**TYPE CASTING**

When an operator has operands of different type, they are converted to a common type according to a small number of rules. In general, the only automatic conversions are those that converts a “**narrower**” operand into a “**wider**” one without losing information, such as converting an integer to floating point in an expression like f + i.

Thus “**lower**” type is promoted to the “**higher**” type before the operation proceeds. The result is of the higher type.

Explicit type conversion: (type-name) expression

The **expression** is converted to the named type. Unary operator called cast is used for this purpose.

If arguments are declared by a function prototype, as they normally should be, the declaration causes the automatic coercion (forced) of any argument when the function is called.

Thus, given a function prototype for sqrt:

Double sqrt(double);

then the call

root2 = sqrt(2);

coerces the integer 2 into the double value 2.0 without any need for a cast.