

INDEKS-SEKVENCIJALNA DATOTEKA

Vežbe

Sadržaj

- Zadaci
- Fizička struktura indeks-sekvencijalne datoteke
- Formiranje indeks-sekvencijalne datoteke
- Traženje sloga
- Upis novog sloga
- Modifikacija sloga
- Logičko brisanje sloga

Zadaci

Zadatak 1

- Napisati C program koji će omogućiti rad sa podacima o evidentiranim prolascima kroz naplatnu rampu kod mesta Grad. Evidencija se vrši u datoteci sa statičkom indeks-sekvencijalnom organizacijom koja koristi zonu indeksa sa propagacijom najvećih vrednosti ključa iz svakog bloka i direktno povezivanje prekoračilaca.
- Za svaku evidenciju beleži se:
 - IDP - identifikacioni broj prolaska [8 cifara]
 - ROV - registarska oznaka vozila [10 karaktera]
 - OKV - oznaka kategorije vozila [2 karaktera]
 - DVP - datum i vreme prolaska [DD-MM-YYYY_HH:MM:SS]
 - PIZ - plaćeni iznos [6 cifara]
 - AKT - oznaka da li je slog logički obrisani [1 karakter: 'A' - aktivan, 'O' - obrisani]

Zadatak 1

- Omogućiti:
 - a. Formiranje datoteke na osnovu slogova iz ulazne sekvencijalne datoteke
 - b. Unos novog sloga
 - c. Modifikaciju sloga
 - d. Logičko brisanje sloga
 - e. Traženje po identifikacionom broju prolaska (IDP)
 - f. Ispis svih slogova

Zadatak 1

- Prilikom implementacije, ispoštovati sledeća ograničenja:
 - vrednost faktora blokiranja u primarnoj zoni f je 4,
 - red stabla traženja u zoni indeksa n je 2,
 - vrednost faktora blokiranja za zonu prekoračenja fz je 1,
 - vrednost faktora popunjenosti bloka pri formiranju datoteke ibf je 75%
 - prilikom rada sa datotekom, dozvoliti preuzimanje i upis isključivo čitavih blokova.

Zadatak 2

- Napisati C program koji će omogućiti rad sa podacima o evidentiranim isporukama kurirske službe *Quick*. Evidencija se vrši u datoteci sa statičkom indeks-sekvencijalnom organizacijom koja koristi zonu indeksa sa propagacijom najmanjih vrednosti ključa iz svakog bloka i indirektno povezivanje prekoračilaca.
- Za svaku evidenciju beleži se:
 - **IDI** - identifikacioni broj isporuke [8 cifara]
 - **ADP** - adresa pošaljioca [50 karaktera]
 - **ADP** - adresa primaoca [50 karaktera]
 - **DVP** - datum i vreme isporuke [DD-MM-YYYY_HH:MM:SS]
 - **CEN** - cena isporuke [6 cifara]
 - **AKT** - oznaka da li je slog logički obrisani [1 karakter: 'A' - aktivan, 'O' - obrisani]

Zadatak 2

- Omogućiti:
 - a. Formiranje sekvencijalne datoteke na osnovu slogova iz ulazne serijske datoteke
 - b. Formiranje indeks-sekvencijalne datoteke na osnovu sekvencijalne datoteke
 - Osloniti se na pristup u kom se podaci za svaku od zona skladište u zasebnim datotekama
 - c. Unos novog sloga
 - d. Modifikaciju sloga
 - e. Logičko brisanje sloga
 - f. Traženje po identifikacionom broju isporuke (IDI)
 - g. Ispis svih slogova

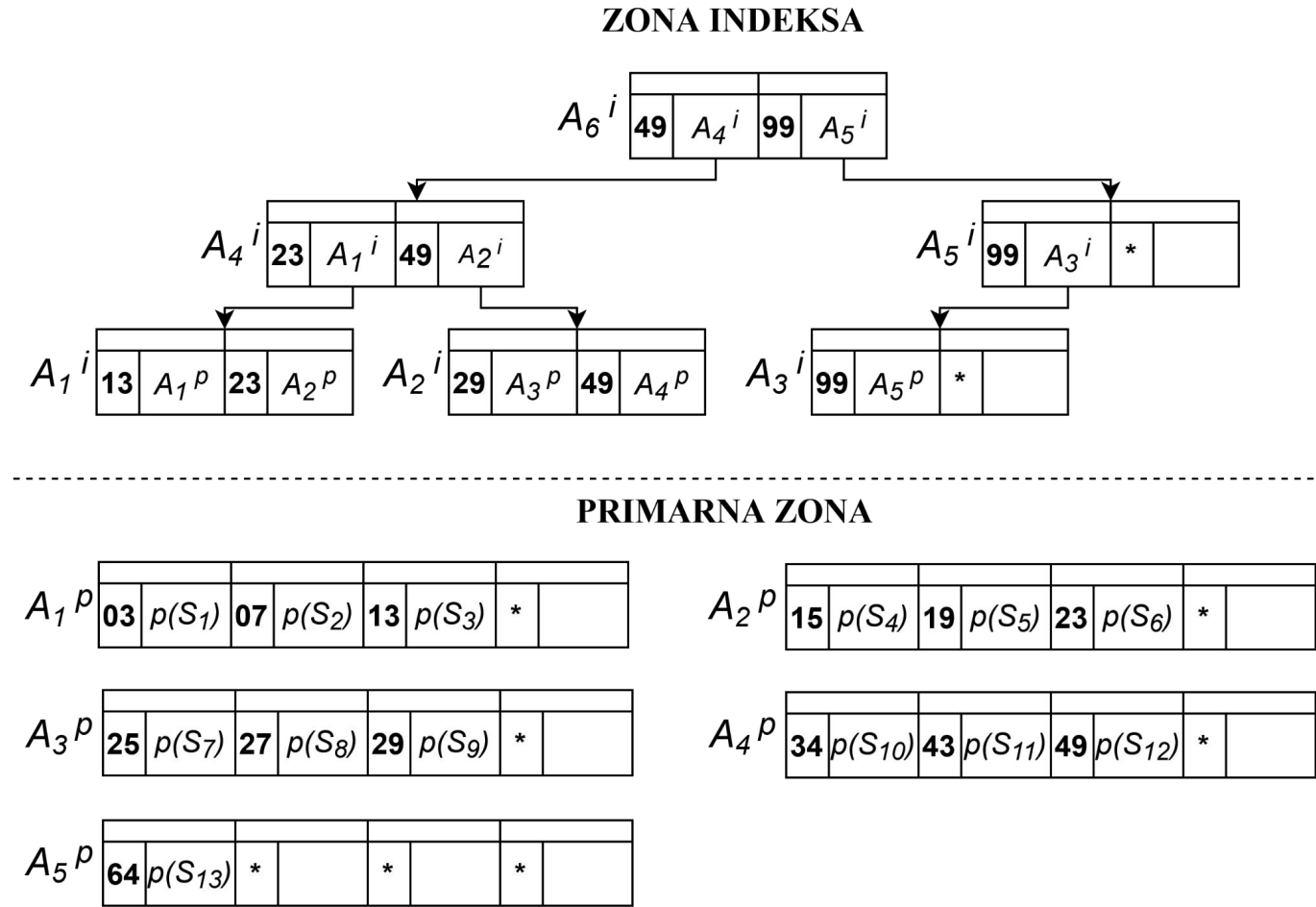
Zadatak 2

- Prilikom implementacije, ispoštovati sledeća ograničenja:
 - vrednost faktora blokiranja u primarnoj zoni f je 5,
 - red stabla traženja u zoni indeksa n je 2,
 - vrednost faktora blokiranja za zonu prekoračenja fz je 1,
 - vrednost faktora popunjenosti bloka pri formiranju datoteke ibf je 80%
 - prilikom rada sa datotekom, dozvoliti preuzimanje i upis isključivo čitavih blokova.

