



编者按:本文是《学习 MISRA - C》系列连载讲座之五,共六讲。

第一讲:“安全第一”的 C 语言编程规范”,简述 MISRA - C 的概况。

第二讲:“跨越数据类型的重重陷阱”,介绍规范的数据定义和操作方式,重点在隐式数据类型转换中的问题。

第三讲:“指针、结构体、联合体的安全规范”,解析如何安全而高效地应用指针、结构体和联合体。

第四讲:“防范表达式的失控”,剖析 MISRA - C 中关于表达式、函数声明和定义等的不良使用习惯,最大限度地减小各类潜在错误。

第五讲:“准确的程序流控制”,表述 C 语言中控制表达式和程序流控制的规范做法。

第六讲:“构建安全的编译环境”,讲解与编译器相关的规范编写方式,避免来自编译器的隐患。

准确的程序流控制

清华大学 张首钊 邵贝贝

程序的执行流程是由条件判断、跳转和循环构成的,没有任何一个程序会缺少程序流的控制。那么像 if、for、while、switch 等这些程序员无比熟悉的语句也存在隐患吗?事实上,C 语言是很灵活的,这种灵活性给程序员编写代码带来了便利,但同时也带来了许多容易导致混淆的表达。这些表达完全符合 C 语言标准,但有时程序员也难以发现自己犯了错误,最终的结果是使程序进入错误的执行流程。即使程序员没有犯错误,但有些容易混淆的表达也会给其他人读懂程序带来困扰,使程序的维护变得困难。除此以外,有少量控制流程的方式还会产生不确定的运行结果,而这些结果也不容易被发觉。

如何使程序的流程控制清晰、准确,不产生混淆的表达呢? MISRA - C 给出了很多的相关规定,使程序流的控制变得规范,避免产生各种混淆和不确定性,从而最大程度上减少程序流控制中的失误,并使程序的维护更加容易。

下面从几个例子出发,讲述这些混淆是如何产生的,最后给出 MISRA - C 关于程序流控制的相关规则,帮助读者规范编程的习惯。

1 容易混淆的表达方式

先来看这样两段代码:

```
uint8_t x,y ;
...
```

```
if ( x == y) {
    foo() ;
}

uint8_t x,y ;
...
if ( x = y) {
    foo() ;
}
```

在 C 标准中,条件语句需要的是布尔值,条件语句表达式的布尔值实际上是按照整型处理的,所以这两段代码在语法和逻辑上都没有任何问题。第一段代码判断 x 是否等于 y,如果相等,调用 foo() 函数;第二段代码首先将 y 的值赋给 x,然后判断 x 是否为 0,如果不为 0,调用 foo() 函数。这两段代码只相差一个等号,却使判断条件大不相同,程序的执行流程会出现很大差别。

相信读者在写程序的时候都碰到过将“==”这个判断语句误写成赋值语句“=”的情况。那么面对这两个语句时,如何能快速准确地判断这是正确的还是程序员的失误呢?当程序比较简单的时候,很容易判断,但当程序流程比较复杂的时候,可能花费大量时间还难以给出确定的答案,而这些地方极有可能是有错误的。

这样的混淆,事实上是可以轻松避免的,MISRA - C 提出了如下强制性的规则。



规则 13.1:赋值表达式不能用在需要布尔值的地方。

按照 MISRA - C 的标准,第二段代码应该写成:

```
uint8_t x,y;
...
x = y ;
if (x != 0) {
    foo() ;
}
```

这样,当看到需要布尔值的地方出现了赋值表达式,就可以立即判断这是一个错误。在这条规则下,如下的表达也是不允许的:

```
if ( (x = y) != 0 ) {
    foo() ;
}
```

与这条规则类似,MISRA - C 还提出了如下推荐的规则,来避免整型变量和布尔型的混淆。

规则 13.2(推荐):判断一个值是否为 0 应该是显式的,除非该操作数是一个布尔值。

这条规则禁止了如下的表达:

```
uint8_t x ;
...
if ( x )
{ ... }
```

再来看一个例子:

```
uint8_t x ;
uint8_t a,b ;
...
switch( x ) {
    case 1 :
        a = b ;
    case 2 :
        a += 2 ;
        break ;
    case 3 :
        ...
}
```

同样,这段代码在语法和逻辑上也没有任何问题,编译器也不会给出任何错误或者警告。在程序执行中,当 x 等于 1 的时候,将 b 的值赋给 a,然后将 a 加 2,退出;当 x 等于 2 的时候,直接将 a 的值加 2,接着退出。但这儿很可能是一段错误的代码,程序员的本意有可能是 x 等于 1 时,将 b 的值赋给 a,当 x 等于 2 时,直接将 a 的值加 2。

