类型转换

类型转换，指的是一种类型的数据转换为另一种类型的数据（此另一种数据也叫**目标转换类型**），由于各不同类型之间有字节大小、精度等的差异，所以在转换过程中会涉及到字节提升、截短等问题。在转换过程中，需尽力保证数据损失最小。

主要有下列三种情况的数据转换：

1、当不同类型数据进行运算的时候，编译器会隐式的将数据类型转换为统一。

2、在赋值、函数调用时，当原数据和目标数据类型不同时，编译器需将原数据转换为目标数据。

3、在遇到强制类型转换运算符时，会强制进行类型转换。

下面分别介绍：

一、当不同类型数据进行运算的时候，编译器会隐式的将数据类型转换为统一。

当多种类型数据进行运算的时候，编译器会按照运算符优先级最终分为两个数据之间的运算（三目运算符和一目运算符不涉及类型转换问题）。对于两个数据，编译器遵循尽量不损坏数据的原则，使用右侧图示顺序进行转换。

图中横向表示必须的转换，例如float，只要有float参与运算，就必须转换为double再运算，此举是为了保证精度。char和short，也必须转换为int参与运算，也是为了保证精度。

对于整型之间的类型提升，当原数据有符号时，在左侧必须补符号位相同的值，当无符号时，补0。

在此要注意char型，因为char到底应该是unsigned还是signed，不同的编译器默认是不一样的。在windows的cl编译器和linux的gcc编译器，char默认signed char，是有符号数，而对于例如linux-arm-gcc等编译器，char默认unsigned char，为无符号数）。

整型往double转换，小数位补零即可。

二、在赋值、函数调用时，当原数据和目标数据类型不同时，编译器需将原数据转换为目标数据。

当赋值时，左值为目标转换类型。

当函数调用实参往形参传值时，形参为目标转换类型。

当函数返回值和return后的类型不匹配时，函数返回值为目标转换类型。

例如：int fun() {double d; return d}; //d为double型，而要求ruturn int型。

在这些情况下，不仅要考虑类型提升的问题（从小类型到大类型），还要考虑类型降低的问题（从大类型到小类型）。

实型到整型，直接截掉小数位和多余的整数高位。

长字节整型到短字节整型，丢弃多余的高位。

double到float，丢掉多余的精度和取值范围。

三、强制类型转换

强制类型转换方法同情况二。

下面是历次的试题中类型转换相关题目，请参考。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

第2题（单选/3分）：下面代码输出结果是:  
char a = 256; //这是个赋值语句，右值为int型，左值为char型，所以目标转换类型为char，需要从长字节往短字节转换，直接截短，所以，去256的低8位，a=00000000.

int d = a; //赋值语句，右值char，左值int，这里是从短到长转换，需要考虑char是unsigned还是signed，因为a为0，所以最高位都是补0。所以d=0.  
printf("%d", d+1); //此处都是int，所以d=1.  
-1  
1  
257  
0



可以考虑，若char a=-1，则这个问题就要考虑char是unsigned还是signed了。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

第3题 单选题(4分)

下面程序的输出结果是什么？

int main(int argc, char \*\*agrv)  
{  
char dataptr[3] = {0x01, 0x80, 0xc2};  
if (dataptr[0] == 0x01 && dataptr[1] == 0x80 && dataptr[2] == 0xc2)  
printf("Mac match\n");  
else   
printf("Mac Not match\n");  
return 0;  
}

1) Mac match  
2) Mac Not match  
3) 二者皆错

解析：这个题目，cl和gcc下，输出"Mac Not match\n"，linux-arm-gcc下，输出"Mac match\n"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/