***Prevođenje programskih jezika – teorija***

1. Navesti i objasniti korake anlize izvornog programa

* Leksička analiza - provjerava jesu li leksičke jedinke u izvornom programu pravilno napisane. Gradi se tablica znakova.
* Sintaksana analiza - provjerava da li niz leksički jedinki zadovoljava sintaksnim pravilima zadanu hijerarhijsku strukturu izvornog programa.
* Semantička analiza – provjerava deklaraciju varijabli, tipove podataka u izrazima, indeksiranje polja, parametre potprograma te tijek izvođenje programa.

1. Objasni kako leksički analizator utvrđuje leksičku pogrešku i navesti najmanje dva postupka oporavka od pogreške

* Ako nijedan prefiks niza znakova izvornog programa nije moguće grupirati ni u jednu od leksičkih jedinki, onda leksički analizator prijavi pogrešku.

Postupci oporavka od pogreške: odbacivanje uzastopnih znakova, odbacivanje ne nužno uzastopnih znakova, dodavanje znakova u niz, zamjena znakova u nizu novim znakovima i zamjena mjesta znakova u nizu.

1. Navesti namjenu programa Lex i opisati strukturu ulazne datoteke Lexa

* Lex je najpoznatiji program koji omogućuje automatiziranje izgradnje leksičkog analizatora.

%{

Deklaracije

}%

Regularne definicije

%%

Pravila prevođenja

%%

Pomoćne procedure

1. Navesti i kratko opisati postupke pretvorbe produkcija u produkcije LL(1) gramatike.

* Lijevo izlučivanje –
* Zamjena nezavršnih znakova –
* Lijeva rekurzija –

1. Objasniti parsiranje od dna prema vrhu metodom prednosti operatora, relaciju prednosi, akcije parsera i određivanje uzorka za zamjenu

* Parser uspoređuje završni znak na vrhu stoga i znak u ulaznom nizu, te na temelju relacije prednosti donosi odluku o primjeni akcije *Pomakni* ili *Reduciraj*.

|  |  |
| --- | --- |
| Relacija | Značenje |
| a <= b | Prednost a je manja od prednosti b |
| a⬄b | a i b su jednake prednosti |
| a=>b | Prednost a je veća od prednosti b |

Akcija *Pomakni* – ako je prednost završnog znaka na vrhu stoga jednaka ili manja od prednosti znaka u ulaznom nizu, onda parser primjenjuje akciju *Pomakni*.

Akcija *Reduciraj* – ako je prednost završnog znaka na vrhu stoga veća od prednosti znaka na ulaznom nizu, onda parser primjenjuje akciju *Reduciraj*.

*Određivanje uzorka za zamjenu* – parser uzima jedan po jedan znak sa stoga sve do završnog znaka koji ima manju prednost od prednosti predhodno uzetog znaka.

1. Objasniti postupke nadziranja i oporavka od pogrešaka tijekom LR parsiranja

* Postupak traženja sinkronizacijskog znaka prekida sintaksnu analizu programske cjeline u kojoj je pronađena pogreška.
* U postupku lokalnih promjena svi elementi tablice LR parsera koji označavaju akciju Odbaci() zasebno se analiziraju.Na temelju pravila izvornog jezika, pretpostavi se najvjerojatnija pogreška korisnika.Na temelju pretpostavljene pogreške odrede se akcije promjene sadržaja stoga i ulaznog niza.Različiti elementi tablice označavaju se različitim akcijama Odbaci(k), koje imaju zasebno definirane akcije prepravka sadržaja stoga i ulaznog niza.

1. Opisati postupak izgradnje potisnog automata za prijevodnu gramatiku

* (184)

1. Objasniti što je provjera vrijednosti obilježja i opisati pojedine postupke za provjeru vrijednosti obilježja

* Provjera vrijednosti obilježja koristi atributnu prijevodnu gramatiku. Ako se tijekom provjere vrijednosti pojedinih obilježja ustanovi pogreška, onda semantički analizator ispisuje poruku o pogrešci, pridruži obilježju vrijednost *Pogreška*, pokrene postupak oporavka od pogreške i nastavi analizu izvornog programa.

a)Provjera vrijednosti obilježja naredbi deklaracija

b)Provjera vrijednosti obilježja izraza

c) Provjera vrijednosti obilježja ulančanih naredbi

1. Navesti kratko objasniti postupke za određivanje djelokruga deklaracije nelokalnih imena

* *Statičko pravilo djelokruga bez ugniježđenih procedura*

Ako ime nije deklarirano naredbama procedure onda je važeća deklaracija zadana naredbama glavnog programa.

