程序的规范化

在 C 语言中我们不遵守编译器就会报错的规定叫规则；约定成俗但不遵守也不会出错的规定叫规范。

比如变量必须先定义后使用，如果我们不定义就直接使用就会出错。

int a =4;

a+=5;

这是正确的。

a+=5;

这是错误的。因为没有事先定义变量 a。这叫规则，必须得遵守。

再比如左右花括号{}必须按列对齐写，这是规范。你不对齐写，程序会显得很乱，但编译器仍可以照常运行它。

不幸的是，我初学 C 的一年中所写的程序全是不规范的。当我后来回头再看当年写的程序时，由于它太乱，也没有任何注释，这一年所敲的程序几乎没有任何可读性（程序员把它称之为垃圾程序）。 如果说，当初因为没有人告诉我程序要写规范，为此而付出惨重代价是可以原谅的话，那您看到我的教训仍然坚持创造出这世界上恐怕除了你没人能看懂的“风格”，对此我只能表示遗憾。

确切的说程序的规范化有两大好处，第一可读性高，无论自己还是他人都可以很容易读懂， 第二可以减少很多不必要的错误。比如漏掉括号），加错花括号}等错误都可以避免。

程序的规范化有两大原则： 第一是成对书写，如敲完(马上敲)然后再在（与）的中间敲语句。 这样永远不会漏掉右边的）。 第二使用缩进。

程序的风格，即程序的规范化涉及到的细节很多，下面我仅以贝尔实验室的风格为例，简明说几点：

1、成对的{}必须按列对齐写，即先写{，再在下几行按列对齐写}，最后返回去在{与}的中间写语句。左右括号类似。

2、声明与语句要分开写。

{与}要独占一行

函数与函数之间要空一行分开

```
/*
    自编的 strcmy()函数
*/
# include <stdio.h>
int main(void)
{                       /* 无论是 { 还是 }， 我们都要独占一行 */
    int mystrcmy(char *，  char *);
    char  str1[20]，;
     char  str2[20];   /* 上面是声明， 与下面的语句要空一行，分开 */

     clrscr();
    gets(str1);
    gets(str2);
    printf("%d\n"，mystrcmp(str1， str2));

     return 0;
}
```

```
/* 主函数与下面的 mystrcmp()函数要空行分开 */
int mystrcmp(char *p，  char *q)
{
    while  ( *p == *q  &&  *p != '\0')
    {
        p++;
        q++;
    }

    return (*p - *q);
}
```

3、地位相等的语句按列对齐写，功能嵌套的按 3 个空格缩进。

```
Status OrderInsertMerge(LinkList *L，ElemType e，int(* compare)(term，term))
{  /* 按有序判定函数 compare()的约定，将值为 e 的结点插入或合并到升序链表 L 的
      适当位置 */
   Position q，s;
   if(LocateElemP(*L，e，&q，compare)) /* L 中存在该指数项 */
   {
       q->data。coef+=e。coef; /* 改变当前结点系数的值 */
       if(!q->data。coef) /* 系数为 0 */
       {  s=PriorPos(*L，q); /* s 为当前结点的前驱 */
          if(!s) /* q 无前驱 */
          s=(*L)。head; DelFirst(L，s，&q);
          FreeNode(&q);
       }

       return OK;
   }
   else  if(MakeNode(&s，e)) /* 生成结点成功 */
   {     InsFirst(L，q，s);
         return OK;
   }
   else /* 生成结点失败 */
   {
        return ERROR;
   }
}
```

如果不按规范写那就成了：

```
int fun(LinkList *L，ElemType e，int(* compare)(term，term))
 { Position q，s;
  if(LocateElemP(*L，e，&q，compare))
  { q->data。coef+=e。coef; if(!q->data。coef){      /*  第 3 行*/
  s=PriorPos(*L，q); if(!s) s=(*L)。head;
```

```
  DelFirst(L，s，&q); FreeNode(&q);
  } return OK;
  }else                    /*  第 7 行  */
  if(MakeNode(&s，e))
{ InsFirst(L，q，s);
return OK;
  }else   return ERROR;
}
```

当你看到这个函数时，能一眼看出它的功能么？ 你能看出第 3 行的 if 的范围是么，答案是不可能的。

另外比如定义的特殊用途的变量要在定义时加注释，函数的前面要加功能的注释，在整个程序的前面要写上日期，最重要的是写上编本程序的目的等等，类似的就不在此赘述了

郝斌

2007-4-6