SEKVENCIJALNA DATOTEKA

Vežbe

Sadržaj

Zadatak

Zadatak

- Napisati C program koji će omogućiti rad sa podacima o evidenciji pozajmica knjiga u gradskoj biblioteci. Pozajmice se evidentiraju u okviru blokirane sekvencijalne datoteke sa faktorom blokiranja f=4. Za svaku pozajmicu beleži se:
  - Evidencioni broj pozajmice (jedinstven broj do 10 cifara)
  - Broj članske karte (do 6 cifara)
  - ISBN knjige (tačno 13 karaktera)
  - Naziv knjige (tekstualni naziv knjige, do 15 karaktera, razmaci odvojeni donjom crtom)
  - Datum i vreme pozajmice (u formatu DD/MM/GGGG_HH:mm)
  - Status pozajmice (aktivno, vraćeno u oznaci ACTIVE, RETURNED)

Zadatak

- Omogućiti:
  a. odabir datoteke,
  b. formiranje datoteke,
  c. unos novog sloga,
  d. ispis svih slogova,
  e. traženje u okviru datoteke,
  f. modifikacija sloga (promena naziva knjige i/ili statusa pozajmice),
  g. fizičko brisanje sloga i
  h. logičko brisanje.

Fizička struktura sekvencijalne datoteke

Primer fizičke strukture sekvencijalne datoteke

- Fizička strukture jedne sekvencijalne datoteke od 12 slogova sa faktorom blokiranja f = 3
- Slogovi su uređeni saglasno rastućim vrednostima ključa

A_1		$p(S_1)$		$p(S_2)$		$p(S_3)$
	3		6		13	
A_2		$p(S_4)$		$p(S_5)$		$p(S_6)$
	19		25		29	
A_3		$p(S_7)$		$p(S_8)$		$p(S_9)$
	49		55		64	
A_4		$p(S_{10})$		$p(S_{11})$		
	68		*			

Formiranje blokirane sekvencijalne datoteke

Formiranje blokirane sekvencijalne datoteke

Koraci formiranja sekvencijalne datoteke:

1. Definisanje strukture
2. Kreiranje sekvencijalne datoteke
3. Otvaranje sekvencijalne datoteke
4. Pokretanje procesa preuzimanja podataka
5. Dok postoje podaci za unos:
    a. Preuzimanje podataka
    b. Validacija podataka
    c. Formiranje sloga
    d. Upis sloga
6. Zatvaranje sekvencijalne datoteke

Napomena: dodavanje slogova je objašnjeno u delu Upis novog sloga

Formiranje blokirane sekvencijalne datoteke - Definisanje strukture

Definisanje strukture sekvencijalne datoteke obuhvata definisanje:

- strukture sloga
  - naziv i tip svih polja sloga
  - ključ sloga
  - polje za logičko brisanje sloga
- faktora blokiranja
- oznake kraja datoteke

Formiranje sekvencijalne datoteke - Kreiranje sekvencijalne datoteke

```
PROCES formiranje_sekvencijalne_datoteke(U(naziv_sekvencijalne_datoteke), I(status), UI() )
POČETAK PROCESA formiranje_sekvencijalne_datoteke

    OTVORI_DATOTEKU( U(naziv_sekvencijalne_datoteke), I(status_otvaranja_datoteke), UI() )

    AKO JE status_otvaranja_datoteke = 0
        INICIJALIZUJ_BLOK(I(prviBlok))
        POSTAVI prviBlok.slogovi[0].ključ <- OZNAKA_KRAJA_DATOTEKE

        UPISI_BLOK( U(naziv_sekvencijalne_datoteke, prviBlok, 0 ), I(status_upisa_u_datoteku), UI() ),
        ZATVORI_DATOTEKU( U(naziv_sekvencijalne_datoteke ), I(status_zatvaranja_datoteke), UI( ))
        POSTAVI status <- 0

    INAČE

        POSTAVI status <- 2

    KRAJ AKO

KRAJ PROCESA formiranje_sekvencijalne_datoteke
```

Ispis sadržaja

Ispis sadržaja - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
3. Učitavanje bloka
4. Provera uspešnosti čitanja i provera statusa čitanja bloka:
   a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
   b. Ako je uspešno, nastavljanje procesa.
5. Prolazak kroz sve slogove u trenutnom bloku
   a. Ako je slog validan (nije logički obrisan), prikazati njegov sadržaj.
6. Prelazak na sledeći blok i ponavljanje koraka 3-6
   a. Završetak kada se dostigne kraj datoteke