* *Statičko pravilo djelokruga ugniježđenih procedura*

Zasniva se na pravilu najbliže ugnježđujuće procedure:

1. Djelokrug deklaracije koja je zadana naredbom procedure uključuje sve naredbe te procedure.
2. Ako ime x nije deklarirano naredbama procedure q(), onda je važeća deklaracija zadana naredbama ugnježđujuće procedure p() za koju vrijedi:
3. Ime x deklarirano je naredbama procedure p()
4. Ne postoji ni jedna druga procedura r() za koju vrijedi: ime x je deklarirano naredbama procedure r(), procedura p() ugnježđuje proceduru r() i procedura r() ugnježđuje proceduru q().

* *Dinamičko pravilo djelokruga*

Određuje se na temelju izvođenja programa. Ako naredbe pozvane procedure ne deklariraju ime identifikatora, onda su važeće deklaracije pozivajuće procedure. Postupak određivanja važeće deklaracije nastavlja pretraživanje ostalih pozivajućih procedura.

1. Navesti osnovne razine međukoda i objasniti namjenu svake razine

* Međukod više razine – počinje sinteza; Sačuvane strukture petlji i indeksa polja izvornog programa; Naredbe pridruživanja bezuvjetnog i uvjetnog grananja, naredbe petlji i upravljanja procedurama
* Međukod srednje razine – npr. skup naredbi upravljanja tijekom izvođenja programa čine jednostavne naredbe uvjetnog i bezuvjetnog skoka, naredbe poziva i povratka iz potprograma; koristi simbolička imena identifikatora izvornog programa, simbolička imena privremenih varijabli i registara;
* Međukod niže razine – jedna naredba se prevodi u jednu strojnu naredbu; ako je naredbu moguće na više načina prevesti, optimiranje bira najpovoljniji; koristi se manji skup različitih načina adresiranja od strojnog jezika; vrijednosti identifikatora se dohvaćaju primjenom registara ili adresa mem. Ćelija; koriste se simbolička imena registara i broj im je neograničen; pogodan za strojno zavisno optimiranje.

1. Navesti elemente strukture generatora ciljnog programa

* Generator ciljnog programa definira se ulazom i izlazom, a čine ga postupci izrade adrese, izbora naredbe, izbora redoslijeda izvođenja naredbi i dodjele registara postupcima.

1. Objasni kako se dobiva i boji graf zavisnosti simboličkih i stvarnih registara te kako se dodjeljuju stvarni registri
2. Objasniti korake jezičnog procesora

* Osnovni koraci u izgradnji jezičnog procesora su definicija, strukturiranje, programsko ostvarenje, ocjena i održavanje jezičnog procesora.

1. Objasniti podjelu jezičnih procesora s obzirom na broj prolaska kroz izvorni program
2. Navesti zahtjeve koje mora ispuniti detekcija pogrešaka u sintaksnom analizatoru

* Zahtijeva se da postupak određivanja mjesta i opisa porešaka precizno odredi mjesto pogreške, kratko i jasno opiše pogrešku, te da značajno ne uspori rad sintaksnog analizatora. Zahtijeva se kratak i jasan opis pogreške. U poruci se označi mjesto u izvornom programu gdje je pronađena pogreška.

1. Objasniti postupak uklanjanja lijeve rekurzije tijekom pretvorbe produkcija u LL(1) oblik.
2. Objasniti razlike u ostvarenju LR(0), SLR(1), LALR() i LR(1) parsera te navesti njihove prednosti i nedostatke.
3. Nabrojati i objasniti formalne metode semantičkog analizatora. (provjeriti je li to isto što i formalni modeli)
4. Objasniti kako se obrađuju svojstva izvorišta koja nemaju dostupne vrijednosti.
5. Definirati L-atributnu prijevodnu gramatiku.

