Programski jezik PINS'24

1 Leksikalna pravila

Programski jezik PINS'24 je pisan v abecedi ASCII in vsebuje naslednje leksikalne enote:

- *Konstante*:
  - *število*:
    Neprazno zaporedje desetiških števk, pred katerimi lahko stoji predznak (`+` or `-`).
  - *znak*:
    Znak zapisan v enojnih navednicah (`'`). Znak je lahko (a) katerikoli ASCII znak s kodo v obsegu $\{32...126\}$, pri čemer morata biti enojna navednica in obratna poševnica (`\`) uvedeni z obratno poševnico, (b) znak za konec vrstice, ki je zapisan kot `\n`, ali (c) katerikoli ASCII znak zapisan s kodo v obliki `\XX`, pri čemer je `X` šestnajstiška števka (`0...9` in `A...F`).
  - *niz znakov*:
    Niz znakov, lahko prazen, zapisan v dvojnih navednicah (`"`). Znak je lahko (a) katerikoli ASCII znak s kodo v obsegu $\{32...126\}$, pri čemer morata biti dvojna navednica in obratna poševnica (`\`) uvedeni z obratno poševnico, (b) znak za konec vrstice, ki je zapisan kot `\n`, ali (c) katerikoli ASCII znak zapisan s kodo v obliki `\XX`, pri čemer je `X` šestnajstiška števka (`0...9` in `A...F`).
- *Simboli*:
  `= , && || ! == != > < >= <= + - * / % ^ ( )`
- *Imena*:
  Neprazno zaporedje črk (`A...Z` in `a...z`), desetiških števk (`0...9`) in podčrtajev (`_`), ki (a) se začne s črko ali podčrtajem in (b) ni ključna beseda.
- *Ključne besede*:
  `fun var if then else while do let in end`
- *Komentarji*:
  Niz znakov, ki se začne z ograjico `#` in se konča na koncu vrstice.
- *Belo besedilo*:
  Presledek in znaki HT, LF in CR. Konec vrstice označuje znak LF, HT je širok 8 znakov.

Leksikalni elementi morajo biti razpoznani od leve proti desni po pravilu najdaljšega ujemanja na skrajno levem mestu.

2 Sintaksna pravila

Sintakso programskega jezika PINS'24 določa kontekstno neodvisna gramatika z začetnim simbolom *program* in naslednjimi produkcijami:

program
$\longrightarrow$ (*definition*)$^+$

definition
- $\longrightarrow$ `fun` IDENTIFIER `(` *parameters* `)`
- $\longrightarrow$ `fun` IDENTIFIER `(` *parameters* `)` `=` *statements*
- $\longrightarrow$ `var` IDENTIFIER `=` *initializers*

parameters
- $\longrightarrow$ $\big($ IDENTIFIER $\big($ `,` IDENTIFIER $\big)^*$ $\big)^?$

statements
- $\longrightarrow$ *statement* $\big($ `,` *statement* $\big)^*$

statement
- $\longrightarrow$ *expression*
- $\longrightarrow$ *expression* `=` *expression*
- $\longrightarrow$ `if` *expression* `then` *statements* $\big($ `else` *statements* $\big)^?$ `end`
- $\longrightarrow$ `while` *expression* `do` *statements* `end`
- $\longrightarrow$ `let` $\big($ *definition* $\big)^+$ `in` *statements* `end`

expression
- $\longrightarrow$ INTCONST $\big|$ CHARCONST $\big|$ STRINGCONST
- $\longrightarrow$ IDENTIFIER $\big($ `(` *arguments* `)` $\big)^?$
- $\longrightarrow$ prefix-operator *expression*
- $\longrightarrow$ *expression* pstfix-operator
- $\longrightarrow$ *expression* binary-operator *expression*
- $\longrightarrow$ `(` *expression* `)`

arguments
- $\longrightarrow$ $\big($ *expression* $\big($ `,` *expression* $\big)^*$ $\big)^?$

initializers
- $\longrightarrow$ $\big($ *initalizer* $\big($ `,` *initializer* $\big)^*$ $\big)^?$

initializer
- $\longrightarrow$ $\big($ INTCONST `*` $\big)^?$ *const*

const
- $\longrightarrow$ INTCONST $\big|$ CHARCONST $\big|$ STRINGCONST

Simboli INTCONST, CHARCONST in STRINGCONST označujejo celoštevilske konstante, znakovne konstante in nize, zaporedoma.

