

Data Types and Variables

Variable :

Abstractions in a programming language for the memory cells of the machine.

Important attributes :

1. name (identifier)
2. Address
3. Type
4. Value

Name :

String of characters used to identify some entity in a program.

Design issues :

1. Maximum length
2. Connectors : ‘_’
3. Case-sensitivity : Good or bad ?
4. Keywords : special when used in right context.
5. Reserved words : special word in a programming language, cannot be used as a name.

Address

Address :

Memory address with which the variable is associated.

Same name can have different addresses :

1. Two functions f_1 and f_2 , each defines a variable *foo*.
2. Recursive functions; same name associated with different addresses at different times.
3. Different function invocations.

Alias : more than one variable used to access same memory location

1. FORTRAN : EQUIVALENCE (A,B)
2. C, C++ : union variables, pointers.

Type and Value

Type :

This is an abstraction. Determines the range of values the variable can have and a set of operation for those values.

Example :

In some FORTRAN implementation
INTEGER type can range from -32,768 to
32,767.

Operations; +, -, *, /

Value :

Contents of the memory cell or cells associated
with the variable

Two types of value associated with a variable

1. *r_value* : actual value
2. *l_value* : address

Binding

Binding :

Association between an entity and an attribute or value.

Binding Time :

Time at which the association happens.

Different types of bindings:

1. Language design time : Symbol $*$, bound to multiplication operation
2. Language implementation time : INTEGER in FORTRAN bound to possible range.
3. Compile time : Variable in C or Pascal bound to particular data type
4. Link time : Subprogram call
5. Load time : Variable may bound to storage cell
6. Runtime : Variable in C functions or Pascal subprograms

Type Bindings

Before variable can be referenced in must be bound to a data type.

Static binding

- Explicit : Statement in a program lists variable to be of a particular type.

Example in C: `int i,j;`

- Implicit : Associating types with variables through default conventions.

Example in FORTRAN: variables starting with i,j,k,l,m,n are integers

Example in ML: `fun circum(r) = 3.14 * r * r;`
type is inferred by checking constant.

Dynamic binding

- Variable is bound to type when assigned a value.
Variable is assigned the type of the value.
- Example:
 - LISP : `(setq foo 13); (setq foo 'abc)`
 - APL : `LIST ← 1.0 2.0 3.0; LIST ← 12`
- Advantage : Flexibility.
- Disadvantage : Error detection hard; incorrect programs.

Storage Bindings and Lifetime

Binding a variable to a memory cell taken from a pool of available memory is called *allocation*

Deallocation is the removal of a binding from the memory cell

Lifetime : Time during which a variable is bound to a specific memory location.

- Static : From before execution to end.
Example in C: `int i,j; (global)`
- Semi-dynamic : Created when declaration is elaborated; but with statically bound type
Example in C, Pascal: local vars.
- Explicit dynamic : Nameless, storage is allocated and deallocated by explicit run-time operations.
Example in C, Pascal: `malloc, new, free`.
- Implicit dynamic : Bound to storage only when assigned a value. Example in LISP, ML