

RASUTA DATOTEKA

Vežbe

Sadržaj

- Zadatak
- Fizička struktura rasute datoteke
- Formiranje rasute datoteke sa linearnim traženjem
- Traženje sloga
- Upis novog sloga
- Modifikacija sloga
- Logičko brisanje sloga
- Fizičko brisanje sloga
- Formiranje rasute datoteke sa linearnim traženjem u dva prolaza

Zadatak

Zadatak

- Napisati C program koji će omogućiti rad sa podacima o evidentiranim dolascima na posao i odlascima sa posla radnika zaposlenih u Naučno-tehnološkom parku u Novom Sadu. Evidencija se vrši u statičkoj rasutoj datoteci sa linearnim traženjem lokacija za smeštaj prekoračilaca. Za svaku evidenciju beleži se:
 - IDU - identifikaciona oznaka ulaska [do 7 cifara]
 - IDR - identifikacioni broj radnika [13 karaktera]
 - OS - oznaka sektora u kome se čekirao [3 karaktera]
 - DVD - datum i vreme dolaska na posao [DD-MM-YYYY HH:MM:SS]
 - DVO - datum i vreme odlaska sa posla [DD-MM-YYYY HH:MM:SS]
 - BRS - broj radnih sati u toku tekućeg dana
 - AKTIVAN - oznaka da li je slog logički obrisan [1 karakter: 'A' - aktivan, 'O' - obrisan]

Zadatak

- Omogućiti:
 - a. Formiranje datoteke na osnovu slogova iz ulazne serijske datoteke u dva prolaza
 - b. Unos novog sloga
 - c. Modifikaciju sloga
 - d. Logičko brisanje sloga
 - e. Traženje po identifikacionom broju radnika
 - f. Ispis svih slogova