Prioriteta operatorjev je sledeča

postfiksni operatorji	`^`	NAJVIŠJA PRIORITETA
prefiksni operatorji	`! + - ^`	
multiplikativni operatorji	`* / %`	
aditivni operatorji	`+ -`	
primerjalni operatorji	`== != < > <= >=`	
konjunkcija	`&&`	
disjunkcija	`||`	NAJNIŽJA PRIORITETA

Multiplikatini in aditivni operatorji so levo asociativni,primerjalni operatorji niso asociativni.

3 Semantična pravila

Imena. Vsa imena so v istem imenskem prostoru.

Vsako ime je vidno v celotnem območju dosega, v katerem je definirano. Območje dosega je ustvarjeno na tri načine:

1. Vse definicije imen zunaj funkcij so v globalnem območju dosega.
2. Funkcija ustvari novo območje dosega (ime funkcije je zunaj ustvarjenega
3. Stavek `let-in` ustvari novo območje dosega: vse definicije območja obsega).

Tipi. Obstaja en sam podatkovni tip: 32-bitno predznačeno število v dvojiškem komplementu, ki služi kot vrednost in kot naslov. Znakovne konstante predstavljajo cela števila glede na ASCII kodiranje. Nizi znakov predstavljajo naslov, na katerem so shranjena števila, ki so dobljena iz znakov niza glede na ASCII kodiranje.

Zaporedje stavkov, ki tvori telo funkcije, se mora končati

1. s stavkom, ki vsebuje zgolj izraz, ali
2. z `let-in` stavkom, katerega zaporedje stavkov se konča s stavkom pod točko 1 ali 2.

Leve vrednosti. Leva vrednost je

1. spremenljivka ali
2. izraz, katerega najbolj zunanji operator je postfiksni `^`.

Operacijska semantika.

1. V prireditvenem stavku se najprej izračuna vrednost na desni strani, potem naslov na levi strani.
2. Pogoj v pogojnem stavku in zanki je izpolnjen, če je njegova vrednost različna od 0.
3. Pri izračunu izraza z dvomestnim operatorjem se najprej izračuna desni, nato levi argument.
4. Pri klicu funkcije se argumenti izračunajo od desne proti levi.

Skladovni stroj. Skladovni stroj je 32-biten: naslovi, celoštevilske vrednosti in registri so 32-bitni.

Skladovni stroj ima tri registre:

1. programski števec (`IP`),
2. skladovni kazalec (`SP`),
3. klicni kazalec (`FP`).

Ukazi skladovnega stroja so naslednji:

1. `LOAD`: z vrha sklada vzame naslov in na vrh sklada naloži vrednost, ki je na tem naslovu.

   $\texttt{addr} \leftarrow \texttt{MEM[SP]}, \texttt{SP++}$
   $\texttt{value} \leftarrow \texttt{MEM[addr]}$
   $\texttt{SP--}, \texttt{MEM[SP]} \leftarrow \texttt{value}$

2. `SAVE`: z vrha sklada vzame naslov in vrednost ter shrani vrednost na ta naslov.

   $\texttt{addr} \leftarrow \texttt{MEM[SP]}, \texttt{SP++}$
   $\texttt{value} \leftarrow \texttt{MEM[SP]}, \texttt{SP++}$
   $\texttt{MEM[addr]} \leftarrow \texttt{value}$

3. `POPN`: z vrha sklada vzame vrednost n in
   z vrha sklada umakne $|\mathrm{n}|$ vrednosti (če $\mathrm{n} \geq 0$)
   ali na vrh sklada naloži nanj $|\mathrm{n}|$ vrednosti 0 (če $\mathrm{n} < 0$).

   $\mathrm{n} \geq 0 \Longrightarrow \forall\, \mathrm{i} = 1 \ldots |\mathrm{n}/4|$: `SP++`
   $\mathrm{n} < 0 \Longrightarrow \forall\, \mathrm{i} = 1 \ldots |\mathrm{n}/4|$: `SP--`, `MEM[SP]` $\leftarrow 0$

4. `PUSH` value: na vrh sklada naloži vrednost value.

   `SP--`, `MEM[SP]` $\leftarrow$ value

5. `NAME` label: na vrh sklada naloži naslov addr, ki ga določa oznaka label.