Fizička struktura indeks-sekvencijalne datoteke

Fizička struktura ind-sekv datoteke - **Primer 1.1**

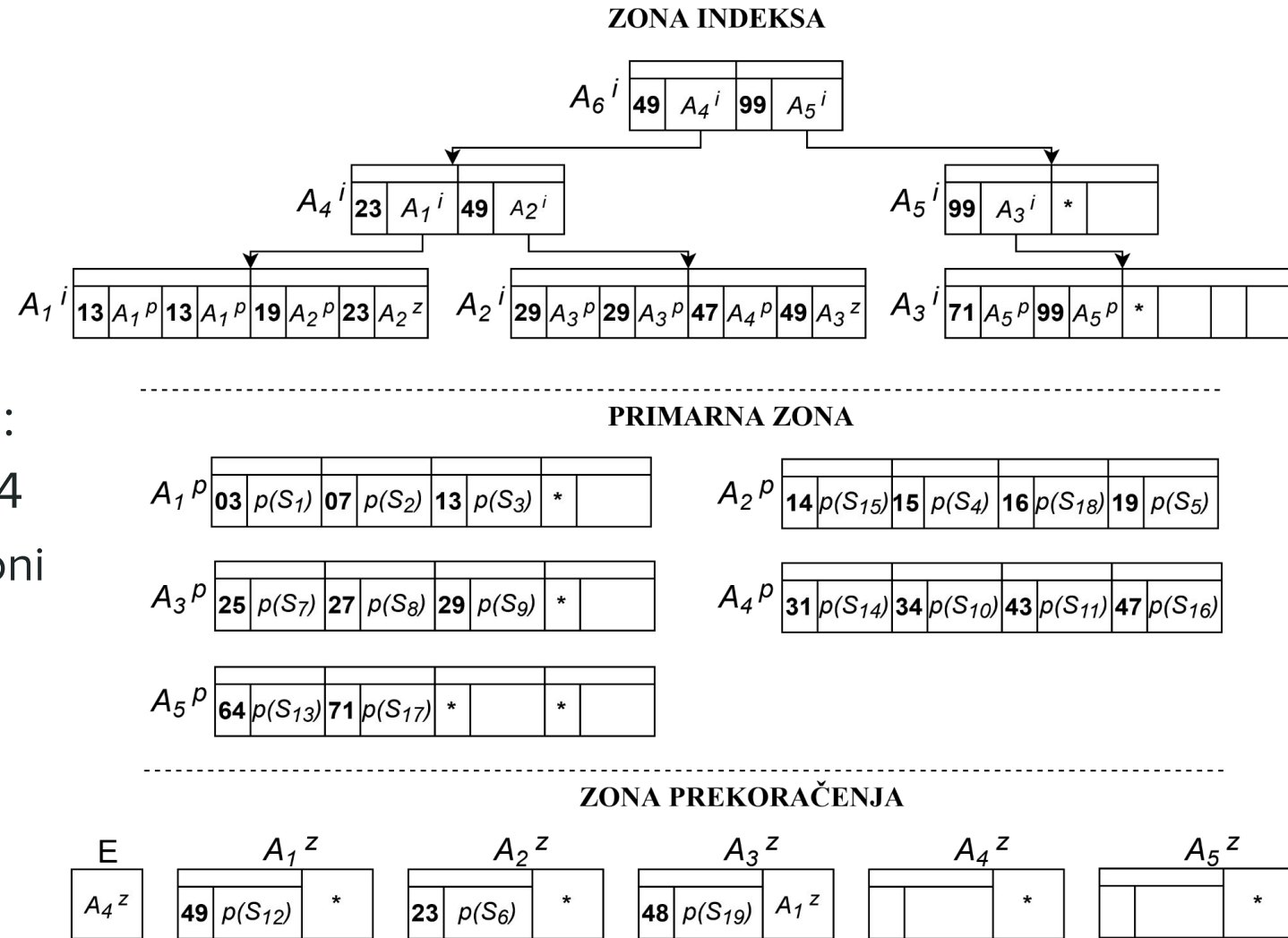
- Za početni ulazni skup od $N = 13$ slogova, sa vrednostima ključa [3, 7, 13, 15, 19, 23, 25, 27, 29, 34, 43, 49, 64], potrebno je rasporediti slogove u statičku indeks-sekvencijalnu datoteku:
- Faktor blokiranja u primarnoj zoni b je 4
- Red stabla traženja i faktor blokiranja u zoni indeksa n je 2
- **Propagacija najvećih vrednosti ključa**
- Faktor popunjenosti bloka pri formiranju datoteke ibf je 75%



