Interpreter programskega jezika JAIS

2. seminarska naloga pri predmetu Programiranje (II. stopnja)

V okviru seminarske naloge boste implementirali interpreter za jezik JAIS (Jezik Aritmetičnih Izrazov v Slovenščini). Kot osnutek bomo uporabili programsko kodo, ki smo jo začeli razvijati na predavanjih. Njo boste dopolnili tako, da bo vsebovala vse potrebne konstrukte programskega jezika, čigar sintaksa in semantika sta opisana na naslednjih straneh.

Navodila za izdelavo seminarja: Vaša seminarska naloga bo avtomatsko točkovana, zato je pomembno, da klic interpreterja pripravite natančno v skladu z navodili. Klic interpreterja naj ima sintakso **`(jais izraz okolje)`**, kjer izraz predstavlja izraz v jeziku JAIS, ki ga želimo interpretirati, okolje pa spremenljivko, ki hrani začetno okolje. Pri ročnih klicih interpreterja bo spremenljivka okolje ponavadi enaka praznemu seznamu (npr. `(jais (konst 5) null)`), tako da le-ta ne igra pomembne vloge uporabniku; potrebna in pomembna je za testni sistem, ki jo bo uporabljal pri preverjanju delovanja vašega interpreterja.

Spomnimo se, da je jezik JAIS definiran v jeziku Racket z uporabo konstrukta struct, ki nam omogoča definiranje lastnih podatkovnih tipov. Z uporabo teh konstruktov definirajte naslednje elemente jezika:

1.) Podatkovni tipi:

- **številčne konstante:** če je `n` numerična vrednost v jeziku Racket, je `(konst n)` konstanta v JAIS,
- **logični konstanti:** če je `b` logična konstanta v jeziku Racket (`#t, #f`), naj bo `(bool b)` logična konstanta v jeziku JAIS,
- **interval:** če sta `od` in `do` numerični vrednosti v Racketu, naj bo `(interval od do)` numerični interval s podano spodnjo mejo (od) in zgornjo mejo (do),
- **par:** če sta `e1` in `e2` izraza v jeziku JAIS, je `(zdruzi e1 e2)` par vrednosti `v1` in `v2`, ki ju dobimo z evalvacijo izrazov `e1` in `e2`,
- **ničelna vrednost:** `(nic)` je izraz v jeziku JAIS, ki predstavlja ničelno vrednost (analogno izrazu `null` v jeziku Racket). Pozor: `(nic)` je izraz, `nic` sam zase pa ni.

2.) Konstrukti za nadzor toka in za delo s podatkovnimi tipi:

- **vejanje:** `(ce-potem-sicer b e1 e2)` predstavlja vejanje v jeziku JAIS. Če je `b` (evalvira se v logično vrednost) resničen, je rezultat izraza `e1`, sicer `e2`. Semantika jezika definira, da se evalvira samo eden od `e1` in `e2`,
- **poizvedbe po tipih izrazov:** če `e` predstavlja izraz v JAIS, potem `(je-konst? e)`, `(je-bool? e)`, `(je-interval? e)` in `(je-nic? e)` predstavljajo logične konstante v JAIS, ki so resnične glede na to, katerega podatkovnega tipa je izraz `e`,
- **negacija:** če `e` predstavlja izraz v JAIS, potem `(negacija e)` predstavlja negacijo izraza `e`. Za logične vrednosti je rezultat negacije nasprotna logična vrednost; za številčne vrednosti je rezultat vrednost z nasprotnim predznakom; za intervale `(interval a b)` je rezultat `(interval -b -a)`,
- **seštevanje:** če sta `e1` in `e2` izraza v JAIS, je `(sestej e1 e2)` izraz, ki predstavlja seštevanje. Semantika tega izraza je opredeljena samo, če se `e1` in `e2` evalvirata v številčni konstanti ali intervala. Če se `e1` in `e2` evalvirata v številčni konstanti, je rezultat evalvacije celega izraza numerična vrednost, ki predstavlja vsoto. Če se `e1` in `e2` evalvirata v intervala, je rezultat evalvacije celega izraza interval, ki ima spodnjo mejo enako vsoti spodnjih mej podanih intervalov in zgornjo mejo enako vsoti zgornjih mej podanih intervalov. Če se eden izmed argumentov evalvira v številčno konstanto in drugi v interval, se številčna konstanta predstavi z intervalom (na primer, konstanta 5 postane interval od 5 do 5) in se izraz evalvira kot vsota dveh intervalov,
- **množenje:** analogno seštevanju, izraz `(zmnozi e1 e2)` predstavlja produkt numeričnih izrazov. Razlika je v tem, da je produkt intervalov `(interval a b)` in `(interval c d)` definiran kot interval `(interval e f)`, kjer je `e = min(a*c, a*d, b*c, b*d)` in `f = max(a*c, a*d, b*c, b*d)`,
- **dostop do elementov v intervalu ali paru:** če je `e` interval v jeziku JAIS, je `(glava e)` izraz v JAIS, ki predstavlja začetno točko intervala in `(rep e)` je izraz v JAIS, ki predstavlja končno točko intervala. Analogno, če je `e` par v jeziku JAIS, `(glava e)` in `(rep e)` predstavljata prvi in drugi element para. Semantika izrazov `(glava e)` in `(rep e)` za druge podatkovne tipe ni definirana,

