (-b) / (2\*a);

x2 = x1; //左边值赋给右边

printf("该一元二次方程有一个唯一解，x1 = x2 = %f\n", x1);

}

else

{

printf("无解\n");

}

return 0;

}

2、判断101-200之间有多少个素数，并输出所有素数

#include "stdio.h"  
 #include "conio.h"  
 #include "math.h"  
 main()  
 {  
  int m,i,k,h=0,leap=1;  
  printf("\n");  
  for(m=101;m<=200;m++)  
  {  
    k=sqrt(m+1);  
    for(i=2;i<=k;i++)  
    if(m%i==0)  
    {  
      leap=0;  
      break;  
    }  
    if(leap)  
    {  
      printf("%-4d",m);  
      h++;  
      if(h%10==0)  
        printf("\n");  
    }  
    leap=1;  
  }  
  printf("\nThe total is %d",h);  
  getch();  
 }