数据结构与算法

C语言代码规范

王　昭
北京大学信息学院软件研究所

wangzhao@infosec.pku.edu.cn

程序的书写格式

- 应该特别注意程序的书写格式，让它的形式反映出其内在的意义结构。好的格式能使程序结构一目了然，帮助你和别人理解它，帮助你的思维，也帮助你发现程序中不正常的地方，使程序中的错误更容易被发现。
- 人们常用的格式形式是：逻辑上属于同一个层次的互相对齐；逻辑上属于内部层次的推到下一个对齐位置。
- 利用集成开发环境（**IDE**）或者其他程序编辑器的功能，可以很方便地维护好程序的良好格式。请注意下面这几个键，在写程序中应该经常用到它们：**Enter**键（换一行），**Tab**键（将输入光标移到下一个对齐位置——进入新的一个层次），**Backspace**键（回到前一个对齐位置——退到外面的一个层次）。

example

```
# include "stdio.h"
int gcd(int x,int y);

main()
{
        int a,b,c;
        scanf("%d,%d",&a,&b);
        c=gcd(a,b);
        printf("%d\n",c);
}

int gcd(int x,int y)
{
     int z;
    while (y)
     {
          z=x%y;x=y;y=z;
          printf("%d,%d\n",y,z);
     }
    return(x);
}
```


标识符名称

- 标识符名称包括函数名、常量名、变量名等。这些名字应该能反映它所代表的实际东西，具有一定的意义，使其能够见名知义，有助于对程序功能的理解。规则如下：
  - 所有宏定义、枚举常数和**const**常变量，用大写字母命名。 **#define ARRAY_SIZE 24**
  - 复合词中每个单词的第一个字母大写。也可以在复合词里可以用下划线隔开每个词。 **SeqList　locate_seq**
  - **typedef**定义的类型名用大写字母表示。

    typedef int INTEGER;
  - 通常，函数的命名也是以能表达函数的动作意义为原则的，一般是由动词打头，然后跟上表示动作对象的名词，各单词的首字母可以大写。 **createNullList_seq**
  - 循环变量可采用**i, j, k**等，不受上述规则限制
  - 对结构体内的变量命名，遵循变量的具体含义命名原则

数据和函数说明

- 数据说明次序应当规范化，使数据属性容易查找，也有利于测试、排错和维护。说明的先后次序应固定，应按逻辑功能排序，逻辑功能块内建议采用下列顺序：整型说明、实型说明、字符说明、逻辑量说明。
- 如果设计了一个复杂的数据结构，应当通过注释对其变量的含义、用途进行说明

程序注释

- 程序注释是程序员与日后的程序读者之间通信的重要手段之一，注释分为文件注释、函数注释和功能注释。正规程序的注释应注意：注释行的数量占到整个源程序的**1/3到1/2**。
- 文件注释位于整个源程序的最开始部分，注释后空两行开始程序正文。它包括：

——程序标题。

——目的、功能说明。

——文件作者、最后修改日期等说明。

文件注释

- 例：
  /*
  （空一行）
  标题: **merglist.c**
  功能: 归并两个有序表.
  说明:
- 归并两个数据元素按非递减有序排列的线性表**palist**和**pblist,**求得线性表**pclist**也具有同样的特性
  当前版本： **x.x**
  修改信息： **2004.08.05 Anni， Initial Version**
  2004.08.20 Tom， Bug xxxx fixed
  ***/**
  (空**2**行，开始程序正文**)**

函数注释

- 函数注释通常置于每函数或过程的开头部分，它应当给出函数或过程的整体说明，对于理解程序本身具有引导作用。一般包括如下条目：

——模块标题。

——有关本模块功能和目的的说明。

——调用格式

——接口说明：包括输入、输出、返回值、异常。

——算法。如果模块中采用了一些复杂的算法。

函数注释示例

- （注释开头与上一函数最后一行间隔两行）

/*
标题：delete_seq
功能：在palist所指顺序表中删除下标为p的元素
格式：int delete_seq(PSeqList palist,int p)
输入： palist所指顺序表, 下标p
输出： palist所指顺序表
返回值: TRUE正常, FALSE错误
***/**

(注释后直接开始程序正文，不空行。**)**

功能性注释

- 功能性注释嵌在源程序体中，用于描述其后的语句或程序段做什么工作，也就是解释下面要做什么，或是执行了下面的语句会怎么样。而不要解释下面怎么做，因为解释怎么做常常与程序本身是重复的。