Ispis sadržaja

fseek(fajl, 0, SEEK_SET);

```
PROCES ispis_sadrzaja(U(sekvencijalna_datoteka), I(status))
POČETAK PROCESA ispis_sadrzaja
    AKO JE sekvencijalna_datoteka == NULL
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))  (* pripremi prostor za blok podataka *)
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI ispis_blokova DOK JE status_čitanja == 0
            POSTAVI i <- 0
            RADI ispis_slogova DOK JE i < VELIČINA_BLOKA
               AKO JE blok.slogovi[i].ključ != OZNAKA_KRAJA_DATOTEKE I blok.slogovi[i].obrisan != 1
                     ISPISI blok.slogovi[i]
                KRAJ AKO
                POSTAVI i <- i + 1
            KRAJ RADI ispis_slogova
            POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
            ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        KRAJ RADI ispis_blokova
        POSTAVI status <- 0
    KRAJ AKO
KRAJ PROCESA ispis_sadrzaja
```

Traženje sloga

Traženje sloga

- Traženje se u sekvencijalnoj datoteci vrši primenom metode linearnog traženja
- Traženje počinje od početka datoteke pristupanjem sukcesivno memorijskim blokovima
- Traženje se vrši sve dok se traženi slog ne pronađe, ili dok argument traženja ne postane manji od od vrednosti ključa sloga (ili veći), ili dok se ne dođe do kraja datoteke
- Dva ishoda traženja:
  - *Uspešno* - pronađen slog sa vrednošću ključa jednakom argumentu traženja
  - *Neuspešno* - ako se naiđe na slog sa vrednošću ključa većom od argumenta traženja ili se dođe do kraja datoteke

Traženje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Inicijalizacija brojača blokova kako bi traženje počelo od prvog bloka
3. Učitavanje bloka
4. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
5. Prolazak kroz sve slogove u trenutnom bloku
    a. Ako je slog validan (nije logički obrisan) i ne označava kraj datoteke, proveriti da li se njegov ključ poklapa sa traženim ključem.
        i. Ako se ključ poklapa, ispisati podatke o slogu i završiti proces.
        ii. Ako je vrednost ključa veća (manja) od traženog ključa prekinuti proces i obavestiti korisnika da slog nije pronađen
        iii. Ako je vrednost ključa manja od traženog ključa, nastaviti sa sledećim slogom.

Traženje sloga - koraci

6. Prelazak na sledeći blok i ponavljanje koraka 3-6
    a. Ako se dostigne kraj datoteke (i traženi slog nije pronađen), obavestiti korisnika da slog nije pronađen.

Traženje sloga

```
PROCES nadji_slog(U(sekvencijalna_datoteka, ključ_sloga), I(pronadjeni_slog), UI())
POČETAK PROCESA nadji_slog
    AKO JE sekvencijalna_datoteka == NULL
        POSTAVI pronadjen_blok <- NULL
    INAČE
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok)) (* pripremi prostor za blok podataka *)
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
            POSTAVI i <- 0
            RADI provera_slogova
                NAREDNI SLAJD
            KRAJ RADI provera_slogova
            POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
            ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        KRAJ RADI provera_trenutnog_bloka
        POSTAVI pronadjeni_slog<- NULL
    KRAJ AKO
KRAJ PROCESA nadji_slog
```

Traženje sloga

- Provera slogova:

```
RADI provera_slogova DOK JE i < VELIČINA_BLOKA
    AKO JE blok.slogovi[i].ključ == ključ_sloga I blok.slogovi[i].obrisan == 0
        POSTAVI pronadjen_slog <- blok.slogovi[i]
        IZLAZ IZ PROCESA
    KRAJ AKO
    AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE ILI
           blok.slogovi[i].ključ > ključ_sloga
        POSTAVI status <- 1
        IZLAZ IZ PROCESA
    KRAJ AKO
    POSTAVI i <- i + 1
KRAJ RADI provera_slogova
```

Upis novog sloga

Upis novog sloga

- Upisu novog sloga prethodi neuspešno traženje
  - Da bi se upisao novi slog, potrebno je prvo proveriti da li se slog sa datim ključem već nalazi u datoteci
- Novi slog se upisuje na lokaciju sloga sa prvom većom vrednošću ključa
  - Svi slogovi sa vrednošću ključa većom od vrednosti ključa novog sloga se pomeraju za jedno mesto u desno
  - Ako ne postoji slog sa vrednošću ključa većom od vrednosti ključa novog sloga, novi slog se upisuje na kraj datoteke, a slog koji označava kraj datoteke se pomera jedno mesto u desno

Upis novog sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za upis
2. Inicijalizacija novog sloga koji će biti upisan
3. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
4. Učitavanje bloka
5. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.