Atributna prijevodna gramatika je L-atributna ako i samo ako vrijedi:

1. Vrijednost nasljednog svojstva znaka desne strane produkcije računa se na temelju vrijednosti nasljednih svojstava nezavršnog znaka lijeve strane produkcije i na temelju vrijednosti svojstava znakova desne strane produkcije koji su lijevo od zadanog znaka.
2. Vrijednost izvedenog svojstva nezavršnog znaka lijeve strane produkcije računa se na temelju vrijednosti nasljednih svojstava nezavršnog znaka lijeve strane produkcije i na temelju vrijednosti svojstava znakova desne strane produkcije.
3. Vrijednost izvedenog svojstva izlaznog završnog znaka računa se na temelju nasljednih svojstava tog istog izlaznog završnog znaka.
4. Objasniti povezivanje imena izvornog programa i objekata ciljnog programa te relaciju okoline i relaciju stanja.

Način povezivanja imena izvornog programa,podatkovnih objekata ciljnog programa i vrijednost podatkovnih objekata određen je vrijednostima relacija:

Relacija okoline – pridružuje imenima izvornog programa podatkovne objekte ciljnog programa.

Relacija stanja – pridružuje podatkovnom objektu vrijednost.

Isto ime izvornog programa je moguće deklarirati više puta naredbama različitih procedura. Naredbe deklaracija različitih procedura pridružuju istom imenu različite podatkovne objekte ciljnog programa.

1. Opisati mehanizam povratne razmjene vrijednosti parametara procedura te navesti način ostvarenja.

U opisnik pozvane procedure zapisuju se vrijednosti aktualnih parametara i njihove adrese:

1. Formalni parametri su lokalni podaci pozvane procedure. U opisniku pozvane procedure ostavljaju se prazna mjesta za zapis vrijednosti i adresa aktualnih parametra.
2. Pozivajuća procedura odredi vrijednosti i adrese aktualnih parametara i zapiše ih u opisnik pozvane procedure. Pozvana procedura koristi i mijenja isključivo lokalne vrijednosti spremljene u svom opisniku. Nakon njenog izvođenja pozvana procedura pročita iz svog opisnika vrijednosti aktualnih parametara i njihove adrese. Vrijednosti aktualnih parametara se spreme u memoriju primjenom pročitanih adresa.
3. Opisati postupak optimiranja petlji kod međukoda niže razine i ciljnog programa.

Postupci optimiranja međukoda niže razine dijele se na jednostsavne koji koriste analizu izvođenja programa. Oni se izvode tijekom generiranja ciljnog programa ili tijekom pripreme ciljnog programa za izvođenje. Preuređivanjem naredbi optimira se uporaba raznih sredstava procesora, na primjer registra, aritmetičko-logičkih jedinki, cjevovoda, priručne memorije,itd. Učinkovita uporaba sredstava procesora skraćuje vrijeme izvođenja ciljnog programa i smanjuje njegovu veličinu.

1. Opisati postupak izrade adresa naredbama.

Adrese naredbi se generiraju primjenom brojača → veličina brojača povećava se za veličinu generirane naredbe izražene u oktetima. Koriste se dvije klase:

Lista unazadnih adresa – čine je zapisi simboličkih programskih oznaka i memorijskih adresa naredbi kojima su te oznake dodijeljene.

Lista unaprijednih adresa – čine je zapisi programskih oznaka i kazaljke na generirane naredbe što koriste te oznake.

1. Objasniti što je funkcija preslikavanja jezičnog procesora i navesti njene vrste.
2. Opisati program simulator leksičkog analizatora zasnovan na tablici prijelaza DKA.
3. Objasniti COBOL sustav oznaka.

