Vt 08

UPPGIFT i INTERPRETATORTEKNIK

Kort formulering: Gör en interpretator för nedanstående språk. Vi gör dock vissa restriktioner.

Längre formulering: För nedan definierade språk skall ni konstruera en interpretator . Ni får använda valfri metod.Syntaxen definieras nedan med hjälp av formella metoder, medan semantiken endast beskrivs informellt. Med de kunskaper ni har om programspråk och med de valda namnen på metasymbolerna bör ni kunna klara av att ge en rimlig semantisk tolkning av program i språket. Nedanståaende grammatik specificerar ett fullständigt språk, men vi ska göra en del restriktioner enligt nedan.

Uppgiften skall normalt göras i grupper om två. Vissa hjälpmedel finns skrivna.

Vad ni ska göra.

1. Klara av ett block med deklarationer och tillordningssatser med aritmetiska uttryck, utan funktionsanrop.
2. Utöver enligt 1 också användning av inre block, dvs. en sats kan vara ett block.
3. Ni ska nu utöka med att också klara funktionsanrop.
4. Ni ska också klara villkorssatser.
5. Av standardfunktiner ska ni minst klara utskrifter, dvs. write.

Allmänt.

De typer som finns i språket är heltal, flyttal och texter. Ni är förlåtna om ni inte klarar flyttal, dvs. typen real.
Ni behöver naturligtvis klara av att skriva ut saker, i början är det lämpligt med automatiska utskrifter av debugskäl.
De saker i språket som inte finns med i punkterna ovan behöver ni inte implementera, detta är t.ex. whilesatser, repetionssatser och objekt och allt som markerats som extra i grammatiken. Vidare behöver ni inte klara array.
Ni bör göra en lösning som inte helt försvårar eller omöjliggör en utvidgning till det fulla språket.

Redovisning: Redovisning sker till mig framför datorn, samt viss skriftlig dokumentation.

Syntax

program --> block
block --> **begin** deklista satser **end**
deklista --> dekl **;** deklista
--> dekl **;**
dekl --> typ idlist
--> typ **array** arrlist
--> procedur
--> typ procedur
--> klass **¤Extra¤**
idlist --> id , idlist

--> id

arrlist --> arr , arrlist
--> arr
procedur --> **function** id par **;** speclista kropp
klass --> **class** id par **;** speclista kropp ¤**Extra obj**¤
typ --> **integer**
--> **real**
--> **text**
--> **ref** (id) ¤**Extra obj**¤
arr --> id (uttr **:** uttr) ¤ **Begränsning** id(fix : fix) ¤
par --> (plista)
--> ε
plista --> id
--> id **,** plista
speclista --> spec **;**
--> spec **;** speclista
--> ε
spec --> typ id
--> typ **array** id
kropp --> block
--> sats
--> ε
satser --> sats
--> sats **;** satser
sats --> block
--> **begin** satser **end**
--> anrop
--> medansats
--> tillordning
--> villkorssats
--> repetitionssats ¤**Extra**¤
--> ε
medansats --> **while** villkor **do** sats
tillordning --> id **:=** uttr
--> id (index) **:=** uttr
villkorssats --> **if** villkor **then** sats **else** sats
repetitionssats --> **for** id **:=** uttr **step** uttr **until** uttr **do** sats ¤**Extra**¤
anrop --> id
--> id (alist)
--> id **.** id (alist) ¤**Extra**¤
alist --> uttr
--> uttr **,** alist
index --> uttr
villkor --> uttr relop uttr
relop --> = | **<** | **>** | **<>** | **<=** | **>=**
uttr --> uttr addop term
--> term
--> sträng
term --> term mulop faktor
--> faktor

faktor --> (uttr)
--> id
--> tal
--> id (index)
--> **new** id par ¤**Extra obj**¤
--> uttr **.** uttr ¤**Extra obj**¤
--> anrop
addop --> **+** | **-**
mulop --> * | **/**

Lexikaliska enheter

id --> bok { bok | siffra } *
tal --> fix
--> flyt
flyt --> fix **.** fix
--> fix **.**
fix --> siffra { siffra } *
bok --> **A | B | a | b**
siffra --> **0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**
sträng --> **"** nästan vad som helst **"**

Vad gäller syntaxen så läs igenom grammatiken ordentligt. Man kan skilja på två huvudriktningar i syntax, vad vi kan kalla "Algol-syntax" och "C-syntax". Pascal hör till den första och java till den andra, vårt språk ovan hör till den första. Lägg särskilt märke till att i java avslutas satser med semikolon, i vårt språk skiljs satser åt av semikolon. Detta märks tydligt i villkorssatser, som ges som exempel nedan.
java: if (a<b) a=5; else a=a+b;
vårt: if a<b then a:=5 else a:=a+b
OBS! i vårt språk inget ; före else (som det skall vara i java)

Semantik
Detta är en ofullständig beskrivning, ni får använda er fantasi också. *uttr* Aritmetiska uttryck har sin normala betydelse, vad gäller prioritet mm. Vid lika prioritet skall vänster till höger gälla. Blandad aritmetik är tillåten, reultatet blir heltal endast om alla ingående operander är heltal, dock division ger alltid flyttal som resultat. Strängar används endast som parametrar och för utskrift, samt i direkta tillordningar av typen T:="detta är en sträng"; <tillordning> är också normal, typkonvertering mellan olika taltyper ska ske vid behov. <anrop> Detta är ett normalt funktionsanrop, aktuella parametrar skall stämma överens med specifikationen av de formella, dock får aritmetisk typkonvertering ske. Procedurer tillåts ej som parametrar. Övriga satser är som sina motvarigheter i java och liknande språk. <procedur> som har en <typ> skall också få ett värde, antingen i en tillordningssats eller med "default"(=0). Observera att rekursiva procedurer är tillåtna. Tänk också på att det är tillåtet med procedurer och funktioner utan parametrar, om F är en sådan så anropas den just med F utan några parenteser. Observera att om man har heltalsproceduren P med en parameter som skall vara ett heltal så är anropet P(P(1)) tillåtet, detta är inte någon procedur som parameter, utan man sänder ett värde beräknat med ett proce-

duranrop som parameter!!!

Standardprocedurer.
Följande procedurer skall finnas: write(uttr), som medför att uttryckets värde skrivs ut. Observera att man skall kunna skriva ut heltal, flyttal och strängar. Readint som är en funktionsprocedur som ger ett heltal, readreal som läser ett flyttal och readtext som läser en text. Ni ska också ha en möjlighet att styra utskrift till ny rad med writln. Ni bör hantera standardprocedurer och standardfunktioner på ett sätt som gör att man lätt skulle kunna lägga till fler. Ett normalt fullskalespråk har oftast ett hundratal sådana!

Hjälpmedel.
Det finns ett program , p1 som gör en lexikalisk analys samt en block och deklarationsanalys. p1 prog , läser en fil prog.prg och producerar en fil prog.p1 med ett format som redan beskrivits i ett särskilt dokument.
Ni kommer att få ytterligare tips och exempel.