  例：

  /*把 amount 加到 total中*/

  total = amount + total;

  这样的注释仅仅是重复了下面的程序，对于理解它的工作并没有什么作用。而下面的注释，有助于读者理解。

  /*将每月的销售额amount加到年销售额total中*/

  total = amount + total;

语句结构-1

- 为保证语句结构的清晰和程序的可读性，在编写软件程序时应注意以下几个方面的问题：
——在一行内只写一条语句，并采用空格、空行和移行保证清楚的视觉效果。
——每一个嵌套的函数块，使用一个**TAB**缩进（可以设定为**4**个空格），大括号必须放在条件语句的下一行，单独成一行，便于匹对：
——文件之中不得存在无规则的空行，比如说连续十个空行。
一般来讲函数与函数之间的空行为**2-3**行；
在函数体内部，在逻辑上独立的两个函数块可适当空行，一般为**1-2**行。
——程序编写首先应考虑清晰性，不要刻意追求技巧性而使得程序难以理解。

语句结构-2

- ——每行长度尽量避免超过屏幕宽度，应不超过**80**个字符。
——除非对效率有特殊要求，编写程序要作到清晰第一，效率第二。
——尽可能使用函数库。
——尽量用公共过程或子程序去代替重复的功能代码段。
——使用括号清晰地表达算术表达式和逻辑表达式的运算顺序。如将 **x=a*b/c*d** 写成 **x=(a*b/c)*d**可避免阅读者误解为**x=(a*b)/(c*d)**。
——避免采用过于复杂的条件测试。
——避免过多的循环嵌套和条件嵌套。
——建议不要使用 ***=**，**^=, /=**等运算符。

语句结构-3

- ——一个函数不要超过**200**行。一个文件应避免超过**2000**行。
——尽量避免使用**go to**语句。
——不鼓励采用**?:**操作符，如**z = (a>b)?a:b;**
——不要使用空的**if else** 语句。如
if(mychar >= ‘A’)
if(mychar <= ‘Z’)
printf(“This is a letter \n”);
else
printf(“This is not a letter \n”);
else到底是否定哪个**if**容易引起误解。可通过加**{ }**避免误解。
——尽量减少使用“否定”条件的条件语句。如：
把 **if(!((mychar<’0’) || (mychar>’9’)))**
改为**if((mychar>=’0’) && (mychar<=’9’))**

一些需要注意的问题-1

- 随时注意表达式计算过程和类型。注意运算符的优先级和结合顺序，不同类型的运算对象将怎样转换，运算的结果是什么类型的，等等。在必要的时候加上括号或显式的类型强制转换。
- C语言的运算符很多，优先级定义也不尽合理，很难完全记清楚，因此要特别注意。需要时查一查（不要怕麻烦，相关网页有运算符表），或者直接按照自己的需要加上几个括号。

一些需要注意的问题-2

- 绝不去写依赖于运算对象求值顺序的表达式。对于普通二元运算符的运算对象，函数调用的各个实际参数，C语言都没有规定特定求值顺序。因此，我们不应该写那种依赖于特定求值顺序的表达式，因为不能保证它一定得到什么结果。

```
#include <stdio.h>

main()
{
    int i;
     i=3;
     printf("%d ,%d\n",i,++i);
     printf("%d ,%d\n",i,i++);
     return 0;
}

j=i++;
printf(“%d ,%d\n",i,j);
```

一些需要注意的问题-3

- 总注意检查数组的界限和字符串（也以数组的方式存放）的结束。C语言内部根本不检查数组下标表达式的取值是否在合法范围内，也不检查指向数组元素的指针是不是移出了数组的合法区域。写程序的人需要自己保证对数组使用的合法性。越界访问可能造成灾难性的后果。
- 例：在写处理数组的函数时一般应该有一个范围参数；处理字符串时总检查是否遇到空字符‘\0’。
- 绝不对空指针或者悬空的指针做间接访问。这种访问的后果不可预料，可能造成系统的破坏，也可能造成操作系统发现这个程序执行非法操作而强制将它终止。

一些需要注意的问题-4

- 对于所有通过返回值报告运行情况或者出错信息的库函数，都应该检查其执行是否正常完成。如果库函数没有完成操作（可能因为各种原因），随后的操作有可能就是非法的。这种错误也可能在程序运行中隐藏很长时间，到很后来才暴露出来，检查错误非常困难

- 参考书:
  - 《程序设计实践》，（**The Practice Of Programming, Brian W. Kernighan & Bob Pike 1999)**。机械工业出版社**2000**。