Za opis jezika COBOL koristi se slijedeći sustav oznaka:

Mala slova → varijabla

Velika slova → konstanta koju je moguće izostaviti

Podcrtana velika slova → konstanta koju nije moguće izostaviti

Znak do znaka → nadovezivanje

[ ] → neobavezni izbor jedne od mogućnosti zadane u zagradama [ ]

{ } → obavezni izbor jedne od mogućnosti zadane u zagradama { }

... → ponavljanje prethodne sintaksne cjeline

1. Opisati postupak računanja skupova *SLIJEDI* za prazne nezavršne znakove.

Na temelju tablice *Ispred* moguće je izravno odrediti skupove *SLIJEDI* za sve prazne nezavršne znakove. Zadana gramatika ima dva prazna nezavršna znaka <B> i <D>. Redak označen znakom <B> određuje sljedeće elemente skupa *SLIJEDI* (<B>)={a, c, d, e, f, ┴}. Redak označen znakom <D> određuje sljedeće elemente skupa *SLIJEDI*(<D>)={a, b, c, e, f, ┴}.

1. Objasniti razliku između strojno nezavisnog i strojno zavisnog optimiranja.

Strojno nezavisno optimiranje optimira višu razinu koda, dok strojno zavisno optimiranje optimira nižu razinu koda.

1. Objasniti konstrukciju epsilon-NKA u postupku izgradnje SLR parsera.
2. Objasniti kako se obrađuju izvedena svojstva izlaznih završnih znakova koji se ne stavljaju na stog.
3. Navesti i objasniti algoritam ispitivanja jednakosti obilježja konstantnih vrijednosti.
4. Opisati algoritam gradnje lanca kazaljki nelokalnih imena i vektora dubine gniježđenja kod statičkog pravila djelokruga ugniježđenih procedura.

Način određivanja vrijednosti kazaljke nelokalnih imena pozvane procedure i gradnja vektora ovise o dubini gniježđenja pozivajuće i pozvane procedure. Algoritam zasebno obrađuje dva različita slučaja: pozvana procedura deklarairana je naredbama pozivajuće procedure i pozvana procedura deklarirana je naredbama procedure koja ugnježđuje pozvanu i pozivajuću proceduru. Budući da se za imena procedura koriste ista pravila pristupa nelokalnim imenima kao i za sve ostale identifikatore, u prvom slučaju dubina gnježđenja pozivane procedure *i* za jedan je veća od dubine gniježđenja pozivajuće procedure (i=j+1) , a u drugom slučaju dubina gniježđenja pozvane procedure *i* jednaka je ili manja od dubine gniježđenja pozivajuće procedure (i≤j).

1. Navesti i kratko opisati linearne oblike međukoda.

Međukod linearnog oblika čine troadresne naredbe: X:= Y op Z, gdje su Y i Z adrese operanada, X adresa rezultata i op iznaka operacije.Budući da koriste tri adrese, naredbe se nazivaju troadresnim. Troadresne naredbe linearnog su oblika, jer je izvršeno izravnavanje sintaksnog stabla. Operandi su korisničke varijable, privremene varijable i konstante koje se programski ostvaruju primjenom kazaljki koje pokazuju na tablicu znakova. Troadresne naredbe upravljaju tijekom izvođenja programa koristeći simboličke programske oznake.

1. Nabrojati komponente koje čine analizu izvođenja programa.

Faza analize rastavlja izvorni program u sastavne dijelove , provjerava pravila jezika, prijavljuje pogreške i zapisuje izvorni program primjenom različitih struktura podataka u memoriju računala. Analiza izvornog programa izvodi se u više zasebnih koraka: leksička analiza, sinstaksna analiza i semantička analiza.

1. Objasniti generiranje ciljnog programa na temelju postfiksnog sustava oznaka.
2. Navesti koja pravila se susreću kod programskih jezika te ih objasniti.

Pravila programskih jezika: približiti jezik programiranja području primjene, osloboditi jezik programiranja ovisnosti o arhitekturi računala, omogućiti što lakše učenje pravila jezika, olakšati razvoj i razumijevanje programa, olakšati ispravljanje i pronalaženje pogrešaka u programu, olakšati održavanje i dokumentiranje programa, povećati prenosivost na računala različitih arhitektura i skratiti vrijeme rješavanja problema. (valjda su mislili na ovo)

1. Objasniti samoprevodilac.
2. Nabrojati i objasniti osnovne klase leksičkih jedinki.

Ključne riječi (npr. ako, onda, inače)

Operatori (pridruživanja, zbrajanja, oduzimanja, množenja i dijeljenja)