Slika 1. Prikaz fizičke strukture ind-sekv datoteke za primer 1.1

Fizička struktura ind-sekv datoteke - Primer 1.2

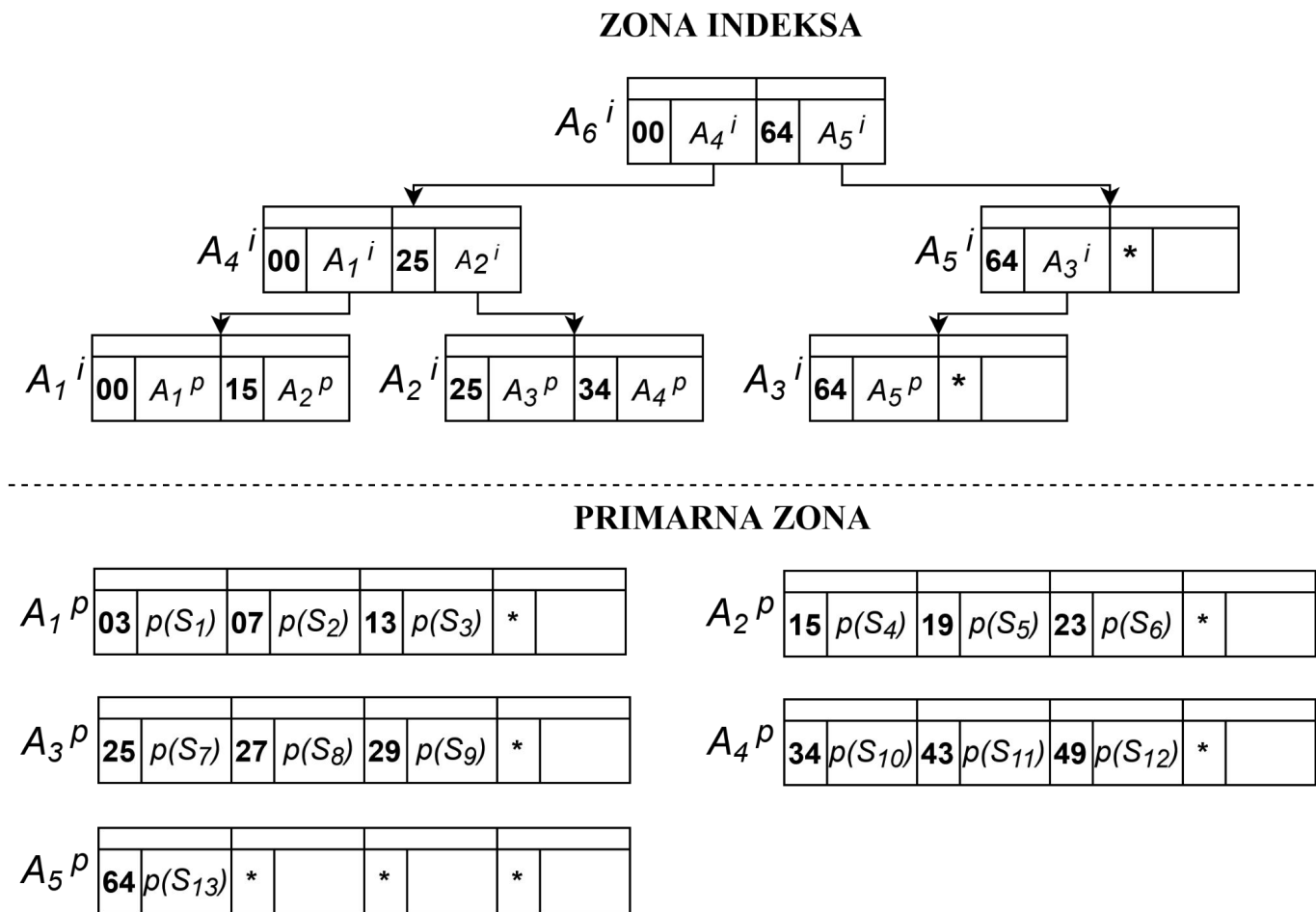
- Za početni ulazni skup od $N = 13$ slogova, sa vrednostima ključa [3, 7, 13, 15, 19, 23, 25, 27, 29, 34, 43, 49, 64], potrebno je rasporediti slogove u statičku indeks-sekvencijalnu datoteku:
- Faktor blokiranja u primarnoj zoni b je 4
- Red stabla traženja i faktor blokiranja u zoni indeksa n je 2
- Propagacija najvećih vrednosti ključa
- **Direktno povezivanje prekoračilaca**
 - Za naknadno uneti skup slogova sa vrednostima ključa [31, 14, 47, 71, 16, 48]



Slika 2. Prikaz fizičke strukture ind-sekv datoteke za primer 1.2

Fizička struktura ind-sekv datoteke - **Primer 2.1**

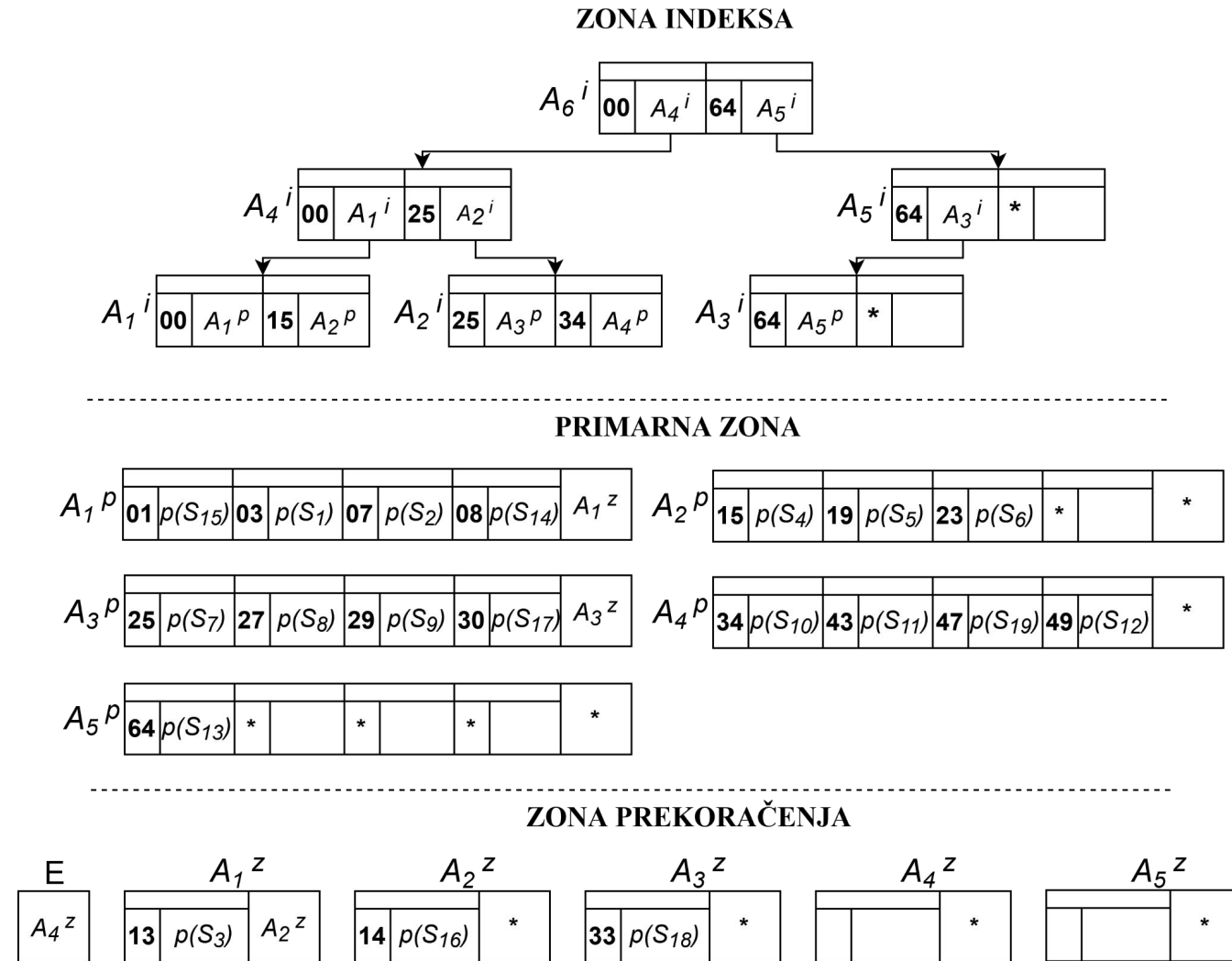
- Za početni ulazni skup od $N = 13$ slogova, sa vrednostima ključa [3, 7, 13, 15, 19, 23, 25, 27, 29, 34, 43, 49, 64], potrebno je rasporediti slogove u statičku indeks-sekvencijalnu datoteku:
- Faktor blokiranja u primarnoj zoni b je 4
- Red stabla traženja i faktor blokiranja u zoni indeksa n je 2
- **Propagacija najmanjih vrednosti ključa**
- Popunjenost bloka pri formiranju datoteke ibf je 75%