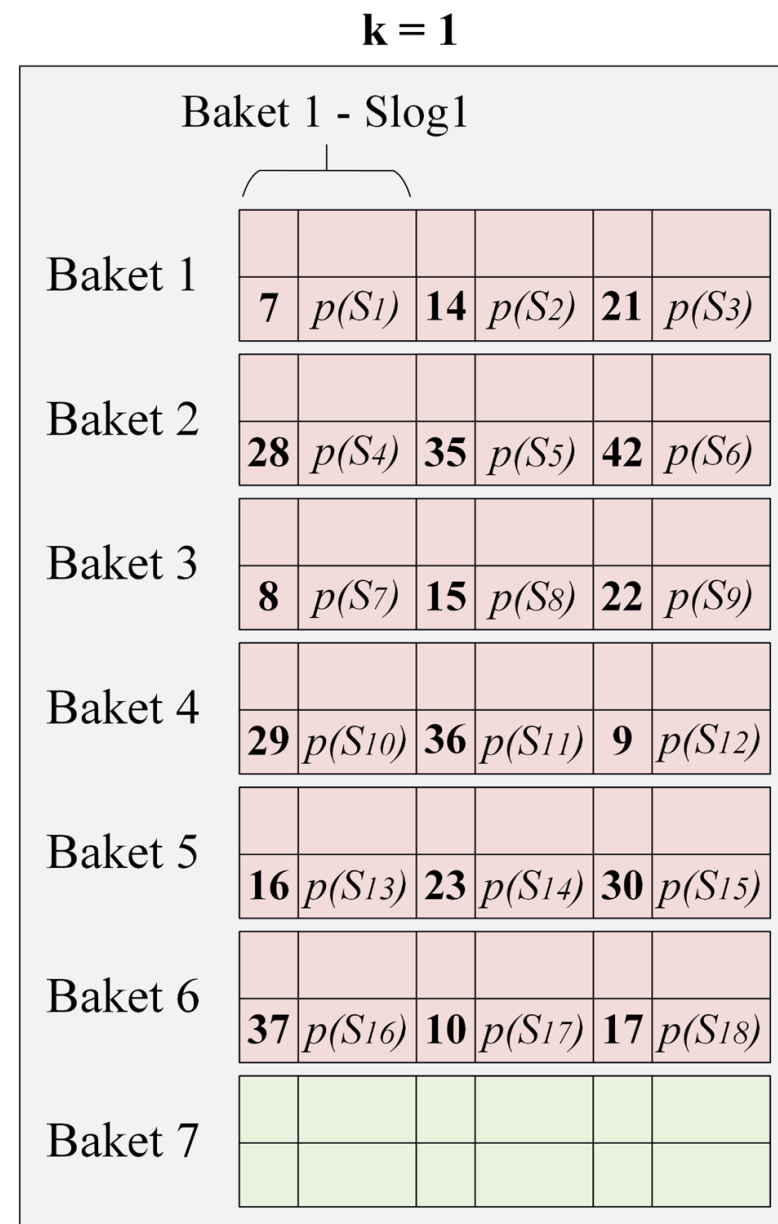
Zadatak

- Prilikom implementacije, ispoštovati sledeća ograničenja:
 - faktor baketiranja: $b = 3$
 - broj baketa: $B = 4$
 - Koristiti linearno traženje sa fiksnim korakom $k = 1$
 - za probabilističku transformaciju identifikatora sloga u adresu baketa koristiti metodu ostatka pri deljenju
 - prilikom rada sa datotekom, dozvoliti preuzimanje i upis isključivo celih baketa.
- Za vežbu dodatno implementirati:
 - ako se dosegne efekat nagomilavanja (više od 5 uzastopnih zauzetih lokacija), povećati korak na $k = 3$

Fizička struktura rasute datoteke

Fizička struktura rasute datoteke - **Primer 1**

- Za početni ulazni skup od $N = 18$ slogova, sa vrednostima ključa [7, 14, 21, 28, 35, 42, 8, 15, 22, 29, 36, 9, 16, 23, 30, 37, 10, 17], potrebno je rasporediti slogove u statičku rasutu datoteku:
- Broj baketa: $B = 7$
- Faktor baketiranja: $b = 3$
- Linearni korak: $k = 1$
- Transformacija za dobijanje adrese memorijske lokacije:
 - Matični baket: $h(k_i) = 1 + k(S_i) \pmod{B}$ *
 - Ukoliko je matični baket popunjen - preći na naredni baket (u skladu sa korakom k)