Specijalni znakovi (npr. zagrade, zarez i točka)

Identifikatori (npr. imena varijabli, polja, programa i potprograma)

Konstante (npr. cjelobrojne, s posmačnim zarezom i znakovne konstante)

1. Definirati S-gramatiku i kratko opisati konstrukciju potisnog automata za S-gramatiku.

Kontekstno neovisna gramatika je S-gramatika a ko i samo ako su zadovoljena sljedeća svojstva:

1. Desna strana bilo koje produkcije započinje završnim znakom gramatike.
2. Desna strana nijedne produkcije nije prazan niz ε.
3. Ako više produkcija ima isti nezavršni znak na lijevoj strani, onda desne strane tih produkcija započinju različitim završnim znakovima.

Izgradnja potisnog automata:

1. Potisni automat ima samo jedno stanje koje je ujedno i početno stanje, Q={q0}
2. Skup ulaznih znakova potisnog automata jednak je skupu završnih znakova gramatike uvećan oznakom kraja niza ┴, ∑=T U {┴}.
3. Skup znakova stoga potisnog automata čine oznaka dna stoga , skup nezavršnih znakova gramatike V i skup svih onih završnih znakova gramatike T' koji su na desnim stranama produkcija, ali ne isključivo na krajnje lijevim mjestima
4. Na početku rada na stogu je oznaka dna stoga i početni nezavršni znak S.
5. Budući da potisni automat ima samo jedno stanje, funkcija prijelaza definira se dvodimenzionalnom tablicom. Reci tablice označavaju znakove stoga,a stupci ulazne znakove. Elementi tablice označavaju akciju potisnog automata na ulaznom nizu i akciju na stogu.
6. Opisati postupak oporavka od pogrešaka u sintaksnom analizatoru.

Oporavak od pogrešaka:

* Uvođenje dodatnih produkcija
* Postupak lokalnih promjena
* Traženje sinkronizacijskog znaka

1. Definirati LL(1) gramatiku i kratko opisati konstrukciju potisnog automata za LL(1) gramatiku.
2. Objasniti akcije parsera od dna prema vrhu koji koristi tehniku pomakni-pronađi. Opisati proturječja koja se pojavljuju.

Ako je na vrhu stoga A i ul. znak a, te postoji:

1. Relacija ISPODZNAKA(A,a), akcija je POMAKNI (a)
2. Relacija REDUCIRANZNAKOM(A,a), akcija je REDUCIRAJ – parser traži uzorak za zamjenu na stogu i zamijeni ga s nezavršnim znakom na lijevoj strani nađene produkcije.

Ako je ulazni znak a s znakom stoga A istodobno u obje relacije: PROTURJEČJE Pomakni/Reduciraj!

1. Objasniti sintaksom vođenu semantičku analizu.

Semantičke akcije – potprogrami za analizu sintaksnih cjelina. Ugrađuju se u produkcije gramatike kao izlazni završni znakovi.

Izlazni završni znakovi – akcijski znakovi. Generiraju izlaz, ne upravljaju postupkom parsiranja, ne definiraju produkcije, pokreću semantičke akcije.

Ulazni završni znakovi – upravljaju parsiranjem, dio ulaznog niza.

1. Definirati atributnu prijevodnu gramatiku.

Atributna prijevodna gramatika je gramatika proširena na slijedeći način:

1. Znakovima gramatike dodjeljuje se konačni skup svojstava (izvedenih ili nasljednih), kojima se pridružuju vrijednosti
2. Računanje vrijednosti nasljednih svojstava
3. Vrijednost nasljednog svojstva znaka desne strane produkcije računa se na temelju vrijednosti ostalih znakova produkcije.
4. Početna vrijednost nasljednog svojstva se zadaje s produkcijama gramatike
5. Računanje vrijednosti izvedenih svojstava
6. Vrijednost izvedenog svojstva nezavršnog znaka lijeve strane produkcije računa se na temelju vrijednosti ostalih znakova produkcije.
7. Vrijednost izvedenog svojstva izlaznog završnog znaka računa se na temelju ostalih svojstava dodijeljenih koja su dodijeljena tom istom znaku.
8. Opisati algoritam provjere jednakosti tipova obilježja temeljen na provjeri jednakosti strukture obilježja.
9. Objasniti načine ostvarenja dinamičkog pravila djelokruga.