Upis novog sloga - koraci

6. Prolazak kroz sve slogove u trenutnom bloku
    a. Ako vrednost ključa trenutnog sloga predstavlja kraj datoteke i trenutni blok nije popunjen
        i. Upis novog sloga na mesto sloga sa oznakom kraja datoteke i pomeranje oznake kraja datoteke za jedno mesto u desno
    b. Ako vrednost ključa trenutnog sloga predstavlja kraj datoteke i trenutni blok je popunjen
        i. Upis novog sloga na mesto trenutnog sloga i inicijalizacija i upis poslednjeg bloka sa oznakom kraja datoteke
    c. Ako vrednost ključa trenutnog sloga odgovara ključu novog sloga a trenutni slog je logički obrisan, tada na mesto logički obrisanog sloga staviti novi slog
    d. Ako je vrednost ključa trenutnog sloga veća od vrednosti ključa novog sloga:
        i. Na mesto trenutnog sloga se upisuje novi slog, a novi slog postaje trenutni slog
            1. Ako je trenutni slog bio poslednji slog u bloku, blok se upisuje u datoteku

Upis novog sloga

```
PROCES dodaj_slog(U(sekvencijalna_datoteka, novi_slog), I(status))
POČETAK PROCESA dodaj_slog
    AKO JE sekvencijalna_datoteka == NULL
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok)) (* pripremi prostor za blok podataka *)
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI ucitavanje_blokova DOK JE status_čitanja == 0
            POSTAVI i <- 0
            RADI pronalazak_pozicije DOK JE i < VELIČINA_BLOKA
                NAREDNI SLAJD
            KRAJ pronalazak_pozicije
            POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
            ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        KRAJ AKO
    KRAJ AKO
KRAJ PROCESA dodaj_slog
```

Upis novog sloga

```
RADI pronalazak_pozicije DOK JE i < VELIČINA_BLOKA
    AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE
        POSTAVI blok.slogovi[i] <- novi_slog
        AKO JE i != VELIČINA_BLOKA - 1
            blok.slogovi[i+1].ključ <- OZNAKA_KRAJA_DATOTEKE
            POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
            UPIŠI_BLOK(U(sekvencijalna_datoteka), UI(blok))
            IZLAZ IZ PROCESA
        INAČE
            POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
            UPIŠI_BLOK(U(sekvencijalna_datoteka), UI(blok))
            INICIJALIZUJ_BLOK(I(novi_blok))
            POSTAVI novi_blok.slogovi[0].ključ <- OZNAKA_KRAJA_DATOTEKE
            UPIŠI_BLOK(U(sekvencijalna_datoteka), UI(novi_blok))
            IZLAZ IZ PROCESA
        KRAJ AKO
    KRAJ AKO
    NAREDNI SLAJD
    POSTAVI i <- i + 1
KRAJ pronalazak_pozicije
```

Upis novog sloga

```
AKO JE blok.slogovi[i].ključ == ključ_sloga
    AKO JE blok.slogovi[i].obrisan = 1 (* Postoji ali je logički obrisan *)
        POSTAVI blok.slogovi[i] <- pronadjen_slog
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
        UPIŠI_BLOK(U(sekvencijalna_datoteka), UI(blok))
        IZLAZ IZ PROCESA
    INAČE
        POSTAVI status <- 2  (* Greška: slog već postoji *)
KRAJ AKO
AKO JE blok.slogovi[i].ključ > ključ_sloga
    INICIJALIZUJ_SLOG(I(privremeni_slog))
    POSTAVI privremeni_slog <- blok.slogovi[i]
    POSTAVI blok.slogovi[i] <- novi_slog
    POSTAVI novi_slog <- privremeni_slog (* novi slog se dodaje u narednoj iteraciji *)
    AKO JE i = VELIČINA_BLOKA - 1 (* ako je to poslednji slog u bloku, upiši ceo blok *)
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, trenutna_pozicija), -VELIČINA_BLOKA)
        UPIŠI_BLOK(U(sekvencijalna_datoteka), UI(blok))
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, trenutna_pozicija), 0)
    KRAJ AKO
KRAJ AKO
```

Logičko brisanje sloga

Logičko brisanje sloga

- Logičko brisanje sloga datoteke podrazumeva izmenu statusnog polja
- Potrebno je proći kroz čitavu datoteku i proveriti da li se ključ nekog sloga poklapa sa ključem sloga za brisanje
- Ako je slog pronađen, potrebno je izmeniti statusno polje i izmenjeni slog upisati nazad u datoteku

