# Data Types

## Need for Many Types

- Different types are needed because one type is not suitable for representing data of another type.
- Mixing types may result in precision loss, overflow, underflow and ability to process full range.
- Application performance suffers while performing numerically intensive computation if inappropriate data types are used.
- Exceptions must be handled excplicitly or they lead to errors.
- So, use of appropriate type is important both for efficiency and correctness.
  - Eg., `short int` can be used if range restricted to [-$2^{15}$, $2^{15}$-1].
  - For positive integers `unsigned int` is more appropriate.

# Data Types

## Integer Types

- Basic numeric types: whole numbers (`int`) & having fractional part (`float`).
- Two different int types: signed and unsigned
- Maximum signed int in 16 bit: 0111111111111111, i,e., $2^{15} - 1$
- Maximum unsigned int in 16 bit: 1111111111111111, i.e., $2^{16} - 1$
- Possible types to suit our needs are: `short int`, `unsigned short int int`, `unsigned int`, `long int`, `unsigned long int`.
- C allows short hand to drop `int`

# Data Types

## Floating Point Types

- Different types: `float, double, long double`
- `double` is used for precision critical calculations (upto 15 digits against 6 digits).
- By default floating point constants are stored as a `double`. To force `float` constant should be suffixed with $f$, i.e., 7.5f or 7.5F.
- Format specifier "%lf", "%Lf" are used for reading `double` and `long double`.

# Data Types

## Details of Different Types

| Type | Size in bytes | Range |
|---|---|---|
| char | 1 | -128 to 127 |
| unsigned char | 1 | 0 to 255 |
| short int | 2 | -32768 to 32767 |
| unsigned short int | 2 | 0 to 65535 |
| int | 4 | -214743648 to 214743647 |
| unsigned int | 4 | 0 to 4294967295 |
| float | 4 (s+8e+23m) | (approx) $\pm$ [$10^{-38}$, $10^{38}$] |
| double | 8 | (approx) $\pm$ [$10^{-308}$, $10^{308}$] |

Note: for 32-bit m/c `long int` and `int` are same.

# Data Types

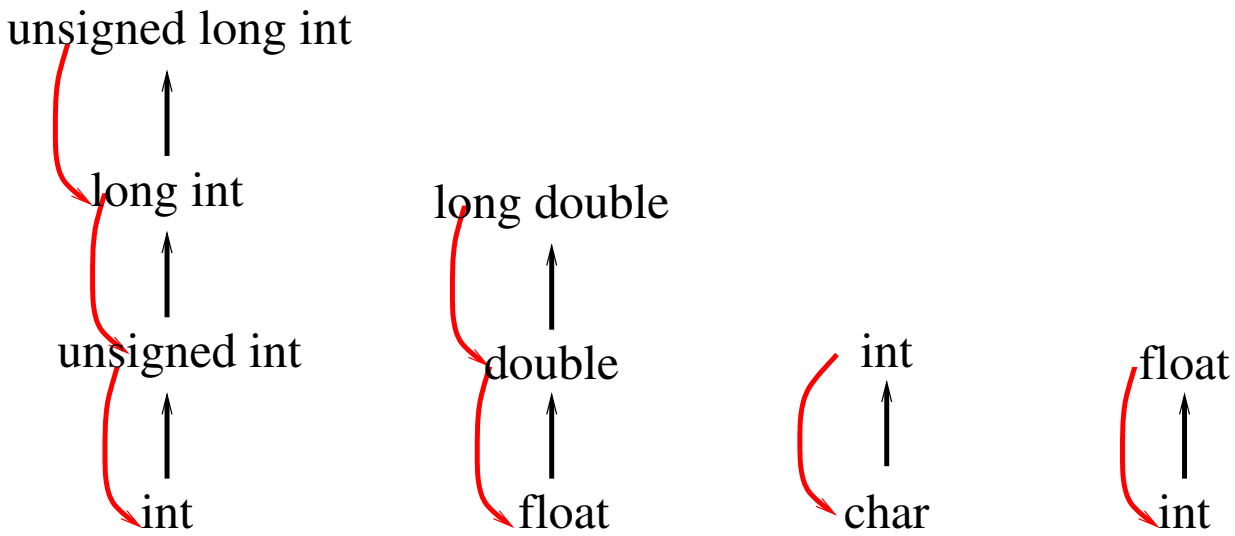
## Char Types

- ASCII character set is most widely used: each character requires 7 bits.
- Digits 0-9 represented by 0110000 – 0111001
- A-Z represented by 1000001 – 1011010
- Characters are treated as small integers, so it is possible to operate (add, substract, compare) on them.
- For example converting lower case to uppercase:

```
if ('a' <= ch <= 'z') ch = ch - 'a' + 'A';
```

# Type Conversions

## Coercion Rules



# Type Conversions

## Example 11

```
char c;
short int s;
int i;
long int l;
unsigned int u;
unsigned long int ul;
float f;
double d;
long double ld;
```

# Type Conversions

## Example 11 (contd)

```
i = i + c       // c  promoted to int
i = i + s       // s  promoted to int
u = u + i       // i  promoted to unsigned int
l = l + u       // u  promoted to long int
ul = ul + l     // l  promoted to unsigned long int
f = f + ul      // ul promoted to float
d = d + f       // f  promoted to double
ld = ld + d     // d  promoted to long double
```