Slika 3. Prikaz fizičke strukture ind-sekv datoteke za primer 2.1

Fizička struktura ind-sekv datoteke - **Primer 2.2**

- Za početni ulazni skup od $N = 13$ slogova, sa vrednostima ključa [3, 7, 13, 15, 19, 23, 25, 27, 29, 34, 43, 49, 64], potrebno je rasporediti slogove u statičku indeks-sekvencijalnu datoteku:
- Faktor blokiranja u primarnoj zoni b je 4
- Red stabla traženja i faktor blokiranja u zoni indeksa n je 2
- Propagacija najmanjih vrednosti ključa
- **Indirektno povezivanje prekoračilaca**
 - Za naknadno uneti skup slogova sa vrednostima ključa [8, 1, 14, 33, 30, 47]



Slika 4. Prikaz fizičke strukture ind-sekv datoteke za primer 2.2

Fizička struktura ind-sekv datoteke - matematičke formule

- Visina stabla (B - broj blokova u prim zoni; n - red stabla)

$$h = \left\lceil \log_n B \right\rceil$$

- Broj čvorova na i -tom nivou hijerarhije stabla ($i=1, \dots, h$)

$$C_i = \left\lceil \frac{B}{n^{h-i+1}} \right\rceil$$

- Ukupni broj čvorova stabla

$$C = \sum_{i=1}^h \left\lceil \frac{B}{n^{h-i+1}} \right\rceil$$

Formiranje indeks-sekvencijalne datoteke

Formiranje indeks-sekvencijalne datoteke - Koraci

1. Definisane strukture
2. Zauzimanje prostora za zonu indeksa
 - a. Obuhvata računanje broja neterminalnih čvorova i zapis praznih neterminalnih čvorova u datoteku na osnovu broja ulaznih slogova
 - b. Analogno, obuhvata računanje broja listova i zapis praznih listova u datoteku na osnovu broja ulaznih slogova
3. Pozivanje procesa *formiranje_primarne_zone*
4. Pozivanje procesa *formiranje_zone_indeksa*
5. Formiranje zone prekoračenja
 - a. Upis zaglavlja zone
 - b. Upis predefinisano broja slogova - svaki slog sadrži pokazivač na naredni u lancu

Formiranje indeks-sekvencijalne datoteke - Definisiranje strukture

- Definisiranje strukture indeks-sekvencijalne datoteke obuhvata **određivanje**:
 - Struktura primarne zone (struktura sloga i bloka)
 - Struktura neterminalnog čvora (čvora koji nije list) i njegovog sloga
 - Struktura terminalnog čvora (lista) i njegovog sloga
 - Struktura zaglavlja i sloga zone prekoračenja
 - Faktor blokiranja
 - Faktor popunjenosti
 - Red stabla

PROCES formiranje(U(*naziv_datoteke*, *ulazna_datoteka*), I(), UI())

POČETAK PROCESA formiranje

POZOVI zauzimanje_prostora_za_zonu_indeksa(U(*naziv_datoteke*), I(), UI())

POZOVI formiranje_primarne_zone(U(*naziv_datoteke*, *ulazna_datoteka*), I(), UI())

POZOVI formiranje_zone_indeksa(U(*naziv_datoteke*), I(), UI())

POZOVI formiranje_zone_prekoracenja(U(*naziv_datoteke*), I(), UI())

KRAJ PROCESA formiranje

```

PROCES zauzimanje_prostora_za_zonu_indeksa(U(naziv_dat), I())
POČETAK PROCESA
    OTVORI_DATOTEKU(U(serijska_dat, režim_otvaranja), I(status_otvaranja_datoteke))
    IZRAČUNAJ broj_neterminalnih_cvorova
    POSTAVI neterminalni_čvor <- prazan_neterminalni_cvor
    POSTAVI i <- 0
    RADI DOK JE i < broj_neterminalnih_cvorova
        PIŠI_ČVOR(indeks_sekv_datoteka, adresa_čvora, status_pisanja, neterminalni_cvor)
        POSTAVI i <- i + 1
    KRAJ RADI
    IZRAČUNAJ broj_listova
    POSTAVI list <- prazan_list
    POSTAVI i <- 0
    RADI DOK JE i < broj_listova
        PIŠI_LIST(indeks_sekv_datoteka, adresa_lista, status_pisanja, list)
        POSTAVI i <- i + 1
    KRAJ RADI
    ZATVORI datoteku
KRAJ PROCESA

```

PROCES formiranje_primarne_zone(U(naziv_dat, ulazna_dat), I())

POČETAK PROCESA

OTVORI_DATOTEKU(U(ulazna_dat, režim_otvaranja), I(status_otvaranja_datoteke))

OTVORI_DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))

IZRAČUNAJ velicina_zone_indeksa

POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)

IZRAČUNAJ broj_ulaznih_slogova

IZRAČUNAJ maks_slogova_po_bloku

IZRAČUNAJ broj_blokova_u_primarnoj_zoni

POSTAVI tekuci_list <- prazan_list

POSTAVI broj_preuzetih_slogova <- 0

POSTAVI maks_ključ_bloka <- 0 // Propagacija najveće vrednosti ključa

POSTAVI i <- 0

RADI DOK JE i < broj_blokova

POSTAVI blok_prim_zone <- prazan_blok

POSTAVI j <- 0

RADI DOK JE j < maks_slogova_po_bloku

ČITAJ_SLOG(U(ulazna_dat, redni_broj_sloga), I(status_čitanja), UI(trenutni_slog))

POSTAVI broj_preuzetih <- broj_preuzetih + 1

POSTAVI maks_ključ_bloka <- ključ_trenutnog_sloga

NASTAVAK NA SLEDEĆEM SLAJDU

NASTAVAK

```
AKO JE broj_preuzetih = broj_ulaznih_slogova
    POSTAVI maks_ključ_bloka <- MAKS_ID // MAKS_ID se zadaje unapred
    KRAJ RADI
```

```
KRAJ AKO
```

```
POSTAVI j <- j + 1
```

```
KRAJ RADI
```

```
PIŠI_BLOK(indeks_sekv_datoteka, adresa_bloka, status_pisanja, blok)
```

```
// U istom prolazu može se odraditi i punjenje listova zone indeksa
```

```
POSTAVI tekuci_slog[i MOD RED_STABLA].idMaxBlok <- maks_ključ_bloka
```

```
POSTAVI tekuci_slog[i MOD RED_STABLA].adresaBlok <- i
```

```
POSTAVI tekuci_slog[i MOD RED_STABLA].idMax <- maks_ključ_bloka
```

```
POSTAVI tekuci_slog[i MOD RED_STABLA].adresaMinPrekoracioca <- -1
```

```
AKO JE i MOD RED_STABLA = RED_STABLA-1 ILI broj_preuzetih_slogova = broj_ulaznih_slogova
```

```
TADA
```

```
    PIŠI_LIST(zona_indeksa, adresa_lista, status_pisanja, tekuci_list)
```

```
    POSTAVI tekuci_list <- prazan_list
```

```
KRAJ AKO
```

```
POSTAVI i <- i + 1
```

```
KRAJ RADI
```

```
ZATVORI_DATOTEKU(ulazna_dat)
```

```
ZATVORI_DATOTEKU(indeks_sekv_dat)
```

```
KRAJ PROCESA
```