为了避免这样的混淆,MISRA - C 提出了如下强制性的规则。

规则 15.2:所有非空的 switch 子句都应该以 break 语句结束。

按照这条规则,上面的程序应该写成:

```
switch( x ) {
    case 1 :
        a = b ;
        break ;
    case 2 :
        a += 2 ;
        break ;
    case 3 :
        ...
}
```

或者:

```
switch( x ) {
    case 1 :
        a = b ;
        a += 2 ;
        break ;
    case 2 :
        a += 2 ;
        break ;
    case 3 :
        ...
}
```

MISRA - C 中还有一些防止程序流控制中出现混淆的规则。

规则 13.5:for 语句中的 3 个表达式只能和循环控制相关。第一个表达式只能为循环变量赋初值,第二个表达式只能进行循环条件的判断,第三个表达式只能进行循环变量增(减)值。

规则 13.6:for 循环中,循环变量只能在 for 语句的第三个表达式中修改,不允许在循环体中修改。

规则 13.7:布尔表达式的值必须是可以改变的。

例如,如下代码是不允许的:

```
uint8_t x;
...
if ( x >= 0 )
```

...
错误在于该条件判断的结果始终为真。

规则 14.1:不能存在无法执行到的代码。

规则 14.2:非空语句必须要么产生副作用(side-effect);或者使程序流程改变。

例如,下面的代码是不允许的:

```
...
x >= 3 ;
...
```

副作用是指表达式执行后对程序运行环境造成的影响。赋值语句、自增操作等都是典型的具有副作用的操作。



错误在于 x 和 3 比较的结果被丢弃了。

规则 14.3:一行中如果有空语句,那么该行只能有这条空语句,不能有别的语句,并且在这条空语句前不能有注释,注释必须在其后,用空格隔开。

例如,如下的代码都是不允许的:

```
while (port & 0x80 == 0) {
    ; foo ();
    /* wait for pin */ ;
    /* wait for pin */
}
```

其中 的错误是除了空语句还有一条语句; 的错误是在空语句前有注释; 的错误是空语句与注释没用空格隔开。

规则 14.8:switch、while、do...while 和 for 语句的主体必须是复合语句(即用大括号包含),即使该主体只包含一条语句。

例如,如下代码是符合 MISRA - C 标准的:

```
for (i = 0 ; i < N_ELEMENTS; ++ i)
{
    buffer[i] = 0 ;
}
```

规则 14.9:if 结构后面必须是一个复合语句(即用大括号包含),else 后面必须是一个复合语句(即用大括号包含)或者另一个 if 语句。

规则 15.1:switch 语句的主体必须是复合语句(即用大括号包含)。

规则 15.2:所有非空的 switch 子句都应该用 break 语句结束。

规则 15.3:switch 的最后一个子句必须是 default 子句,如果 default 中没有包含任何语句,那么应该有注释来说明为什么没有进行任何操作。

规则 15.4:switch 表达式不能是一个有效的布尔值。

例如,下面的代码是不允许的:

```
switch ( x == 0 )
{ ... }
```

规则 15.5:switch 语句必须至少包含一个 case 子句。

2 导致混乱的表达方式

在 C 语言中,有一些表达方式可以使程序的代码量减少,但却会使程序的结构化程度降低,流程控制变得混乱,可读性大大降低。看下面一段代码:

```
if ( a > 0x02 )
{
loop1:  b += 1 ;
    if ( c > 0xA0 ) {
        goto loop3 ;
    }
}
```

```
}
loop2:  a = 2 * b ;
    c = a + b ;
    if ( c < 0x40 ) {
        goto loop1 ;
    } else {
        goto loop2 ;
    }
}
```

loop3: ...

这段代码读起来很困难。实际编程时,程序员实现这段功能的代码自然不会这样写,但是当程序流程复杂的时候,各种看起来能使编程工作变得轻松的表达,例如 goto、continue 等语句,却会使程序流程变得混乱,可读性降低,而隐藏其中的问题,很可能就无法发现了。

针对这种情况,MISRA - C 给出了下面几条强制规则。

规则 14.4:不允许使用 goto 语句。

规则 14.5:不允许使用 continue 语句。

规则 14.6:循环体中最多只能出现一个 break 语句用于结束循环。

规则 14.7:函数只能有一个出口,这个出口必须在函数末尾。

规则 14.10:if...else if 结构必须由一个 else 子句结束。

当 if 语句后面有一个或者多个 else if 语句时,最后的一个 else if 必须有一个与之对应的 else 语句。如果只有一个 if 语句时,else 不是必须的。

