Data Types

Alark Joshi
Introduction

- A **data type** defines a collection of data objects and a set of predefined operations on those objects
- A **type system**
  - Defines how a type is associated with each expression in the language
  - Includes rules for type equivalence and type compatibility
Introduction

• Design issues for data types:
  o What operations are defined and how are they specified?
  o Can users define new types
Definitions

• A descriptor is the collection of the attributes of a variable

• An object represents an instance of a user-defined (abstract data) type

• A design issue for all data types
  o What operations are defined and how are they specified?
Theory and Data Types

• Type theory is a broad area of study in mathematics, logic, computer science, and philosophy

• Two branches of type theory in computer science:
  o Practical – data types in commercial languages
  o Abstract – typed lambda calculus
Type Systems

• A type system is a set of types and a collection of functions that define the type rules
  o Either an attribute grammar or a type map could be used for the functions
  o Finite mappings – model arrays and functions
  o Cartesian products – model tuples and records
  o Set unions – model union types
  o Subsets – model subtypes
Primitive Data Types

• Almost all programming languages provide a set of *primitive data types*
• Primitive data types: Those not defined in terms of other data types
• Some primitive data types are merely reflections of the hardware
• Others require only a little non-hardware support for their implementation
Primitive Data Types: Integer

- Almost always an exact reflection of the hardware so the mapping is trivial
- There may be as many as eight different integer types in a language
  - Java’s signed integer sizes: byte, short, int, long
  - C and C++ have these plus a set of corresponding unsigned types
- Scripting languages generally have one integer type
Representing Integers

• Positive numbers can be converted to base 2 – [e.g. represent 7, 12, 14]

• How do you represent the sign when you only have 0s and 1s?
  o Sign bit
  o Ones complement
  o Twos complement
Using a Sign Bit

• Use one bit of the representation for the sign

• Unsigned data types don’t have a sign bit
  o unsigned char can represent 0-255 values
  o signed char can represent
One’s Complement Representation

- Negative numbers are the complement of the corresponding positive number.
Two’s Complement

• Take the complement and add one
• This representation is continuous from -1 to 1
Exercise

• Represent the following numbers using
  1. Sign bit
  2. One’s complement
  3. Two’s complement

• 7, -10, -8, 0, -1

• Check with your teammates
Floating Point Types

• Model real numbers, but only as approximations
• Languages for scientific use support at least two floating-point types - `float` and `double` (sometimes more)
• Usually exactly like the hardware, but not always
Representing Real numbers

• Decimal numbers can also be converted to base 2
• Now we need a way to represent both the sign and the decimal point
  o Use one bit for the sign
  o For the decimal point two possibilities
    • Always have the same number of bits before and after
    • Use scientific notation – $1.1 \times 10^2 = 110$
IEEE Floating-Point Standard 754

- Use binary scientific notation
- Normalize the binary representation
  - One bit before the decimal point (always a 1)
- First bit is sign
- Use fixed number of bits for exponent and rest for fractional part of mantissa
  - \( \exp = \text{exponent} - \text{offset} \)
  - \( (-1)^{\text{sign}} \times \text{fraction} \times 10^{\exp} \)