PROCES formiranje_zone_indeksa(U(naziv_dat), I())

POČETAK PROCESA

OTVORI_DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))

POSTAVI tekuci_cvor <- prazan_neterminalni_cvor

INICIJALIZUJ_LIST(I(list))

INICIJALIZUJ_NETERMINALNI_CVOR(I(neterminalni_cvor))

POSTAVI s <- VISINA_STABLA

RADI DOK JE s > 1

POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), broj_cvorova_u_visim_slojevima *
veličina(NETERM_CVOR))

POSTAVI kljuc_za_propagaciju <- 0

POSTAVI i <- 0

RADI DOK JE i < broj_cvorova_u_sloju

AKO JE sloj = VISINA_STABLA **TADA**

ČITAJ_LIST(U(indeks_sekv_dat, redni_broj_lista), I(status_čitanja), UI(list))

ODREDI najveći_kljuc_iz_lista

INAČE

ČITAJ_NETERMINALNI_ČVOR(U(indeks_sekv_dat, redni_broj_čvora), I(status_čitanja),
UI(neterminalni_cvor))

ODREDI najveći_kljuc_iz_cvora

KRAJ AKO

NASTAVAK NA SLEDEĆEM SLAJDU

NASTAVAK

```
POSTAVI tekuci_slog[i MOD RED_STABLA].id <- kljuc_za_propagaciju
POSTAVI tekuci_slog[i MOD RED_STABLA].adresa <- broj_cvorova_u_visim_slojevima + i
AKO JE i MOD RED_STABLA = RED_STABLA - 1 ILI i = broj_čvorova_u_sloju - 1 TADA
    PIŠI_ČVOR(indeks_sekv_dat, adresa_tekuceg_cvora, status_pisanja, tekuci_cvor)
    POSTAVI tekuci_cvor <- prazan_neterminalni_cvor
KRAJ AKO
POSTAVI i <- i + 1
KRAJ RADI
KRAJ RADI
ZATVORI_DATOTEKU(indeks_sekv_dat)
KRAJ PROCESA
```

PROCES formiranje_zone_prekoracenja(U(naziv_dat), I())

POČETAK PROCESA

OTVORI_DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))

POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa +
velicina_primarne_zone)

POSTAVI zaglavlje_zone_prekoracenja <- prazno_zaglavlje_zone_prekoracenja

PIŠI_ZAGLAVLJE_ZONE_PREKORACENJA(indeks_sekv_dat, adresa_zaglavlja, status_pisanja, zaglavlje)

POSTAVI slog_zone_prekoracenja <- prazan_slog_prekoracenja

PIŠI_SLOG(indeks_sekv_dat, adresa_sloga, status_pisanja, slog_zone_prekoracenja)

ZATVORI_DATOTEKU(indeks_sekv_dat)

KRAJ PROCESA

Napomene

- * Putem pseudokoda predstavljena je indeks-sekvencijalna datoteka smeštena u jednu fizičku datoteku, u kojoj se nalaze sve tri zone. Implementacija može da se razdvoji i na tri odvojene fizičke datoteke.
- * Zona indeksa formira se propagacijom najveće vrednosti ključa. U slučaju propagacije najmanje vrednosti ključa, u algoritmu treba izmeniti da se čuva vrednost ključa za prvi slog u bloku, jer je po sekvencijalnoj organizaciji ona je najmanja za tekući blok.
- * U pseudokodu je korišćen pristup za direktno povezivanje prekoračilaca sa listom stabla traženja, što znači da se u zoni indeksa čuvaju pokazivači na zonu prekoračenja. U slučaju indirektnog povezivanja prekoračioca, potrebno je u svakom bloku primarne zone čuvati i pokazivač na početak lanca prekoračilaca bloka u zoni prekoračenja; takođe, što se tiče zone indeksa, u tom slučaju ne postoji razlika između strukture listova i strukture ostalih čvorova zone indeksa.

Traženje sloga

Traženje sloga - Koraci

1. Definicija strukture rezultata traženja sloga
 - a. Vraća adresu bloka pronađenog sloga, adresu pronađenog sloga, pronađen slog i indikator zone u kojoj se nalazi (0 - primarna zona, 1 - zona prekoračenja)
2. Otvaranje indeks-sekvencijalne datoteke
3. Prolazak kroz neterminalne čvorove (kreće se od korena stabla)
 - a. Ukoliko je traženi ključ manji od ključa sloga u tekućem neterminalnom slogu, prelazi se na odgovarajuću adresu (par *ključ-adresa*)
4. Prolazak kroz terminalne čvorove (listove)
 - a. Ukoliko je traženi ključ manji ili jednak maksimalnom ključu u trenutnom slogu lista
 - i. Ukoliko je manji ili jednak maksimalnom ključu odgovarajućem bloku primarne zone, poziva se proces *trazenje_u_primarnoj_zoni*
 - ii. U suprotnom, ukoliko postoje prekoračioc, poziva se proces *trazenje_u_zoni_prekoracenja*
5. Povratna vrednost je tip strukture rezultata traženja sloga

```

PROCES trazenje_sloga(U(naziv_dat, kljuc), I(rezultat_trazenja, adr_bloka, adr_sloga, zona) UI())
POČETAK PROCESA trazenje_sloga
    OTVORI_DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))
    POSTAVI rezultat_trazenja ← prazan_rezultat
    INICIJALIZUJ LIST
    POSTAVI relativna_adresa ← 0
    POSTAVI i ← 0
    RADI DOK JE i < visina_stabla - 1
        AKO JE i != visina_stabla - 1 TADA
            POMERI_POKAZIVAČ(U(datoteka, početak), relativna_adresa * veličina_neterminalnog_čvora)
            ČITAJ_NETERMINALNI_ČVOR(U(indeks_sekv_dat, relativna_adresa), I(status_čitanja),
            UI(neterminalni_cvor))
            POSTAVI j ← 0
            RADI DOK JE j < RED_STABLA
                AKO JE kljuc <= ključ_trenutnog_sloga TADA
                    POSTAVI relativna_adresa ← adresa_trenutnog_sloga
                    IZAĐI IZ PETLJE
                INAČE
                    POSTAVI j ← j + 1
                KRAJ AKO
            KRAJ RADI
        KRAJ RADI

