

Numeric Types in C++

- Data types representing numbers
- Binary form internally
  - In almost all current computers
- Integer vs floating-point

---

---

---

---

---

---

---

---

Decimal Numbers

100's → **374** ← 1's

10's

$$\sum_{i=0}^{n-1} a_i \cdot 10^i \quad \text{base} = 10$$


---

---

---

---

---

---

---

---

Binary Numbers

**01001001**

64's 8's 1's

$$\sum_{i=0}^{n-1} a_i \cdot 2^i \quad \text{base} = 2$$


---

---

---

---

---

---

---

---



Integer Numeric Range

Unsigned: 0...255  
 $0 \dots (2^n - 1)$   
 Signed: -128...+127  
 $-(2^{n-1}) \dots +(2^{n-1} - 1)$

---

---

---

---

---

---

---

---

Integer Types

<u>Signed</u>	<u>Unsigned</u>
short int	unsigned short int
int	unsigned int
long int	unsigned long int

Actual lengths (bits) may vary by system and compiler (limits.h?).

---

---

---

---

---

---

---

---

Numeric Operators & Precedence

See Appendix C for more information.

<u>Operator</u>	<u>Meaning</u>
( )	Grouping (extra parens OK)
-, +	Unary minus, plus
*, /, %	Multiply, divide, modulus
+, -	Addition, subtraction
=, *=, /=, %=, +=, -=	Assignment

---

---

---

---

---

---

---

---

Division & Modulus

- **Integer division produces two results**

- Quotient
- Remainder

$7/3 \rightarrow 2$
$7\%3 \rightarrow 1$

- **C++: one operator for each**

- “/” for quotient, “%” for remainder

---

---

---

---

---

---

---

---

Another Integer Type: char

- **Represents a single character**

- Often in ASCII/ANSI/ISO code
- E.g., ‘A’ is decimal 65

- **One byte in size**

- Other, wider, character types exist

---

---

---

---

---

---

---

---

Floating-Point Types

- **Differ from integer types**

- Can represent fractional parts
- Wider value range

- **C++ types**

- float, double, long double
- Literal format:  $-23.45e-4$

$$-23.45 \times 10^{-4}$$

---

---

---

---

---

---

---

---

⋮  
Floating-Point Operations

- **Operators**
  - Same as integers (except modulus)
- **Standard math library**
  - Mathematical functions
  - Must reference: #include <math.h>
- **Round-off errors**

---

---

---

---

---

---

---

---

⋮  
Example: Quadratic Roots

```
#include <math.h>
...
float a,b,c; // Coef. (a not zero)
float root1;
float root2;
float discr = b*b -4*a*c;
if (discr >= 0.0)
{
  root1 = (-b+sqrt(discr))/(2*a);
  root2 = (-b-sqrt(discr))/(2*a);
}
```

---

---

---

---

---

---

---

---

⋮  
Selection Statement (Intro)

- **Tests a condition**
  - If true, executes a statement
  - If false, does not execute it
- **Syntax**
  - if (condition) statement-T
- **More in chapter 4**

---

---

---

---

---

---

---

---

Polymorphism & Conversions

- **Numbers come in several different forms**
  - E.g., integer, floating-point
- **Same operators for all**
- **Can even mix numeric types**
  - Requires conversion between types

---

---

---

---

---

---

---

---

Numeric Conversions

- **Integer to floating**
  - Usually no problem
    - May be too big for exact conversion
- **Floating to integer**
  - Fractional part lost
  - Value may be too big for integer




---

---

---

---

---

---

---

---

Conversion Examples (1)

```
int x = 3;
int y = 5;
double r = 23.4;
float s = 5;
r += x;
```

s =  $x / y;$

Division first,  
then conversion

s =  $3.0 / y;$

Conversion first,  
then division

---

---

---

---

---

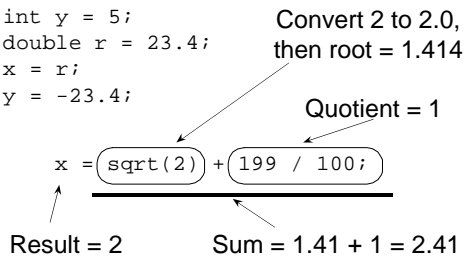
---

---

---

Conversion Examples (2)

```
int x = 3;  
int y = 5;  
double r = 23.4;  
x = r;  
y = -23.4;
```



---

---

---

---

---

---

---

---

Type Casting

- **So far, automatic**
  - Conversions done as needed
- **Can convert explicitly**
  - Use type name as if it were function
    - E.g., `r = double(y) / 2;`
  - Other forms possible (later)
  - Casting can be dangerous

---

---

---

---

---

---

---

---