Pretraživanje po dubini (prateći kazaljke..)

Pretraživanje statičke memorije (za sva lokalna imena u statičkoj memoriji)

1. Opisati graf zavisnosti programa i navesti sve zavisnosti koje se njime opisuju.

Graf zavisnosti je nelinearni oblik međukoda posebno prilagođen postupcima optimiranja. To je usmjereni graf čije čvorove čine naredbe međukoda. Grane između čvorova predstavljaju različite zavisnosti među naredbama. Graf se sastoji od dva podgrafa, graf zavisnosti upravljačkog tijeka i graf zavisnosti podataka.

Četiri su vrste zavisnosti podataka: unaprijedna zavisnost, unazadna zavisnost, zavisnost odredišta i zavisnost izvorišta.

1. Navesti faze rada od kojih se sastoje faza analize i faza sinteze jezičnog procesora.

Faza analize: leksička analiza, sinstaksna analiza i semantička analiza.

Faza sinteze: generiranje međukoda, strojno nezavisno optimiranje, generiranje stojnog programa, strojno zavisno optimiranje i priprema strojnog programa za izvođenje.

1. Objasniti postupak razrješenja nejednoznačnosti pretraživanjem desnog konteksta.

Pretraživanje desnog konteksta zadaje se na slijedeći način: r/r'

Gdje su r i r' regularni izrazi. Niz u ulaznom spremniku grupira se u leksičku jedinku definiranu regularnim izrazom r ako i samo ako iza niza definiranog tim regularnim izrazom u ulaznom spremniku slijedi niz koji koji je definiran regularnim izrazom r'.

1. Navesti i kratko opisati podatkovnu strukturu sintaksnog analizatora.

Podatkovnu strukturu sintaksnog analizatora čine globalni i lokalni podaci. Osim sintaksnog analizatora, globalne podatke koriste i ostali koraci rada jezičnog procesora. Globalnu strukturu podataka sačinjavaju tablica znakova i stog. Tablica uniformnih znakova jest osnovna i najznačajnija podatkovna struktura sintaksnog analizatora. Tablica ključnih riječi, operatora i specijalnih znakova je druga po važnosti. Sintaksni analizator rijetko koristi tablicu identifikatora. Većina sintaksnih i semantičkih analizatora razmjenjuje podatke primjenom stoga. Lokalna struktura podataka sintaksnog analizatora gradi se za potrebe procesa parsiranja.