Slika 1. Prikaz fizičke strukture rasute datoteke za primer 1

Fizička struktura rasute datoteke - Primer 2

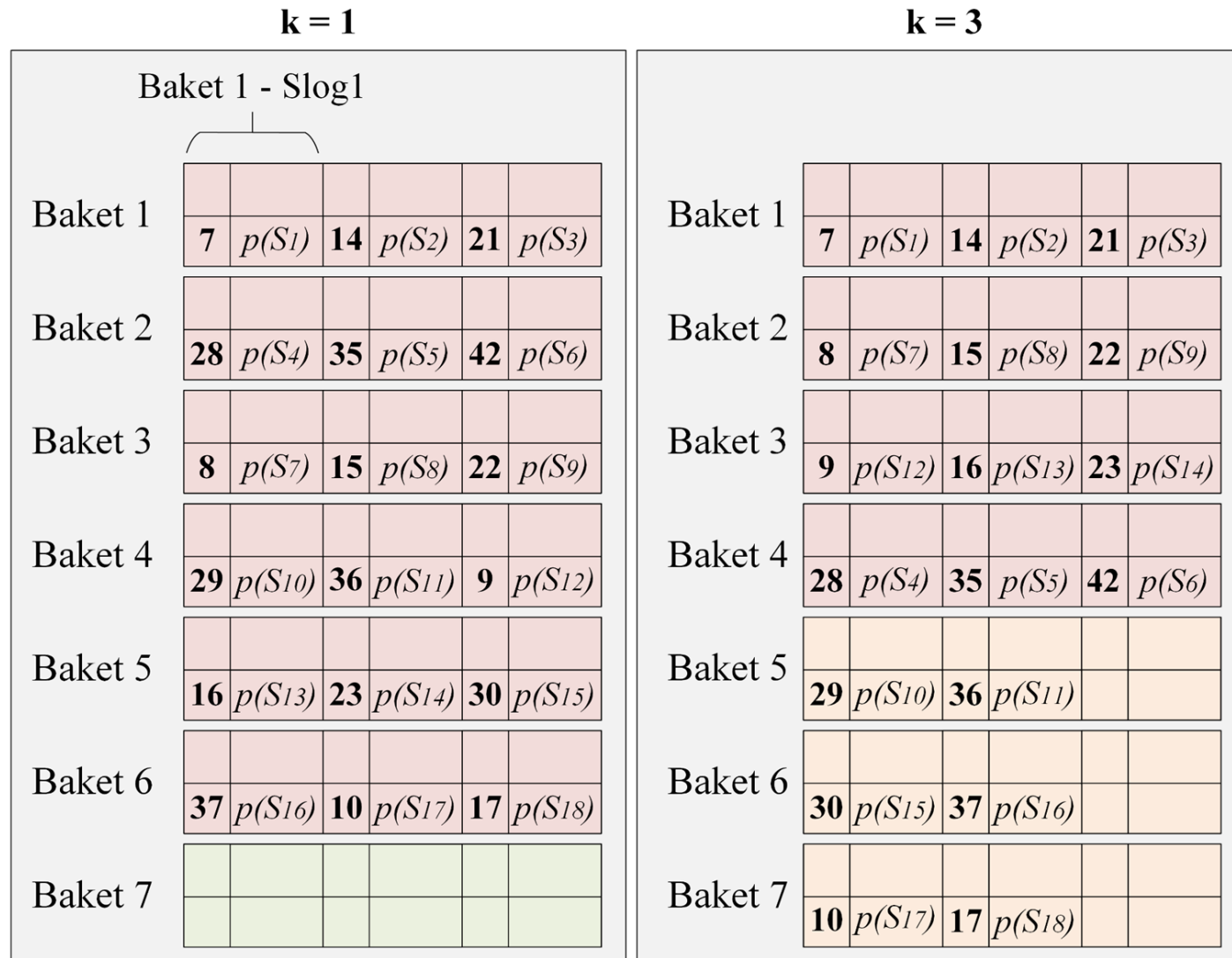
- Za početni ulazni skup od $N = 18$ slogova, sa vrednostima ključa [7, 14, 21, 28, 35, 42, 8, 15, 22, 29, 36, 9, 16, 23, 30, 37, 10, 17], potrebno je rasporediti slogove u statičku rasutu datoteku:
- Broj baketa: $B = 7$
- Faktor baketiranja: $b = 3$
- Linearni korak: $k = 3$
- Transformacija za dobijanje adrese memorijske lokacije:
 - $h(k_{in}) = (h(k_j) + 3*j) \pmod{B}$
 - j - redni broj pokušaja pristupa narednog baketu pri pojavi kolizije

$k = 3$

Baket 1						
	7	$p(S_1)$	14	$p(S_2)$	21	$p(S_3)$
Baket 2						
	8	$p(S_7)$	15	$p(S_8)$	22	$p(S_9)$
Baket 3						
	9	$p(S_{12})$	16	$p(S_{13})$	23	$p(S_{14})$
Baket 4						
	28	$p(S_4)$	35	$p(S_5)$	42	$p(S_6)$
Baket 5						
	29	$p(S_{10})$	36	$p(S_{11})$		
Baket 6						
	30	$p(S_{15})$	37	$p(S_{16})$		
Baket 7						
	10	$p(S_{17})$	17	$p(S_{18})$		