Logičko brisanje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Provera da li slog postoji
3. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
4. Učitavanje bloka
5. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
6. Prolazak kroz sve slogove u trenutnom bloku
    a. Proveriti da li se njegov ključ poklapa sa ključem sloga za logičko brisanje
        i. Ako se ključ poklapa:
            1. izmeniti polje za logičko brisanje sloga
            2. i upisati izmenjeni slog u datoteku
            3. prekid procesa
        ii. Ako je vrednost ključa trenutnog sloga veća od ključa traženog sloga , prekinuti proces.

Logičko brisanje sloga - koraci

6. ... Prolazak kroz sve slogove u trenutnom bloku
    iii. Ako je vrednost ključa trenutnog sloga manja od ključa traženog sloga, nastaviti sa sledećim slogom.
7. Prelazak na sledeći blok i ponavljanje koraka 3-6
    a. Ako se dostigne kraj datoteke (i traženi slog nije pronađen), obavestiti korisnika da slog nije pronađen.

Logičko brisanje sloga

```
PROCES logicko_brisanje_sloga(U(sekvencijalna_datoteka, ključ_sloga), I(status), UI())
POČETAK PROCESA logicko_brisanje_sloga
    AKO JE sekvencijalna_datoteka == NULL
        POSTAVI status <- 1
    INAČE
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka ), I(status_čitanja), UI(blok))
        (* učitavanje po blokovima *)
        RADI učitavanje_sadržaja_datoteke DOK JE status_čitanja == 0
                    POSTAVI i <- 0
                    RADI provera_slogova DOK JE i < VELIČINA_BLOKA
                        NAREDNI SLAJD
                    KRAJ RADI provera_slogova
                POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
                ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka ), I(status_čitanja), UI(blok))
        KRAJ RADI učitavanje_sadržaja_datoteke
    KRAJ AKO
KRAJ PROCESA logicko_brisanje_sloga
```

Logičko brisanje sloga - *provera_slogova*

```
RADI provera_slogova DOK JE i < VELIČINA_BLOKA

    (* dosli smo do kraja datoteke ili slog ne postoji *)
    AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE ILI blok.slogovi[i].ključ > ključ_sloga
        POSTAVI status <- 2
    KRAJ AKO

    (* dosli smo do sloga koji treba logicki obrisati *)
    AKO JE blok.slogovi[i].ključ == ključ_sloga TADA
        POSTAVI blok.slogovi[i].obrisan <- 1;
        POSTAVI redni_broj_bloka <- redni_broj_bloka - 1
        UPIŠI_BLOK(U(sekvencijalna_datoteka, blok, redni_broj_bloka), I(status_upisa_u_datoteku), UI(
))
        POSTAVI status <- 0
    KRAJ AKO

    POSTAVI i <- i + 1
KRAJ RADI provera_slogova
```

Fizičko brisanje sloga

Fizičko brisanje sloga

- Fizičko brisanje sloga u datoteci zahteva njegovo prethodno nalaženje
- Nakon uspešnog nalaženja sloga, sve sledeće slogove u datoteci je potrebno pomeriti za jedno mesto unazad
- Počevši od bloka u kome je nađen slog za brisanje, svi sledeći blokovi se ažuriraju
- Zbog popunjavanja poslednjeg sloga u bloku, za izmenu svakog bloka potrebno je učitavanje i narednog bloka
- Da bi se pomeranje slogova datoteke nastavilo, vrednost sloga koji se briše se ažurira ključem prvog sloga u narednom bloku
- Ako je poslednji slog datoteke obrisan, potrebno je uraditi skraćivanje datoteke

Fizičko brisanje sloga - koraci

1. Provera da li je datoteka otvorena ili dostupna za čitanje
2. Provera da li slog postoji
3. Inicijalizacija brojača blokova kako bi čitanje počelo od prvog bloka
4. Učitavanje bloka
5. Provera uspešnosti čitanja i provera statusa čitanja bloka:
    a. Ako je čitanje neuspešno (npr. kraj datoteke ili greška) - prekidanje procesa
    b. Ako je uspešno, nastavljanje procesa.
6. Prolazak kroz sve slogove u trenutnom bloku
    a. Proveriti da li se njegov ključ poklapa sa ključem sloga za fizičko brisanje
        i. Ako se ključ poklapa potrebno je pomeriti sve slogove do kraja datoteke za jedno mesto
        ii. Po potrebi obrisati poslednji blok ukoliko je ostao prazan