3 不确定的执行结果

除了导致混淆和混乱的表达外,还有一些对浮点数的操作会导致不确定的结果。来看如下一段代码:

```
float32_t x,y ;
... /* 一些运算 */
if ( x == y )
{ ... }
```

if 的条件无法肯定什么情况为真。这是因为浮点数在计算机中无法用二进制精确表示,其运算总会存在舍入和切断误差,很多人看起来相等的结果,但计算机给出的两个浮点数并不相等,所以上面代码中 if 的主体语句什么情况执行是不确定的。MISRA - C 给出了两条相关的规定来解决这一问题。

规则 13.3:不允许对浮点数进行相等或者不相等的比较,即使是非直接的比较也是不允许的。

例如,如下非直接的比较也是不允许的:



```
float32_t x,y ;
...
if ( x <= y )
{ ...}
```

规则 13.4:for 循环的控制表达式不应包含浮点数据类型。

3 小结

好的代码,要安全可靠、有很好的可读性和可维护性。在 C 语言中,一些表达方式,可能会稍微减少程序员编程的工作量,但却会使程序的流程变得难以判断,其中的错误可能就无法发现。

按照 MISRA - C 的标准来写代码,就可以避免程序流程产生混淆和混乱,排除其中的不确定因素,使程序真

正按照程序员设想的工作,并使代码更清晰易懂,真正实现安全可靠,并具有良好的可读性和可维护性。ME

参考文献

- 1 MISRA - C:2004, Guidelines for the use of the C language in critical systems. The Motor Industry Software Reliability Association, 2004
- 2 Harbison III. Samuel P, Steele Jr. Guy L. C 语言参考手册, 邱仲潘, 等译, 第 5 版. 北京:机械工业出版社, 2003
- 3 Les Hatton. The MISRA C Compliance Suite - The next step, Oakwood Computing. <http://www.misra-c2.com/>
- 4 ISO/IEC 9899:1999. International Organization of Standardization, 1999

(收稿日期:2005-12-31)

全国第六届嵌入式系统学术交流暨产品展示会 征文通知

中国计算机学会学术年会之一:全国第六届嵌入式系统学术交流暨产品展示会定于 2006 年 10 月 28 至 30 日在旅游圣地、科技教育文化名城西安召开。

主办单位:中国计算机学会微机(嵌入式系统)专业委员会

承办单位:北京大学信息科学技术学院、陕西省计算机学会、《电子产品世界》杂志社

协办单位:西北工业大学、航天 771 研究所、航空第 631 研究所

会议主题:嵌入式系统与技术创新

会议将邀请两院院士、863 专项专家和国内外著名专家及知名厂商的精英就嵌入式系统与 SoC 开发创新及应用做主题报告,并同时进行论文交流和嵌入式系统及其应用多国产品展示会。

一、征文内容:

1. 嵌入式系统的新结构、新技术、新发展、新产品、新器件、新成果;
2. 嵌入式系统仿真平台、开发技术、开发工具、综合平台;
3. 嵌入式系统及系统芯片(SOC)设计、IP Core、接口综合技术及其资源库;
4. 嵌入式操作系统、实时操作系统、嵌入式数据库及其软件;
5. 专用集成电路(ASIC)、电子设计自动化(EDA)、FPGA、DSP、多 MCU 及相关技术;
6. 嵌入式系统与单片机的应用:
 - ①面向 Internet 的嵌入式系统:通信技术、网络安全、宽带接入、E-Home 等。
 - ②面向数码消费电子的嵌入式系统:信息家电、数码产品、娱乐游戏等。
 - ③面向控制的嵌入式系统:工业控制、安防监控、智能仪表、医疗设备等。
 - ④面向汽车电子的嵌入式系统:车载 AV 系统、控制系统、GPS、电子地图等。
 - ⑤面向无线网络的嵌入式系统:无线传感器网络、无线网络、RFID 技术、蓝牙技术、遥测遥控等。

二、论文集出版:中国计算机学会指定其会刊《计算机技术与发展》(国家一级刊物)以专刊形式出版,同时选拔优秀论文推荐至权威期刊发表。

三、投稿方法:论文最好以电子邮件形式投稿,若为打印稿请将论文软盘一同寄出。用英文投稿者请同时附中文全稿。投稿论文经专家评审后,对录用论文将及时发给录用通知和有关参会要求。对中国计算机学会会员、微机(嵌入式系统)专委会会员、博士(硕士、大学)生及有基金项目者投稿,将优惠减收版面费。未录用论文不退稿。

四、截稿日期:2006 年 8 月 31 日(以邮戳为准)

五、联系方式:张 维	北京大学信息科学技术学院	邮编: 100871
	电话: 010 - 62763331	Email: wwzhang@pku.edu.cn
王铁鹰	《计算机技术与发展》编辑部	邮编: 710054
	电话: 029 - 85522163	Email: wjz@sninfo.gov.cn
	029 - 68810921	wjz@163.com

欢迎投稿 欢迎参加会议