```

NASTAVAK NA SLEDEĆEM SLAJDU

INAČE

IZRAČUNAJ broj_neterminalnih_cvorova_do_sloja

POSTAVI broj_listova_za_preskakanje \leftarrow relativna_adresa - broj_neterminalnih_cvorova

ČITAJ_LIST(U(indeks_sekv_dat, relativna_adresa), I(status_čitanja), UI(list))

POSTAVI j \leftarrow 0

RADI DOK JE j < RED_STABLA

AKO JE kljuc \leq max_kljuc_trenutnog_sloga_lista **TADA**

AKO JE kljuc \leq max_kljuc_bloka_trenutnog_sloga_lista **TADA**

POZOVI rezultat_trazenja \leftarrow trazenje_u_primarnoj_zoni(adresa_bloka, kljuc)

POSTAVI zona \leftarrow 0

INAČE

AKO JE adresa_min_prekoracioca_u_trenutnom_listu = -1 **TADA**

IZAĐI IZ PETLJE

INAČE

POZOVI rezultat_trazenja \leftarrow trazenje_u_zoni_prekoracenja(adresa_prek, kljuc)

POSTAVI zona \leftarrow 1

KRAJ AKO

KRAJ AKO

KRAJ RADI

NASTAVAK NA SLEDEĆEM SLAJDU

INAČE

POSTAVI $j \leftarrow j + 1$

KRAJ AKO

KRAJ RADI

KRAJ AKO

KRAJ ZA

ZATVORI_DATOTEKU(*indeks_sekv_dat*)

VRATI rezultat_trazenja

KRAJ PROCESA trazenje_sloga

PROCES trazenje_u_primarnoj_zoni(U(indeks_sekv_dat, adresa_bloka, kljuc), I(rezultat_trazenja, adr_bloka, adr_sloga), UI(blok_primarni))

POČETAK PROCESA trazenje_u_primarnoj_zoni

POSTAVI rezultat_trazenja ← prazan_rezultat

POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)

POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), adresa_bloka * veličina(BLOK_PRIM))

ČITAJ_BLOK(U(indeks_sekv_dat, adresa_bloka), I(status_čitanja), UI(blok_primarni))

POSTAVI i ← 0

RADI DOK JE i < FAKTOR_BLOKIRANJA

AKO JE kljuc_trenutnog_sloga = kljuc **TADA**

POSTAVI rezultat_trazenja.adresa_bloka ← adresa_trenutnog_bloka

POSTAVI rezultat_trazenja.adresa_trenutnog_sloga ← i

POSTAVI rezultat_trazenja.slog ← trenutni_slog

POSTAVI rezultat_trazenja.zona ← 0

VRATI rezultat_trazenja

INAČE

AKO JE kljuc_trenutnog_sloga > kljuc **TADA**

VRATI rezultat_trazenja

INAČE

POSTAVI i ← i + 1

KRAJ AKO

KRAJ AKO

KRAJ RADI

VRATI rezultat_trazenja

KRAJ PROCESA trazenje_u_primarnoj_zoni

```

PROCES trazenje_u_zoni_prekoracenja(U(indeks_sekv_dat, adresa_min_prek, kljuc), I(rezultat_trazenja, adr_bloka,
adr_sloga), UI(slog_prekoracenja))
POČETAK PROCESA trazenje_u_zoni_prekoracenja
  POSTAVI rezultat_trazenja ← prazan_rezultat
  INICIJALIZUJ_PREKORAČIOCA(I(slog_prekoracenja))
  POSTAVI trenutna_adresa ← adresa_minimalnog_prekoracioca
  RADI DOK JE trenutna_adresa != -1
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_prekoracenja)
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), veličina(ZAGLAVLJE_ZONE_PREKORACENJA) + trenutna_adresa *
veličina(SLOG_ZONE_PREKORACENJA))
    ČITAJ_BLOK(U(indeks_sekv_dat, trenutna_adresa), I(status_čitanja), UI(slog_prekoracenja))
    AKO JE kljuc_trenutnog_sloga = kljuc TADA
      POSTAVI rezultat_trazenja.adresa_bloka ← trenutna_adresa
      POSTAVI rezultat_trazenja.adresa_trenutnog_sloga ← 0
      POSTAVI rezultat_trazenja.slog ← trenutni_slog
      POSTAVI rezultat_trazenja.zona ← 1
      IZAĐI IZ PETLJE
    INAČE
      AKO JE kljuc_trenutnog_sloga > kljuc TADA
        KRAJ RADI
      INAČE
        POSTAVI trenutna_adresa ← adresa_narednog_sloga
      KRAJ AKO
    KRAJ AKO
  KRAJ RADI
  VRATI rezultat_trazenja
KRAJ PROCESA trazenje_u_zoni_prekoracenja

```

Upis novog sloga

Upis novog sloga - Koraci

1. Pozivanje procesa traženja:
 - a. Pozvati *trazenje_sloga* sa datim podacima i ključem slog-a
 - i. Ovaj proces će pokušati pronaći odgovarajući ključ sloga prvobitno u zoni indeksa, a zatim i u primarnoj zoni ili zoni prekoračilaca
2. Provera rezultata traženja:
 - a. Ako je rezultat funkcije traženja neuspešan i nastavljamo sa upisom (stavka 3.)
 - b. Ako je rezultat funkcije traženja uspešan završavamo proces dodavanja novog sloga
3. Nastavak upisa:
 - a. Pozivamo proces *pronadji_list*, kako bismo našli odgovarajući list za upis sloga
 - b. Ukoliko je ključ novog sloga manji ili jednak najvećem ključu u odgovarajućem listu, pokušavamo da dodamo slog u primarnu zonu. Ukoliko je neuspešno, dodajemo slog u zonu prekoračenja

Upis novog sloga - Koraci

c. Ukoliko je ključ veći od najvećeg ključa u odgovarajućem listu, upisujemo slog u zonu prekoračenja

5. Na kraju upisujemo novi ključ sloga u odgovarajući list

Upis novog sloga - Pseudokod

PROCES dodaj_slog_indeks_sekvencijalna(U(naziv_datoteke, novi_slog), I(status))

POČETAK PROCESA dodaj_slog_indeks_sekvencijalna

AKO JE novi_slog.id > MAKS_ID

POSTAVI status <- 2

VRATI

KRAJ AKO

POSTAVI rt <- trazenje_sloga(U(naziv_datoteke, novi_slog.id))

AKO JE rt.adresaBloka != -1

POSTAVI status <- 2 (* slog već postoji *)

VRATI

KRAJ AKO

OTVORI_DATOTEKU(U(naziv_datoteke), I(status_otvaranja_datoteke), UI(f))

