Type Conversion

Type Conversion

- C provides two methods of changing the type of an expression:
  - Type conversion (done implicitly)
  - Cast expressions (done explicitly)

- Examples of implicit conversion:

```
float x;
int i;
if (x < i)  ... /* Example 1 */
x = i; /* Example 2 */
```

In each case, i is implicitly converted to float type.

The Usual Arithmetic Conversions

- **The usual arithmetic conversions are applied to operands in an arithmetic or logical expression.**
- **Operands are converted to the “smallest” type that will safely accommodate both values.**
- **This is often done by converting (promoting) the operand of the “smaller” type to the type of the other operand.**

The Usual Arithmetic Conversions

- **Examples of the usual arithmetic conversions:**

```
char c;
short s;
int i;
long int l;
float f;
double d;
long double ld;
i = i + c; /* c is converted to int */
i = i + s; /* s is converted to int */
l = l + i; /* i is converted to long */
f = f + l; /* l is converted to float */
d = d + f; /* f is converted to double */
ld = ld + d; /* d is converted to long double */
```

Conversion During Assignment

- The usual arithmetic conversions do not apply to assignment
  Instead, the value on the right side of the assignment is conv
  to the type of the variable on the left side.

- *Warning*: Converting from a "large" type to a "small" type may not always produce a meaningful result:

```
int i;
float f;
i = f;
```

This assignment is meaningful only if the value of `f`—when converted to an integer—lies between the smallest and largest values allowed for an `int` variable. If `f` is too large or too small, `i` will be assigned an apparently meaningless number.

Conversion During Assignment

- The conversion of a floating-point number to an integer is done by dropping the fractional part of the number (not by rounding to the nearest integer):

```
int i;

i = 842.97; /* i is now 842 */
i = -842.97; /* i is now -842 */
```

Casting

- Forces one type into another
- For example,

```
yourFloat = (float) myinteger ;
```

- Will force the integer myInteger into a float, yourFloat
- `iMyInteger = (int) fYourFloat ;`
  - Will work too, but you'll lose the decimal portion
  - i.e. 1.9 forced into an integer will result in 1

Rules of Promotion (cont'd)

Use a Cast Operator

```
int num = 4;
float avg = ( 90 + 92 + 95 + 100 ) /
(float) num;
```

In this case, num is explicitly *cast* to a float. And, because we have one float, everything else is promoted.