   `SP--`, `MEM[SP]` $\leftarrow$ addr

6. `REGN` label: na vrh sklada naloži vrednost registra r.

   `SP--`, `MEM[SP]` $\leftarrow$ r

7. `OPER` op: z vrha sklada vzame en ali dva operanda in na vrh sklada naloži rezultat operacije o.

   enomestna operacija:

   $\mathrm{oper}_1 \leftarrow$ `MEM[SP]`, `SP++`
   result $\leftarrow$ op oper_1
   `SP--`, `MEM[SP]` $\leftarrow$ result

   dvomestna operacija:

   $\mathrm{oper}_1 \leftarrow$ `MEM[SP]`, `SP++`
   $\mathrm{oper}_2 \leftarrow$ `MEM[SP]`, `SP++`
   result $\leftarrow \mathrm{oper}_1$ op oper_2
   `SP--`, `MEM[SP]` $\leftarrow$ result

8. `UJUMP`: z vrha sklada vzame novo vrednost programskega števca.

   addr $\leftarrow$ `MEM[SP]`, `SP++`
   `PC` $\leftarrow$ addr

9. `CJUMP`: z vrha sklada vzame vrednost in dva naslova ter v programski števec vpiše enega od naslovov glede na vrednost: če je ta ničelna, v programski števec vpiše prvega, sicer drugega.

   $\mathrm{addr}_1 \leftarrow$ `MEM[SP]`, `SP++`
   $\mathrm{addr}_2 \leftarrow$ `MEM[SP]`, `SP++`
   value $\leftarrow$ `MEM[SP]`, `SP++`
   value $= 0 \Longrightarrow$ `PC` $\leftarrow \mathrm{addr}_1$
   value $\neq 0 \Longrightarrow$ `PC` $\leftarrow \mathrm{addr}_2$

10. `CALL`: z vrha sklada vzame naslov, shrani klicni kazalec in programski števec, nastavi klicni kazalec in v programski števec vpiše naslov.

    addr $\leftarrow$ `MEM[SP]`, `SP++`
    `SP--`, `MEM[SP]` $\leftarrow$ `FP`
    `SP--`, `MEM[SP]` $\leftarrow$ `IP` $+ 4$
    `FP` $\leftarrow$ `SP` $+ 8$
    `IP` $\leftarrow$ addr

11. `RETN`: z vrha sklada vzame velikost parametrov in rezultat funkcije, restavrira klicni in skladovni kazalec, nastavi programski števec, umakne parametre z vrha sklada in na vrh sklada potisne rezultat.

    size $\leftarrow$ `MEM[SP]`, `SP++`
    result $\leftarrow$ `MEM[SP]`, `SP++`
    `SP` $\leftarrow$ `FP`
    `FP` $\leftarrow$ `MEM[SP]` $- 4$
    `PC` $\leftarrow$ `MEM[SP]` $- 8$

`SP` $\leftarrow$ `SP` $+$ size $+ 4$
`SP--`, `MEM[SP]` $\leftarrow$ result

12. `INIT`: z vrha sklada vzame naslov opisa začetne vrednosti in naslov spremenljivke ter inicializira spremenljivko z opisano začetno vrednostjo.

    init $\leftarrow$ `MEM[SP]`, `SP++`
    addr $\leftarrow$ `MEM[SP]`, `SP++`
    `MEM[`addr`...]` $\leftarrow$ *začetna vrednost*

    Začetna vrednost na naslovu je init opisana kot

    $$n\, n_1\, l_1\, d_{1,1} d_{1,2} \ldots d_{1,l_1}\; n_2\, l_2\, d_{2,1} d_{2,2} \ldots d_{2,l_2} \ldots n_n\, l_n\, d_{n,1} d_{n,2} \ldots d_{n,l_n},$$

    na naslovu addr pa se razvije v

    $$(d_{1,1} d_{1,2} \ldots d_{1,l_1})^{n_1} (d_{2,1} d_{2,2} \ldots d_{2,l_2})^{n_2} \ldots (d_{n,1} d_{n,2} \ldots d_{n,l_n})^{n_n}.$$

Psevdoukazi skladovnega stroja so naslednji:

1. `LABEL` label: oznaka label.
2. `SIZE` size: odmeri size bajtov pomnilnika.
3. `DATA` value: 32-bitna konstanta value v pomnilniku.

(V primeru dvomov poglejte v `Machine.java`.)