- **primerjava številčnih vrednosti in intervalov:** `(vecje e1 e2)` je izraz v JAIS, ki predstavlja primerjanje izrazov `e1` in `e2`, ki sta bodisi oba hkrati številski konstanti bodisi intervala. Rezultat je `(bool #t)`, če je `e1` večji od drugega. Relacija "je večje" je za števila definirana kot običajna urejenost, za intervale pa glede na širino intervala (širši je *večji*, ne glede na začetno in končno točko intervala),
- **presek intervalov:** če sta `e1` in `e2` intervala v jeziku JAIS, je `(presek e1 e2)` interval, ki predstavlja njun presek.

3.) Spremenljivke

Implementirajte spremenljivke in okolje, v katerem se hranijo. Ob zagonu interpreterja naj bo okolje prazno, spreminja pa naj se skladno z izvajanjem interpreterja. Okolje naj bo predstavljeno s seznamom parov v jeziku Racket, ki naj bo oblike `(list (ime`$_1$ `. vrednost`$_1$`) ... (ime`$_n$ `. vrednost`$_n$`))`. Za definicijo spremenljivk in dostop do le-teh definirajte naslednja dva konstrukta:

- `(naj-bo s e1 e2)` je izraz v JAIS, ki trenutno okolje razširi z imenom spremenljivke `s`, ki ima vrednost `e1`, ter v razširjenem okoju evalvira izraz `e2`. Za definicijo večih spremenljivk lahko torej gnezdimo izraze `(naj-bo ...)`,
- `(sprem s)` je izraz v JAIS, ki vrne vrednost spremenljivke z imenom `s`, če je le-ta definirana. Upoštevajte, da se v okolju lahko nahaja več spremenljivk z istim imenom, ki senčijo ena druge. Izraz naj vrne pravilno vrednost spremenljivke, ki ni zasenčena.

4.) Funkcije

Implementirajte zapise funkcij in funkcijske klice. Pri implementaciji bomo ločili prave funkcije, ki uporabljajo leksikalni doseg, in t.i. skripte, ki uporabljajo dinamični doseg. Za te namene definirajte konstrukta v jeziku JAIS:

- **`(funkcija ime farg telo)`** in **`(skripta ime telo)`**, kjer je `ime` niz (jezika Racket), ki predstavlja ime funkcije, `farg` je seznam (v jeziku Racket), ki predstavlja seznam formalnih argumentov, `telo` pa je izraz v jeziku JAIS (telo funkcije). Namesto imena funkcije/skripte lahko nastopa tudi logična vrednost `#f` - v tem primeru zapisan izraz predstavlja anonimno funkcijo. Skripta nima argumentov, ima zgolj ime in telo.

Napotki za implementacijo leksikalnega dosega za konstrukt `(funkcija ...)`: Ob interpretiranju programa naj se konstrukt `(funkcija ...)` evalvira v konstukt `(ovojnica env f)`, kjer je `f` funkcijski konstrukt, `env` pa okolje na mestu, kjer je bila funkcija definirana (leksikalno pravilo). Bodite pozorni na to, da ovojnice niso del izvorne kode programa v jeziku JAIS, temveč le interni mehanizem interpreterja, ki funkcijam omogoča rabo okolja, v katerem so te bile definirane. Funkcija je torej vrednost, ki jo interpreter evalvira v ovojnico.

