Fortran 95+ → can declare >31 char names

C-99 → 63-character names, any higher means others truncated internally
C++ → all characters
(But you can write more readable programs)

PHP → $

Reserved words
↳ can't be used as names
too many ⇒ it becomes difficult for a programmer

Keyword, special only in certain contexts
Real Apple; → type specifier } fortran still
Real = 3.4 → variable } maintain keywords

Variables

- A *variable* in a program is an abstraction of a computer memory cell or collection of cells.
- Programmers often think of variables as names given to memory cells.

In machine language memory locations have binary addresses.
In Assembly language we can give names (more readable)
A variable is characterized by following attributes (is a six-tuple):

- Name
- Address (l-value)
- Content (r-value)
- Type
- Lifetime
- Scope

Names are associated with any variable

Address of a variable is the machine memory addr. with which it's associated

address of var is also known as its l-value

- Address of variable is also known as its *l – value*, because the address is what is required when the name of variable appears on the LHS of assignment.

Ex: sum= total;

- It is possible t have multiple variables have same address. In such case the variables are called as *aliases*.
- It can create problems. It sometimes reduces the readability.
- Ex: Unions, Pointers, parameters, object references etc.

formal & actual

Unions → stores enough space for largest datatype in it. an example of Aliasing! (its internal variables stored in same space)

→ meh, these just point to same loc, actually stored in diff. loc.

LVALUE = ADDR

RVALUE = VALUE → value stored in mem loc

eg: sum = total (sum: address is taken; total: value is stored in there)

The Concept of Binding (5.4)

- It is an association between an attribute and an entity
  Ex: variable and type/value
  Operation and symbol
- The time at which binding takes place is called as *binding time*.

The binding and the binding time are the prominent features in the semantics of a programming language.

* Language design time (not program design) — * → multiplication
* Lang implementation time — int → range (-2^{32} to $2^{32}-1$)
* Compile time binding — binding of vars to datatypes
* Load time binding — variable binding to mem loc
* Link time binding — call library subprogram (binding to subprogram code happens at link time)
* Run time — binding variables to memory or values assigned to variables etc.

Static binding

→ Specified by a declaration statement

⇒ Slides/book

Dynamic binding

bound to a type when it is assigned a value in assignment statements

Cannot be inferred by name

Can cause a problem at times