AKO JE status_otvaranja_datoteke != 0

POSTAVI status <- 2

VRATI

KRAJ AKO

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

```
POZOVI PRONADJI_LIST_ZA_ID(U(f, novi_slog.id),I(list,listIndex,j,adrLista,status_pronalaska))
AKO JE status_pronalaska != 1
    ZATVORI_DATOTEKU( U(f), I(status_zatvaranja) )
    POSTAVI status <- 2
    VRATI
KRAJ AKO
POSTAVI ls <- list.slogovi[j]
AKO JE novi_slog.id <= ls.idMaxBlok
    POSTAVI izbacen <- 0
    POZOVI dodaj_u_primarnu_zonu(U(f, ls.adresaBloka, novi_slog),I(izbaceni_slog, izbacen))
    AKO JE izbacen == 1
        POZOVI
            dodaj_u_zonu_prekoracenja(U(f,zbaceni_slog,I(ls.adresaMinPrekoracioca),
                                                    I(ls.idMax)),I(status_zp))

        AKO JE status_zp != 0
            ZATVORI_DATOTEKU( U(f), I(st) )
            POSTAVI status <- 2
            VRATI
        KRAJ AKO
    KRAJ AKO
```

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

INAČE

(* novi je veći od idMaxBlok *)

AKO JE ls.adresaMinPrekoracioca != -1

POZOVI

dodaj_u_zonu_prekoracenja(U(f, novi_slog, I(ls.adresaMinPrekoracioca, ls.idMax, status_zp)))

AKO JE status_zp != 0

ZATVORI_DATOTEKU(U(f), I(st))

POSTAVI status <- 2

VRATI

KRAJ AKO

INAČE

(* ako nema lanca, ubaci u primarnu; izlazi se ignorišu *)

POZOVI dodaj_u_primarnu_zonu(U(f, ls.adresaBloka, novi_slog), I(_, _))

KRAJ AKO

KRAJ AKO

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

```
(* ažuriranje idMaxBlok čitanjem bloka *)  
  POSTAVI blok <- procitaj_blok_primarne( U(f, ls.adresaBloka) )  
  POSTAVI ls.idMaxBlok <- maksimum_ključa_u_bloku( UI(blok) )  
  
  (* upiši list nazad u datoteku *)  
  POSTAVI list.slogovi[j] <- ls  
  UPIŠI_LIST( U(f, adrLista), UI(list) )  
  
  ZATVORI_DATOTEKU( U(f), I(status_zatvaranja) )  
  POSTAVI status <- 0  
  
KRAJ PROCESA dodaj_slog_indeks_sekvencijalna
```

Upis novog sloga - Pseudokod

PROCES dodaj_u_primarnu_zonu(U(f, adresaBloka, novi_slog), I(izbaceni_slog, izbacen))

POČETAK PROCESA dodaj_u_primarnu_zonu

ČITAJ_BLOK_PRIM(U(f, adresaBloka), UI(blok))

(* nađi poziciju: prvi prazan (-1) ili prvi veći od novog *)

POSTAVI i <- 0

DOK JE i < FAKTOR_BLOKIRANJA I blok.slogovi[i].id != -1 I novi_slog.id >= blok.slogovi[i].id

POSTAVI i <- i + 1

KRAJ DOK

POSTAVI izbacen <- 0

POSTAVI blokPun <- (blok.slogovi[FAKTOR_BLOKIRANJA-1].id != -1)

AKO JE blokPun == 1 I i == FAKTOR_BLOKIRANJA

(* novi je veći od svih, blok pun -> ne ubacuje se ništa *)

VRATI

KRAJ AKO

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

```
AKO JE blokPun == 1  
    POSTAVI izbaceni_slog <- blok.slogovi[FAKTOR_BLOKIRANJA-1]  
    POSTAVI izbacen <- 1
```

```
KRAJ AKO
```

```
(* pomeri slogove udesno da se oslobodi pozicija i *)
```

```
ZA j = FAKTOR_BLOKIRANJA-1 DO i+1 KORAK -1  
    POSTAVI blok.slogovi[j] <- blok.slogovi[j-1]
```

```
KRAJ ZA
```

```
POSTAVI blok.slogovi[i] <- novi_slog
```

```
UPIŠI_BLOK_PRIM( U(f, adresaBloka), UI(blok) )
```

```
KRAJ PROCESA dodaj_u_primarnu_zonu
```

Upis novog sloga - Pseudokod

PROCES dodaj_u_zonu_prekoracenja(U(f, novi_slog, I(adresaGlave), I(idMax)), I(status))

POČETAK PROCESA dodaj_u_zonu_prekoracenja

(* proveri da li postoji slobodna ZP lokacija i uzmi je *)

PROČITAJ_ZAGLAVLJE_ZP(U(f), UI(zag))

AKO JE zag.adresaPrvogSlobodnog == -1

POSTAVI status <- 2 (* nema slobodnih lokacija u ZP *)

VRATI

KRAJ AKO

POSTAVI novaAdresa <- zag.adresaPrvogSlobodnog

POSTAVI zag.adresaPrvogSlobodnog <- zag.adresaPrvogSlobodnog + 1

UPIŠI_ZAGLAVLJE_ZP(U(f), UI(zag))

POSTAVI noviZP.slog <- novi_slog

POSTAVI noviZP.sledeci <- -1

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

```
AKO JE adresaGlave == -1
    (* lanac ne postoji: novi postaje glava *)
    POSTAVI adresaGlave <- novaAdresa
INAČE
    (* pronađi mesto u sortiranom lancu *)
    POSTAVI prethodni <- -1
    POSTAVI tekuci <- adresaGlave
    DOK JE tekuci != -1
        ČITAJ_ZP( U(f, tekuci), UI(zp) )
        AKO JE novi_slog.id < zp.slog.id
            PREKINI
        KRAJ AKO
        POSTAVI prethodni <- tekuci
        POSTAVI tekuci <- zp.sledeci
    KRAJ DOK
```

NASTAVAK NA SLEDEĆEM SLAJDU

Upis novog sloga - Pseudokod

AKO JE prethodni == -1

(* ubacivanje na početak *)

POSTAVI noviZP.sledeci <- adresaGlave

POSTAVI adresaGlave <- novaAdresa

INAČE

(* ubacivanje posle prethodnog *)

ČITAJ_ZP(U(f, prethodni), UI(zpPrev))

POSTAVI noviZP.sledeci <- zpPrev.sledeci

POSTAVI zpPrev.sledeci <- novaAdresa

UPIŠI_ZP(U(f, prethodni), UI(zpPrev))

KRAJ AKO

KRAJ AKO

(* upiši novi čvor na dodeljenu adresu *)

UPIŠI_ZP(U(f, novaAdresa), UI(noviZP))

