Imperative Sprachen

Variablen: charakterisiert durch 6-Tupel

- Name
- Adresse
- Typ
- Wert
- Scope
- Lebenszeit

Namen:

- Case-sensitiv ?
- Reservierte Worte /Schlüsselworte ?
- Länge, signifikante Zeichen?

Imperative Sprachen

Case-Sensitivität:

- C-basierte Sprachen sind case-sensitiv: rose ≠ Rose ≠ ROSE
- verletzt gute Lesbarkeit
- verletzt gute Schreibbarkeit (Bsp. parseInt)

- Pascal ist nicht case-sensitiv: i = I , funktion1 = Funktion1
- Lisp ist nicht case-sensitiv: alle Bezeichner werden in Großbuchstaben umgewandelt

Imperative Sprachen

Reservierte Worte / Schlüsselworte:

- Schlüsselworte: haben nur an best. Stellen vordefinierte Bedeutung
- Beispiel (Fortran): Real x
  Real = 14.5
- → schlechte Lesbarkeit

- Reservierte Wörter: nicht für eigene Bezeichner erlaubt
- Beispiel Cobol: über 300 reservierte Wörter (u.a. LENGTH, COUNT,…) → schwierig eigene erlaubte zu finden

- Zwischenweg: vorbelegte Namen, die überschrieben werden dürfen
- Beispiel Ada: vordefinierte Namen Integer, Float
  sind überschreibbar → auch nicht gut für die Lesbarkeit…

Imperative Sprachen

Länge, signifikante Zeichen:

Beispiele:

- Fortran 95: Namen aus bis zu 31 Zeichen
- C (89): keine Längenbegrenzung, nur erste 31 sind signifikant
- C (99): keine Längenbegrenzung, nur erste 63 sind signifikant
- Java, Ada, C#: keine Längenbegrenzung, alle signifikant
- C++: keine Längenbegrenzung; manche C++-Implementierungen begrenzen, damit Symboltabelle nicht zu groß wird

Imperative Sprachen

Erlaubte Zeichen:

- meist 1. Zeichen Buchstabe, gefolgt von Buchstaben, Ziffern und _
- in 70ger u. 80ger Jahren: _ extrem viel genutzt: my_stack
- heute: Kamel-Notation myStack, sucheGroesstesElement
  → KEINE Design-Entscheidung, sondern Programmierstil !!
- Fortran 90: Blanks in Namen erlaubt: my Stack equivalent zu myStack

Imperative Sprachen

Adresse:

- kann während Programmausführung wechseln
- l-Value genannt: linke Zuweisungsseite benötigt Adresse
- mehrere Variablen können gleiche Adresse haben (Aliasing)
  → schlecht für Lesbarkeit

Typ:

- legt Wertebereich fest
- legt Grundmenge an erlaubten Operationen auf Variablen fest

Imperative Sprachen

Wert:

- Inhalt der Speicherzelle an entsprechender Adresse
- r-Value genannt: Wert einer Variablen wird auf rechter Zuweisungsseite benötigt
- um auf r-Value zuzugreifen muss man erst l-Value bestimmen

Scope:

- Anweisungsbereich, in dem Variable sichtbar ist
- durch ALGOL 60 eingeführt: static scope → Scope zur Compilezeit bestimmbar
- in APL, Snobol4, Lisp: dynamic scope → Scope erst zur Laufzeit bestimmbar
- Perl, Common Lisp heute: Programmierer kann dyn.scope wählen

Imperative Sprachen

Lebenszeit:

- ≠ scope
- Zeitraum in dem Variable Speicherplatz belegt
- Lebensdauer lokaler Variablen: vom Funktionsaufruf bis Verlassen der Funktion
- Lebensdauer globaler Variablen: ganze Programmdauer

- statische lokale Variablen (z.B. C): Lebensdauer wie globale, Gültigkeitsbereich: nur lokale Funktion