Funkcijski klici. Implementirajte konstrukt `(klici e arg)`, ki predstavlja funkcijski klic. Izraz `e` se mora ovrednotiti bodisi v funkcijsko ovojnico `(ovojnica ...)` bodisi v skripto `(skripta ...)`; `arg` predstavlja seznam argumentov funkcije, ki je podan kar kot seznam v jeziku Racket, ki vsebuje izraze, ki se evalvirajo v izraze jezika JAIS. Ločimo dve vrsti funkcijskih klicev:

- klici funkcijskih ovojnic: telo funkcije, ki je vsebovana v ovojnici, evalviramo v okolju, ki je vsebovano v ovojnici, ki ga razširimo z: (1) imeni in vrednosti argumentov (imena so podana v funkciji, ki je vsebovana v ovojnici - `farg`; njihove vrednosti pa so v klicu ovojnice - `arg`) in (2) imenom funkcije, ki ga povežemo z ovojnico (potrebno za rekurzivne klice),
- klici skriptne kode: telo skripte se izvede v lokalnem okolju (okolju, kjer je skripta klicana - uporablja se torej dinamični doseg). Ker skripta ne uporablja argumentov, je vrednost parametra `arg` pri klicu skripte nepomembna, zato lahko uporabimo `(klici (skripta ...) (nic))`.

Funkcijsko ovojnico **optimizirajte** tako, da iz nje izločite nepotrebne spremenljivke: (1) senčenja zunanjih spremenljivk, (2) senčenja spremenljivk z lokalnimi argumenti in (3) spremenljivke, ki v funkciji niso potrebne.

5.) Makro sistem

V jeziku Racket lahko implementiramo funkcije, ki delujejo kot makri v jeziku JAIS, ter na ta način olepšajo sintakso JAIS in razširijo jezik z novimi konstrukti. Definirajmo naslednje makre:

- `(odstej e1 e2)`, ki generira izraz v JAIS, ki predstavlja razliko `e1-e2`,
- `(manjsi e1 e2)`, ki preveri ali velja urejenost med elementoma,
- `(enak e1 e2)`, ki preveri, ali sta elementa enaka,
- `(vsebuje i1 i2)`, ki preveri, ali interval `i1` v celoti vsebuje interval `i2`.

Dodatna naloga: Sami implementirajte svoj makro sistem, ki je higieničen (ne izvaja naivne makro razširitve, ampak ločuje pomen spremenljivk v makrih od spremenljivk v programski kodi). Način implementacije tovrstnega sistema je v celoti prepuščen vam, uporabite lahko tudi drugačne mehanizme kot so funkcije v Racketu, ki simulirajo makre. Ta naloga ne bo avtomatsko točkovana, temveč jo boste, v primeru implementacije, razložili in zagovarjali na zagovoru pri asistentu.

ODDAJA IN VREDNOTENJE SEMINARSKE NALOGE:

Način ocenjevanja: Ocenjevanje oddanih nalog bo izvedeno avtomatsko (v dveh krogih ocenjevanja), vsebino kode pa bo potrebno razložiti tudi na zagovoru naloge. Avtomatsko testiranje bo izvedeno z množico testnih programov, ki ne bodo podani vnaprej. Delovanje programa lahko med razvojem preizkušate s testnimi primeri, ki so podani, in s svojimi lastnimi testnimi primeri.

Predrok in rok za oddajo: Kočni rok za oddajo seminarske naloge je **sobota, 17. 1. 2015**. Študenti, ki boste oddali svoje naloge do vključno **sredo, 14. 1. 2015**, sodelujete v predroku za oddajo – prejeli boste rezultate testiranja vašega programa na testnih primerih ter imeli možnost popraviti in ponovno oddati program do končnega roka za oddajo. S točkami ocenjevanja končne verzije boste seznanjeni na zagovoru, katerega potek bo vplival na končno oceno seminarske naloge.

Točkovanje: Seminarska naloga bo vrednotena po podanem točkovalniku:

Naloga	**Točke**
1. Implementacija podatkovnih tipov	10
2. Konstrukti za kontrolo toka	15
3. Spremenljivke	15
4. Funkcije	
skripte	10
ovojnice	15
delovanje rekurzije	10
optimizacija ovojnic	10
5. Makro sistem	
zahtevani makri	10
dodatno: higieničen makro sistem	+10
Sintaksa (izraznost, raba nevidnih znakov, komentarji)	5
	100 (+10)