Språkspecifikation vers. 2.0

Viktor Norlin och Warren Crutcher

26 February 2023

1. Generella egenskaper

- Språket ska, om möjligt, vara imperativt, interpreterat och statiskt typat.
- Språket ska stödja de "vanliga" datatyperna, så som float, int, bool etc.
- Språket ska ha stöd för funktioner. Funktioners returtyp ska vara hårt typat.
- Språket är "general purpose", alltså har det inget specifikt syfte utöver att vara ett generellt programspråk.

2. Nedan följer några kodexempel på hur språket skrivs.

Deklarera variabler

- let age -> int = 30;
- let height -> float = 5.6;
- let name -> string = "John Doe";

Funktion som tar två int som argument och returnerar en int

- let add -> (num1 -> int, num1 -> int) int {
  ...
  }

Funktion som tar inga argument och inte returnerar något

- let print -> () void {
  ...
  }

Styrstruktur - if, for och while

- let x -> int 10;
- let x -> int 5;
- if (x > y) {
  print(x);
  } else {
  print(y);
  }
- for (let i -> int = 0; i < 10; ++i) {
  ...
  }

- for (i -> int in 1 to 10) {
  ...
  }
- while (1) {
  ...
  }

3. Felhantering

Vi anser att felhanteringen kommer vara en väldigt viktig del av språkets utveckling och användbarhet. Just nu är vi osäkra om detaljarna angående s.k. "Stack Traces" men vi planerar utveckla den delen löpande under språkets utveckling.

4. Grammatik

Arithmetic and logical grammar

- <stmt> ::= <term> | <assign>
- <term> ::= <factor> | <term> <addop> <factor> | <term> <orop> <factor>
- <factor> ::= <atom> | <factor> <mulop> <atom> | <factor> <andop> <atom>
- <atom> ::= <number> | <bool> | <variable> | "(" <term> ")" | <addop> <atom> | <notop> <atom>
- <number> ::= <integer> | <real>
- <integer> ::= <digit> | <digit> <integer>
- <real> ::= <integer> "." <integer>
- <digit> ::= [0-9]
- <variable> ::= <letter> | <letter> <variable>
- <letter> ::= [a-z] | [A-Z]
- <bool> ::= "true" | "false"
- <addop> ::= "+" | "-"
- <mulop> ::= "*" | "/"
- <orop> ::= "||"
- <andop> ::= "&&"
- <notop> ::= "!"
- <assign> ::= <variable> "=" <term>

Alternative arithmetic and logical grammar

- <stmt> ::= <term> | <assign>
- <term> ::= <arithmetic_term> | <logical_term>
- <arithmetic_term> ::= <arithmetic_factor> | <arithmetic_term> <addop> <arithmetic_factor>
- <arithmetic_factor> ::= <arithmetic_atom> | <arithmetic_factor> <mulop> <arithmetic_atom>
- <arithmetic_atom> ::= <number> | <variable> | "(" <arithmetic_term> ")" | <addop> <arithmetic_atom>
- <logical_term> ::= <logical_factor> | <logical_term> <orop> <logical_factor>
- <logical_factor> ::= <logical_atom> | <logical_factor> <andop> <logical_atom>
- <logical_atom> ::= <bool> | <variable> | "(" <logical_term> ")" | <notop> <logical_atom>
- <number> ::= <integer> | <real>

- <integer> ::= <digit> | <digit> <integer>
- <real> ::= <integer> "." <integer>
- <digit> ::= [0-9]
- <variable> ::= <letter> | <letter> <variable>
- <letter> ::= [a-z] | [A-Z]
- <bool> ::= "true" | "false"
- <addop> ::= "+" | "-"
- <mulop> ::= "*" | "/"
- <orop> ::= "||"
- <andop> ::= "&&"
- <notop> ::= "!"
- <assign> ::= <variable> "=" <term>

Problemet med grammatiken ovan är att en variabel kan vara "false", så uttryck som "false=true" blir valid. Även uttryck som "21+false" är valid, då "false" i detta fallet räknas som en variabel och inte en faktisk bool.

Kopior vs Referenser

Vi vill att alla variabler som skickas till en funktion är referenser och inte kopior. Vi vet dock inte om det blir svårare eller lättare att implementera. Det känns bara mer robust att om ett värde skickas till en funktion för att muteras att värdet faktiskt ändras.

Däremot vill vi att det händer per automatik och att programmeraren inte behöver bestämma manuellt (som i C eller C++)

Scope

Språket ska var väldigt statiskt skopat. Globala variabler är inte tillåtet. Vi tycker att detta kommer göra den funktionella sidan av språket mer ren och minska sidoeffekter. Tanken är att vi har nästlade Hash där variable namn eller funktionsnamn blir nyckel och värdet på variabler är nyckelns värde, samt funktionskropp blir värde till en funkitonsnamns nyckel.

Vi tycker överlag att vi har svårt att spendera tiden som krävs för att göra en djupgående utredning om det här just nu i och med att vi har fullt upp med nätverkskursen och duggan för TDP007 nu på onsdag. Efter vi har gjort färdigt både duggan och labbarna i nätverkskursen kommer vi har mer tid att dyka djupare i de här frågorna.