


十六进制		无符号		补码	
原始值	截断值	原始值	截断值	原始值	截断值
0	0	0		0	
2	2	2		2	
9	1	9		-7	
B	3	11		-5	
F	7	15		-1	

解释如何将等式(2.9)和等式(2.10)应用到这些示例上。

2.2.8 关于有符号数与无符号数的建议


就像我们看到的那样，有符号数到无符号数的隐式强制类型转换导致了某些非直观的行为。而这些非直观的特性经常导致程序错误，并且这种包含隐式强制类型转换的细微差别的错误很难被发现。因为这种强制类型转换是在代码中没有明确指示的情况下发生的，程序员经常忽视了它的影响。

下面两个练习题说明了某些由于隐式强制类型转换和无符号数据类型造成的细微的错误。

 **练习题 2.25** 考虑下列代码，这段代码试图计算数组 a 中所有元素的和，其中元素的数量由参数 length 给出。

```
1  /* WARNING: This is buggy code */
2  float sum_elements(float a[], unsigned length) {
3      int i;
4      float result = 0;
5
6      for (i = 0; i <= length-1; i++)
7          result += a[i];
8      return result;
9  }
```

当参数 length 等于 0 时，运行这段代码应该返回 0.0。但实际上，运行时会遇到一个内存错误。请解释为什么会发生这样的情况，并且说明如何修改代码。

 **练习题 2.26** 现在给你一个任务，写一个函数用来判定一个字符串是否比另一个更长。前提是你要用字符串库函数 strlen，它的声明如下：

```
/* Prototype for library function strlen */
size_t strlen(const char *s);
```

最开始你写的函数是这样的：

```
/* Determine whether string s is longer than string t */
/* WARNING: This function is buggy */
int strlonger(char *s, char *t) {
    return strlen(s) - strlen(t) > 0;
}
```

当你在一些示例数据上测试这个函数时，一切似乎都是正确的。进一步研究发现在头文件 stdio.h 中数据类型 size_t 是定义成 unsigned int 的。

- 在什么情况下，这个函数会产生不正确的结果？
- 解释为什么会出现这样不正确的结果。
- 说明如何修改这段代码好让它能可靠地工作。