1. Navesti četiri jezika koji su vezani uz definiciju jezičnog procesora.
2. Navesti kako se izvodi nadziranje i oporavak od pogrešaka kod LR parsiranja.
3. Postupak traženja sinkronizacijskog znaka – prekida sintaksnu analizu programske cjeline u kojoj je pronađena pogreška.
4. Postupak lokalnih promjena – svi elementi tablice LR parsera koji označavaju akciju *Odbaci()* zasebno se analiziraju. Na temelju pravila izvornog jezika, pretpostavi se najvjerojatnija pogreška korisnika. Na temelju pretpostavljene pogreške odrede se akcije promjene sadržaja stoga i ulaznog niza. Različiti elementi tablice označavaju se različiti akcijama *Odbaci(k)*, koje imaju zasebno definirane akcije prepravka sadržaja stoga i ulaznog niza.
5. Objasniti parsiranje od dna prema vrhu metodom Pomakni-Reduciraj.
6. Navesti i objasniti tri najčešće primjenjivana formalna modela semantičkog analizatora.
7. Prevođenje jezika – u jezik koji ima definiranu semantiku
8. Simulacija na apstraktnom stroju – zasniva se na skupu pravila primjenom kojih se simulira izvođenje izvornog programa. Apstraktno računalo definira se stanjima.
9. Skup aksioma – logičke tvrdnje koje korisnik zadaje u različitim dijelovima izvornog programa, a njma izriče očekivani rezultat izvođenja tog dijela izvornog programa.
10. Opisati algoritam za izračunavanje IS\_FOLLOWED\_BY relacije.
11. Definirati L-atributnu prijevodnu gramatiku.
12. Opisati opisnik procedure.
13. Opisati algoritam koji ispituje jednakost strukture tipova obilježja koji ne sadrži varijable.
14. Opisati postupke pretraživanja po dubini i pretraživanja statičke memorije kod dinamičkog pravila djelokruga.
15. Navesti prednosti i nedostatke uporabe jezičnih procesora.(str. 1.)
16. Objasniti proces grupiranja leksičkih jedinki u generatoru leksičkog analizatora (str. 57)
17. Opisati dinamiku izvođenja leksičke analize.
18. Opisati program simulator leksičkog analizatora zasnovan na tablici prijelaza epsilon-DKA.
19. Navesti postupak izgradnje kanonskog LR parsera na temelju izgrađenog DKA.
20. Objasniti postupak određivanja relacija prednosti na temelju zadane gramatike.
21. Opisati primjenu lijevog faktoriranja u pretvorbi pravila produkcija tako da zadovoljavaju uvjete LL(1) gramatike.
22. Navesti algoritam za izračunavanje FIRST skupova za produkcije.
23. Opisati način izrade globalne tablice znakova.
24. Opisati što se nastoji utvrditi analizom toka podataka.( samo 69. I 70. Su iz 3.kolokvija)
25. Opisati postupak razrješavanja nejednoznačnosti pretraživanjem lijevog konteksta.
26. Opisati program simulator leksičkog analizatora zasnovan na tablici prijelaza DKA.
27. Opisati strukturu jezičnog procesora C.
28. Opisati kriterije vrednovanja jezičnih procesora.
29. Opisati postupak razrješavanja nejednoznačnosti pretraživanjem desnog konteksta.
30. Opisati program simulator leksičkog analizatora zasnovan na tablici prijelaza NKA.
31. Objasniti LALR metodu za gradnju tablice LR parsera.
32. Definirajte tipove nejednoznačnosti kod LR parsiranja i opišite razrješavanje nejednoznačnosti kod YACC-a.
33. Opisati rekurzivni algoritam koji ispituje strukturnu jednakost tipova obilježja.
34. Kod dinamičkog pravila određivanja djelokruga opisati dva osnovna načina njihovog ostvarenja.
35. Opisati postupak relociranja za generirani premjestivi kod.
36. Opisati podjelu jezičnih procesora s obzirom na funkciju preslikavanja naredbi izvornog jezika u ciljni.
37. Opisati doprinos razvoja jezika ALGOL teoriji jezičnih procesora.
38. Kojim funkcijama je potrebno proširiti generator konačnog automata da bi se izgradio generator leksičkog analizatora?
39. Opisati postupak razrješavanja nejednoznačnosti pretraživanjem desnog konteksta u leksičkoj analizi.
40. Pomoću COBOL notacije opisati BNF notaciju.
41. Opišite osnovne značajke jezikaPL/1 (29. Ili 39.str)
42. Opišite rad dinamičkog interpretera.
43. Opišite postupak oporavka od pogrešaka u leksičkoj analizi.
44. Opišite program simulator leksičkog analizatora zasnovan na tablici prijelaza nedeterminističkog konačnog automata s epsilon prijelazima.
45. Opišite postupak konstrukcije epsilon-NKA koji prihvaća skup svih mogućih živih prefiksa za zadanu gramatiku.
46. Opišite formalne metode semantičkog analizatora.
47. Opišite postupke oporavka od pogrešakaza parsiranje tehnikom od vrha prema dnu.
48. Opišite način prenošenja semantičkih svojstava u tehnici rekurzivnog spusta.
49. Opišite postupak određivanja relacija prednosti na temelju zadane gramatike.
50. Opisati osnovne postupke u procesu optimiranja ciljnog koda.
51. Opisati postupke relociranja i rezolucije tijekom pripreme ciljnog programa za izvođenje.
52. Navesti algoritam za izračunavanje FIRST skupova za produkcije.

99)Opišite nasljedna i izvedena svojstva atributnih gramatika i na koji se način ta svojstva prennose kod tehnike rekruviznog spusta i potisnog automata.