C 语言常见错误

C 语言的最大特点是：功能强、使用方便灵活。C 编译的程序对语法检查并不象其它高级语言那么严格，这就给编程人员留下“灵活的余地”，但还是由于这个灵活给程序的调试带来了许多不便，尤其对初学 C 语言的人来说，经常会出一些连自己都不知道错在哪里的错误。

1.书写标识符时，忽略了大小写字母的区别。

```
main()
{   int a=5;
   printf("%d",A);
}
```

编译程序把 a 和 A 认为是两个不同的变量名，而显示出错信息。C 认为大写字母和小写字母是两个不同的字符。习惯上，符号常量名用大写，变量名用小写表示，以增加可读性。

2.忽略了变量的类型，进行了不合法的运算。

```
main()
{   float a,b;
   printf("%d",a%b);
}
```

%是求余运算，得到 a/b 的整余数。整型变量 a 和 b 可以进行求余运算，而实型变量则不允许进行“求余”运算。

3.将字符常量与字符串常量混淆。

```
char c;
c="a";
```

在这里就混淆了字符常量与字符串常量，字符常量是由一对单引号括起来的单个字符，字符串常量是一对双引号括起来的字符序列。C 规定以“\”作字符串结束标志，它是由系统自动加上的，所以字符串“a”实际上包含两个字符：‘a'和‘\'，而把它赋给一个字符变量是不行的。

4.忽略了“=”与“= =”的区别。

在许多高级语言中，用“=”符号作为关系运算符“等于”。如在 BASIC 程序中
可以写

```
if (a=3) then …
```

但 C 语言中，“=”是赋值运算符，“= =”是关系运算符。如：

```
if (a==3) a=b;
```

前者是进行比较，a 是否和 3 相等，后者表示如果 a 和 3 相等，把 b 值赋给 a。由于习惯问题，初学者往往会犯这样的错误。

5.忘记加分号。

分号是 C 语句中不可缺少的一部分，语句末尾必须有分号。

```
a=1
b=2
```

编译时，编译程序在“a=1”后面没发现分号，就把下一行“b=2”也作为上一行语句的一部分，这就会出现语法错误。改错时，有时在被指出有错的一行中\未发现错误，就需要看一下上一行是否漏掉了分号。

```
{ z=x+y;
t=z/100;
printf("%f",t);
}
```

对于复合语句来说，最后一个语句中最后的分号不能忽略不写(这是和 PASCAL
不同的)。

6.多加分号。

对于一个复合语句，如：

```
{ z=x+y;
t=z/100;
```

printf("%f",t);
};
复合语句的花括号后不应再加分号，否则将会画蛇添足。
又如：
if (a%3==0);
I++;
本是如果 3 整除 a，则 I 加 1。但由于 if (a%3==0)后多加了分号，则 if 语句到此结束，程序将执行 I++语句，不论 3 是否整除 a，I 都将自动加 1。
再如：
for (I=0;I<5;I++);
{scanf("%d",&x);
printf("%d",x);}
本意是先后输入 5 个数，每输入一个数后再将它输出。由于 for()后多加了一个分号，使循环体变为空语句，此时只能输入一个数并输出它。

7.输入变量时忘记加地址运算符“&”。
int a,b;
scanf("%d%d",a,b);
这是不合法的。Scanf 函数的作用是：按照 a、b 在内存的地址将 a、b 的值存进去。“&a”指 a 在内存中的地址。

8.输入数据的方式与要求不符。
① scanf("%d%d",&a,&b);
输入时，不能用逗号作两个数据间的分隔符，如下面输入不合法：
3，4
输入数据时，在两个数据之间以一个或多个空格间隔，也可用回车键，跳格键 tab。
② scanf("%d,%d",&a,&b);
C 规定：如果在“格式控制”字符串中除了格式说明以外还有其它字符，则在输入数据时应输入与这些字符相同的字符。下面输入是合法的：
3，4
此时不用逗号而用空格或其它字符是不对的。
3 4　3：4
又如：
scanf("a=%d,b=%d",&a,&b);
输入应如以下形式：
a=3,b=4

9.输入字符的格式与要求不一致。
在用“%c”格式输入字符时，“空格字符”和“转义字符”都作为有效字符输入。
scanf("%c%c%c",&c1,&c2,&c3);
如输入 a b c
字符“a”送给 c1，字符“ ”送给 c2，字符“b”送给 c3，因为%c 只要求读入一个字符，后面不需要用空格作为两个字符的间隔。

10.输入输出的数据类型与所用格式说明符不一致。
例如，a 已定义为整型，b 定义为实型
a=3;b=4.5;
printf("%f%d\n",a,b);
编译时不给出出错信息，但运行结果将与原意不符。这种错误尤其需要注意。

11.输入数据时，企图规定精度。
scanf("%7.2f",&a);
这样做是不合法的，输入数据时不能规定精度。

12.switch 语句中漏写 break 语句。
例如：根据考试成绩的等级打印出百分制数段。

```
switch(grade)
{ case 'A':printf("85~100\n");
case 'B':printf("70~84\n");
case 'C':printf("60~69\n");
case 'D':printf("<60\n");
default:printf("error\n");
```

由于漏写了 break 语句，case 只起标号的作用，而不起判断作用。因此，当 grade 值为 A 时，printf 函数在执行完第一个语句后接着执行第二、三、四、五个 printf 函数语句。正确写法应在每个分支后再加上“break;”。例如

```
case 'A':printf("85~100\n");break;
```

13.忽视了 while 和 do-while 语句在细节上的区别。

```
(1)main()
{int a=0,I;
scanf("%d",&I);
while(I<=10)
{a=a+I;
I++;
}
printf("%d",a);
}
(2)main()
{int a=0,I;
scanf("%d",&I);
do
{a=a+I;
I++;
}while(I<=10);
printf("%d",a);
}
```

可以看到，当输入 I 的值小于或等于 10 时，二者得到的结果相同。而当 I>10 时，二者结果就不同了。因为 while 循环是先判断后执行，而 do-while 循环是先执行后判断。对于大于 10 的数 while 循环一次也不执行循环体，而 do-while 语句则要执行一次循环体。

14.定义数组时误用变量。

```
int n;
scanf("%d",&n);
int a[n];
```

数组名后用方括号括起来的是常量表达式，可以包括常量和符号常量。即 C 不允许对数组的大小作动态定义。

15.在定义数组时，将定义的“元素个数”误认为是可使的最大下标值。

```
main()
{static int a[10]={1,2,3,4,5,6,7,8,9,10};
printf("%d",a[10]);
}
```

C 语言规定：定义时用 a[10]，表示 a 数组有 10 个元素。其下标值由 0 开始，所以数组元素 a[10]是不存在的。

16.初始化数组时，未使用静态存储。

```
int a[3]={0,1,2};
```

这样初始化数组是不对的。C 语言规定只有静态存储(static)数组和外部存储(exterm)数组才能初始化。应改为：

```
static int a[3]={0,1,2};
```

17.在不应加地址运算符&的位置加了地址运算符。

scanf("%s",&str);
C 语言编译系统对数组名的处理是：数组名代表该数组的起始地址，且 scanf 函数中的输入项是字符数组名，不必要再加地址符&。应改为：scanf("%s",str);

18.同时定义了形参和函数中的局部变量。

```
int max(x,y)
int x,y,z;
{z=x>y?x:y;
return(z);
}
```

形参应该在函数体外定义，而局部变量应该在函数体内定义。应改为：

```
int max(x,y)
int x,y;
{int z;
z=x>y?x:y;
return(z);
}
```