Fizičko brisanje sloga

```
PROCES fizičko_brisanje_sloga(U(sekvencijalna_datoteka, ključ_sloga), I(status), UI())
POČETAK PROCESA nadji_slog
    nadji_slog(U(sekvencijalna_datoteka, ključ_sloga), I(status_traženja_sloga)) TADA
    AKO JE pronadjeni_slog == NULL:
        POSTAVI status <- 2
    INAČE
        POMERI_POKAZIVAČ(U(sekvencijalna_datoteka, početak datoteke), 0)
        INICIJALIZUJ_BLOK(I(blok))
        INICIJALIZUJ_BLOK(I(naredni_blok))
        POSTAVI trenutni_ključ_sloga <- ključ_sloga
        POSTAVI redni_broj_bloka <- 0
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
        RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
            NAREDNI SLAJD
        KRAJ RADI provera_trenutnog_bloka
        POSTAVI redni_broj_bloka <- redni_broj_bloka + 1
        ČITAJ_BLOK(U(sekvencijalna_datoteka, redni_broj_bloka), I(status_čitanja), UI(blok))
    KRAJ AKO
KRAJ PROCESA fizičko_brisanje_sloga
```

Fizičko brisanje sloga

```
RADI provera_trenutnog_bloka DOK JE status_čitanja == 0
    POSTAVI i <- 0
    RADI provera_slogova DOK JE i < VELIČINA_BLOKA
        (* došli smo do poslednjeg sloga datoteke, svi slogovi su pomereni * )
        AKO JE blok.slogovi[i].ključ == OZNAKA_KRAJA_DATOTEKE TADA
            AKO JE i == 0
            (* nakon pomeranja, oznaka kraja datoteke je prvi slog u poslednjem bloku * )
                SKRATI_DATOTEKU(I(sekvencijalna_datoteka, redni_broj_bloka - 1))
            KRAJ AKO
            POSTAVI status <- 0
        KRAJ AKO
        AKO JE blok.slogovi[i].ključ == trenutni_ključ_sloga_za_brisanje TADA
            reorganizuj_blokove(I(sekvencijalna_datoteka, blok, naredni_blok),
                                IU(trenutni_ključ_sloga, ključ_sloga, redni_broj_bloka))
        KRAJ AKO
        i <- i + 1
    KRAJ RADI provera_slogova
KRAJ RADI provera_trenutnog_bloka
```

Fizičko brisanje sloga

```
PROCES reorganizuj_blokove(I(datoteka, blok, naredni_blok), IU(trenutni_ključ, ključ, rbr_bloka))
POČETAK PROCESA reorganizuj_blokove
    AKO JE blok.slogovi[i].kljuc == ključ I blok.slogovi[i].obrisan == 1 TADA
        KRAJ reorganizuj_blokove (* slog koji korisnik želi da obriše je već logički obrisan*)
    KRAJ AKO
    POSTAVI j <- i + 1
    RADI pomeranje_slogova DOK JE j < VELIČINA_BLOKA
        POSTAVI blok.slogovi[j-1] <- blok.slogovi[j]
        j <- j + 1
    KRAJ pomeranje_slogova
    ČITAJ_BLOK(U(datoteka, rbr_bloka + 1), I(status_čitanja_narednog_bloka), UI(naredni_blok))
    AKO JE status_čitanja_narednog_bloka == 0  TADA (* postoje blokovi posle trenutnog *)
        POSTAVI rbr_bloka <- rbr_bloka - 1
        POSTAVI blok.slogovi[VELIČINA_BLOKA -1] <- naredniBlok.slogovi[0]
        POSTAVI trenutni_ključ <- naredniBlok.slogovi[0].ključ
    KRAJ AKO
    POSTAVI rbr_bloka <- rbr_bloka - 1
    UPIŠI_BLOK(U(datoteka, blok, rbr_bloka), I(status_upisa_u_datoteku), UI( ))
    AKO JE status_čitanja_narednog_bloka == 0 TADA
        POSTAVI status <-0
    KRAJ AKO
KRAJ reorganizuj_blokove
```

Ažuriranje sloga

Ažuriranje sloga

- Isti postupak kao kod logičkog brisanja
- Koraci ažuriranja sloga:
  1. Uspešno traženje sloga
  2. Ažuriranje vrednosti sloga
  3. Upis bloka u kom se nalazi slog sa ažuriranim vrednostima

Kraj!

Hvala na pažnji!