第七章 命名规则

比较著名的命名规则当推 Microsoft 公司的“匈牙利”法，该命名规则的主要思想是“在变量和函数名中加入前缀以增进人们对程序的理解”。例如所有的字符变量均以 ch 为前缀，若是指针变量则追加前缀 p。如果一个变量由 ppch 开头，则表明它是指向字符指针的指针。

“匈牙利”法最大的缺点是烦琐，例如

```
int    i,  j,  k;
float  x,  y,  z;
```

倘若采用“匈牙利”命名规则，则应当写成

```
int    iI,  iJ,  ik;  // 前缀 i 表示 int 类型
float  fX,  fY,  fZ;  // 前缀 f 表示 float 类型
```

如此烦琐的程序会让绝大多数程序员无法忍受。

据考察，没有一种命名规则可以让所有的程序员赞同，程序设计教科书一般都不指定命名规则。命名规则对软件产品而言并不是“成败悠关”的事，我们不要化太多精力试图发明世界上最好的命名规则，而应当制定一种令大多数项目成员满意的命名规则，并在项目中贯彻实施。

7.1 共性规则

本节论述的共性规则是被大多数程序员采纳的，我们应当在遵循这些共性规则的前提下，再扩充特定的规则（见 7.2 节）。

- **【规则 7-1-1】**标识符应当直观且可以拼读，可望文知意，不必进行“解码”。

标识符最好采用英文单词或其组合，便于记忆和阅读。切忌使用汉语拼音来命名。程序中的英文单词一般不会太复杂，用词应当准确。例如不要把 CurrentValue 写成 NowValue。

- **【规则 7-1-2】**标识符的长度应当符合“min-length && max-information”原则。

几十年前老 ANSI C 规定名字不准超过 6 个字符，现今的 C++/C 不再有此限制。一般来说，长名字能更好地表达含义，所以函数名、变量名、类名长达十几个字符不足为怪。那么名字是否越长约好？不见得！例如变量名 maxval 就比 maxValueUntilOverflow 好用。单字符的名字也是有用的，常见的如 i,j,k,m,n,x,y,z 等，它们通常可用作函数内的局部变量。

- **【规则 7-1-3】**命名规则尽量与所采用的操作系统或开发工具的风格保持一致。

例如 Windows 应用程序的标识符通常采用“大小写”混排的方式，如 AddChild。而

Unix 应用程序的标识符通常采用“小写加下划线”的方式，如 add_child。别把这两类风格混在一起用。

- **【规则 7-1-4】**程序中不要出现仅靠大小写区分的相似的标识符。
  例如：
  ```
  int   x,   X;          // 变量 x 与 X 容易混淆
  void foo(int x);   // 函数 foo 与 F00 容易混淆
  void F00(float x);
  ```

- **【规则 7-1-5】**程序中不要出现标识符完全相同的局部变量和全局变量，尽管两者的作用域不同而不会发生语法错误，但会使人误解。

- **【规则 7-1-6】**变量的名字应当使用“名词”或者“形容词＋名词”。
  例如：
  ```
  float    value;
  float    oldValue;
  float    newValue;
  ```

- **【规则 7-1-7】**全局函数的名字应当使用“动词”或者“动词＋名词”（动宾词组）。类的成员函数应当只使用“动词”，被省略掉的名词就是对象本身。
  例如：
  ```
  DrawBox();          // 全局函数
  box->Draw();        // 类的成员函数
  ```

- **【规则 7-1-8】**用正确的反义词组命名具有互斥意义的变量或相反动作的函数等。
  例如：
  ```
  int  minValue;
  int  maxValue;

  int  SetValue(…);
  int  GetValue(…);
  ```

✧ **【建议 7-1-1】**尽量避免名字中出现数字编号，如 Value1,Value2 等，除非逻辑上的确需要编号。这是为了防止程序员偷懒，不肯为命名动脑筋而导致产生无意义的名字（因为用数字编号最省事）。我长这么大，从来都没有见到过有人把子女的名字叫做张三或李四。

7.2 简单的 Windows 应用程序命名规则

作者对“匈牙利”命名规则做了合理的简化，下述的命名规则简单易用，比较适合于 Windows 应用软件的开发。

- **【规则 7-2-1】**类名和函数名用大写字母开头的单词组合而成。

  例如：

```
class Node;                  // 类名
class LeafNode;              // 类名
void  Draw(void);            // 函数名
void  SetValue(int value);// 函数名
```

- **【规则 7-2-2】**变量和参数用小写字母开头的单词组合而成。

  例如：

```
BOOL flag;
int  drawMode;
```

- **【规则 7-2-3】**常量全用大写的字母，用下划线分割单词。

  例如：

```
const int MAX = 100;
const int MAX_LENGTH = 100;
```

- **【规则 7-2-4】**静态变量加前缀 s_（表示 static）。

  例如：

```
void Init(…)
{
    static int s_initValue;  // 静态变量
    …
}
```

- **【规则 7-2-5】**如果不得已需要全局变量，则使全局变量加前缀 g_（表示 global）。

  例如：

```
int g_howManyPeople;// 全局变量
int g_howMuchMoney;     // 全局变量
```

- **【规则 7-2-6】**类的数据成员加前缀 m_（表示 member），这样可以避免数据成员与成员函数的参数同名。

  例如：

```
void Object::SetValue(int width, int height)
```

```
{
    m_width = width;
    m_height = height;
}
```

- **【规则 7-2-7】**为了防止某一软件库中的一些标识符和其它软件库中的冲突，可以统一为各种标识符加上能反映软件性质的前缀。例如三维图形标准 OpenGL 的所有库函数均以 gl 开头，所有常量（或宏定义）均以 GL 开头。