Variable Typing Strategies

Robert Lowe

Department of Computer Science
Southeast Missouri State University

Outline

Anatomy of a Variable

- Variables carry **attributes**.
- Attributes are **bound** to variables.
- Attributes that are statically bound are typically only stored at compile time.

Common Variable Attributes

- Name
- Address
- Type
- Value

Variable Types

- A variable's **type** is an attribute that describes its range of values and operations.
- The variable type system of a language reveals a lot about its semantic behavior.
- Languages should check for type errors and/or perform **type coercion** in operations.

Type Coercion Examples

- C/C++
  - *double* + *int* → *double*
  - *double* ∗ *int* → *double*
  - *double*/*int* → *double*
- Javascript
  - *string* + *numeric* → *string*

Variable Typing Throughout History

- Early machine code languages provided no typing.
- In the 1950's, FORTRAN provided integer, real, and character typing.
  - Name Based Types: I, J, K, L, M, N are integers. All others real.
  - Explicit typing to override name typing.
- In the 1960's, ALGOL introduced fully declarative typing and user-defined types.
- COBOL implemented typing by image.
- 1970's, Strong typing with enumerated types and record types.
- 1980's to present saw the rise of user-defined abstract types.

Strong Variable Typing

Definition

Strong Typing is when a language reports all type errors.

- Most languages are not fully strongly typed.
- C/C++ tend toward strong typing, however they have union types, which are not checked.
- Strong typing exists on a spectrum.

Numeric Types

Integer	- Stores a whole/integral number. Usually supported directly by hardware. Represented internally as a raw binary number. Can be signed or unsigned.
Floating Point	- Usually represented using the IEEE 754 representation. These are often supported by hardware, but sometimes implemented in software. They have a high degree of error.
Decimal	- Fixed precision decimal number. Usually represented as BCD (Binary Coded Decimal). These are exact with no round-off error.
Boolean	- Usually represented by a single byte, with 0 representing false and other values representing true.

Character Types

Character - A single character. Often this is how a programming language represents an individual byte.

String - A collection of characters. Sometimes additional attributes are stored.

Reference Types

Pointer - A variable which stores the address of a value in memory. Often has type attributes associated with it.

Reference - An abstraction of a pointer. Provides an alias to an actual variable, or it can provide reference to a complex type (as in Java)

Enumerated Types

- Types with a fixed range of labeled values.
- For example, in C:
```
enum day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};
enum day d = TUESDAY;
```
- Usually implemented as integers, internally.
- Some languages perform error checking on the values.

Non-Scalar Types

- Non-Scalar types store lists and groups of information.
- **Arrays** - Homogeneous sequence of items.
- **Associate Arrays** - Associates a key/value pair. Usually heterogeneous.
- **Lists** - Usually internally a doubly linked list, can be heterogeneous.
- **Tuple** - An immutable list.

User-Defined Types

- First popularized in the ALGOL languages.
- Data types derived through combinations of primitive types.

Record Types

- A record is formed out of several fields.
- Example: structs in C/C++:

```
struct point {
    double x;
    double y;
};
struct point p;
p.x = 0;
p.y = 0;
```

- A record type has all of its fields present at all times. Its size is at least as big as the sum of the sizes of its fields.

Union Types

- A union is a data structure that can take on one of several types.
- Defined much like a record, but only one field contains data at a given point in time.
- Example: union in C

```
union eval_result {
    int i;
    double d;
    float f;
    char c;
    void *ptr;
};
```

Abstract Types

- Abstract types include relationships between types.
- The most common system of abstract types is Object Oriented Programming.

Object Oriented Programming

Object

An **object** is any entity with state and behavior.

Elements of Object Oriented Programming

Abstraction	- Objects act like black boxes. Details are hidden.
Encapsulation	- An object maintains its own internal state. It carries all necessary data within itself.
Inheritance	- An object can be generalized by another object.
Polymorphism	- Code can be written to the most general case. An object can be referenced by any time from which it descends.

Reading and Reference

- Read Chapter 6 – Data Types
- IEEE 754 Format: https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/amp/
- BCD Format: https://www.geeksforgeeks.org/bcd-or-binary-coded-decimal/