AKO JE novi_slog.id > idMax

POSTAVI idMax <- novi_slog.id

KRAJ AKO

POSTAVI status <- 0

Modifikacija sloga

Modifikacija sloga - Koraci

1. Pozivanje procesa traženja:
 - a. Pozvati *trazenje_sloga* koristeći identifikator (ključ) sloga.
2. Provera rezultata traženja:
 - a. Ako je rezultat funkcije traženja neuspešan završavamo proces modifikacije
 - b. Ako je rezultat funkcije traženja uspešan nastavljamo sa procesom modifikacije
3. Ažuriranje u primarnoj zoni
 - a. Pozicionirati se na tačnu adresu bloka unutar primarne zone
 - b. Izdvojiti postojeći ključ iz sloga koji se modifikuje
 - c. Izvršiti modifikaciju sloga
 - d. Formirati novi slog kombinovanjem postojećeg ključa sa modifikovanim podacima sloga
4. Ažuriranje u zoni prekoračenja
 - a. Preskočiti primarnu zonu i pozicionirati se na zonu prekoračenja
 - b. Izdvojiti postojeći ključ iz sloga koji se modifikuje
 - c. Izvršiti modifikaciju sloga
 - d. Formirati novi slog kombinovanjem postojećeg ključa sa modifikovanim podacima sloga

Modifikacija sloga - Koraci

5. Na osnovu rezultata pisanja bloka, postaviti konačni rezultat operacije na 0 (uspešno) ili 1 (neuspešno)

```
PROCES azuriraj_slog(U(naziv_dat, novi_slog), I(status_azuriranja), UI(blok_primarni, slog_prek))
```

```
POČETAK PROCESA azuriraj_slog
```

```
  OTVORI DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))
```

```
  POZOVI PROCES rezultat_trazenja <- trazenje_sloga(U(naziv_dat, novi_slog.id), I(), UI())
```

```
  AKO JE rezultat_trazenja.adresa_bloka = -1 TADA
```

```
    ZATVORI DATOTEKU(indeks_sekv_dat)
```

```
    POSTAVI status_azuriranja <- 0
```

```
    VRATI status_azuriranja
```

```
  KRAJ AKO
```

```
  AKO JE rezultat_trazenja.zona = 0 TADA
```

```
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)
```

```
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), rezultat_trazenja.adresa_bloka *  
    velicina(BLOK_PRIM))
```

```
    ČITAJ_BLOK(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_čitanja),  
    UI(blok_primarni))
```

```
    POSTAVI blok_primarni.slog[rezultat_trazenja.adresa_sloga] <- novi_slog
```

```
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)
```

```
    POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), rezultat_trazenja.adresa_bloka *  
    velicina(BLOK_PRIM))
```

```
    PIŠI_BLOK(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_pisanja),  
    UI(blok_primarni))
```

```
NASTAVAK NA SLEDEĆEM SLAJDU
```

NASTAVAK

INAČE

```
POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa +
velicina_primarne_zone)
POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), veličina(ZAGLAVLJE_ZONE_PREKORACENJA) +
rezultat_trazenja.adresa_bloka * veličina(SLOG_ZONE_PREKORACENJA))
ČITAJ_SLOG(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_čitanja),
UI(slog_zone_prekoracenja))
POSTAVI slog_zone_prekoracenja.slog <- novi_slog
POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), -veličina(SLOG_ZONE_PREKORACENJA))
PIŠI_SLOG(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_pisanja),
UI(slog_zone_prekoracenja))
KRAJ AKO
ZATVORI_DATOTEKU(indeks_sekv_dat)
POSTAVI status_azuriranja <- 1
VRATI status_azuriranja
KRAJ PROCESA azuriraj_slog
```

Logičko brisanje sloga

Logičko brisanje sloga - Koraci

1. Pozivanje procesa traženja:
 - a. Pozvati *trazenje_sloga* koristeći identifikator (ključ) sloga.
2. Provera rezultata traženja:
 - a. Ako je rezultat funkcije traženja neuspešan završavamo proces modifikacije
 - b. Ako je rezultat funkcije traženja uspešan nastavljamo sa procesom modifikacije
3. Logičko brisanje u primarnoj zoni
 - a. Pozicionirati se na tačnu adresu bloka unutar primarne zone
 - b. Promeniti vrednost statusa sloga sa 0 (aktivan) na 1 (obrisan), čime slog postaje “nevidljiv”, ali fizički ostaje u datoteci
 - c. Formirati novi slog kombinovanjem postojećeg ključa sa modifikovanim podacima sloga
4. Logičko brisanje u zoni prekoračenja
 - a. Preskočiti primarnu zonu i pozicionirati se na zonu prekoračenja
 - b. Promeniti vrednost statusa sloga sa 0 (aktivan) na 1 (obrisan), čime slog postaje “nevidljiv”, ali fizički ostaje u datoteci
 - c. Formirati novi slog kombinovanjem postojećeg ključa sa modifikovanim podacima sloga

Logičko brisanje sloga - Koraci

5. Na osnovu rezultata pisanja bloka, postaviti konačni rezultat operacije na 0 (uspešno) ili 1 (neuspešno)

```

PROCES logičko_brisanje(U(naziv_dat, novi_slog), I(status_azuriranja), UI(blok_primarni, slog_prek))
POČETAK PROCESA azuriraj_slog
    OTVORI_DATOTEKU(U(indeks_sekv_dat, režim_otvaranja), I(status_otvaranja_datoteke))
    POZOVI PROCES rezultat_trazenja <- trazenje_sloga(U(naziv_dat, novi_slog.id), I(), UI())
    AKO JE rezultat_trazenja.adresa_bloka = -1 TADA
        ZATVORI_DATOTEKU(indeks_sekv_dat)
        POSTAVI status_azuriranja <- 0
        VRATI status_azuriranja
    KRAJ AKO
    AKO JE rezultat_trazenja.zona = 0 TADA
        POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)
        POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), rezultat_trazenja.adresa_bloka *
veličina(BLOK_PRIM))
        ČITAJ_BLOK(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_čitanja),
UI(blok_primarni))
        POSTAVI blok_primarni.slog[rezultat_trazenja.adresa_sloga].obrisan <- 1
        POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa)
        POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), rezultat_trazenja.adresa_bloka *
veličina(BLOK_PRIM))
        PIŠI_BLOK(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_pisanja),
UI(blok_primarni))

```

NASTAVAK NA SLEDEĆEM SLAJDU

NASTAVAK

INAČE

```
POMERI_POKAZIVAČ(U(indeks_sekv_dat, POČETAK_DATOTEKE), velicina_zone_indeksa +
velicina_primarne_zone)
POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), velicina(ZAGLAVLJE_ZONE_PREKORACENJA) +
rezultat_trazenja.adresa_bloka * velicina(SLOG_ZONE_PREKORACENJA))
ČITAJ_SLOG(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_čitanja),
UI(slog_zone_prekoracenja))
POSTAVI slog_zone_prekoracenja.slog.obrisan <- 1
POMERI_POKAZIVAČ(U(indeks_sekv_dat, TEKUĆA_POZICIJA), -velicina(SLOG_ZONE_PREKORACENJA))
PIŠI_SLOG(U(indeks_sekv_dat, rezultat_trazenja.adresa_bloka), I(status_pisanja),
UI(slog_zone_prekoracenja))
KRAJ AKO
ZATVORI_DATOTEKU(indeks_sekv_dat)
POSTAVI status_azuriranja <- 1
VRATI status_azuriranja
KRAJ PROCESA logičko_brisanje
```

Kraj!

Hvala na pažnji!