Slika 2. Prikaz fizičke strukture rasute datoteke za primer 2

- Niz ključeva: [7, 14, 21, 28, 35, 42, 8, 15, 22, 29, 36, 9, 16, 23, 30, 37, 10, 17]
- $B = 7$, $b = 3$, $q = 0.86$



Slika 3. Poređenje fizičkih struktura rasute datoteke za primer 1 i primer 2

Formiranje rasute datoteke u jednom prolazu sa linearnim traženjem

Formiranje rasute datoteke sa linearnim traženjem - Koraci

1. Definisavanje strukture
2. Otvaranje serijske datoteke
3. Otvaranje rasute datoteke
4. Kreiranje prazne strukture rasute datoteke, sa potrebnim brojem baketa i lokacija
5. Čitanje slogova iz serijske datoteke, sve dok se ne pročita ceo sadržaj serijske datoteke
6. Poziv funkcije traženja za svaki slog iz trenutnog bloka u serijskoj datoteci
 - a. Za svaki slog iz trenutnog bloka, poziva se funkcija koja traži odgovarajuću poziciju u rasutoj datoteci koristeći hash funkciju i metodu linearnog traženja
7. Na osnovu indikatora koje vraća funkcija traženja
 - a. Ako slog ne postoji i ima slobodnog mesta, slog se upisuje na pronađenu poziciju u rasutoj datoteci
 - b. Ako nema slobodnog mesta, proces se prekida
 - c. Ako slog sa istim ključem već postoji, prelazi se na naredni slog

Formiranje rasute datoteke sa linearnim traženjem - Koraci

8. Nakon obrade svih slogova iz trenutnog bloka, čita se naredni blok iz serijske datoteke
 - a. Ponoviti postupak od koraka 6
9. Zatvaranje rasute datoteke
10. Zatvaranje serijske datoteke

Formiranje rasute datoteke sa linearnim traženjem - Definisane strukture

- Definisane strukture rasute datoteke obuhvata **određivanje**:
 - Strukture sloga
 - Broj baketa u rasutoj datoteci
 - Faktora baketiranja
 - Korak za linearnu pretragu

Formiranje rasute datoteke sa linearnim traženjem - Pseudo-kod

PROCES formiranje_ras_dat_sa_lin_tr(U(*serijska_dat*, *rasuta_dat*, *broj_slogova*, *faktor_popunjenosti*, *faktor_baketiranja*), I(), UI())

POČETAK PROCESA formiranje_ras_dat_sa_lin_tr

POSTAVI *broj_baketa* <- *broj_slogova* / (*faktor_popunjenosti* * *faktor_baketiranja*)

OTVORI_DATOTEKU(U(*serijska_dat*, *režim_otvaranja*), I(*status_otvaranja_datoteke*))

OTVORI_DATOTEKU(U(*rasuta_dat*, *režim_otvaranja*), I(*status_otvaranja_datoteke*))

POZOVI formiranje_prazne_datoteke(U(*rasuta_dat*, *broj_baketa*, *faktor_baketiranja*))

ČITAJ_SLOG(*serijska_dat*, *trenutni_slog_serijska*, *status_čitanja*)

RADI formiranje dok je *status_čitanja* = 0

POZOVI traženje_sloga(*rasuta_dat*, *kljuc_sloga*, *ind_usp_tr*, *ind_psl*,
(*baket_idx*, *faktor_popunjenosti*), *baket*)

AKO JE (*ind_usp_tr* == 1 I *ind_psl* == 0)

POSTAVI *slog_za_upis* <- *trenutni_slog_serijska*

PIŠI_BAKET(*rasuta_dat*, *redni_broj_baketa*, *status_pisanja*, *baket*)

ČITAJ_BLOK(*serijska_dat*, *blok*, *status_čitanja*)

NASTAVAK NA SLEDEĆEM SLAJDU

Formiranje rasute datoteke sa linearnim traženjem - Pseudo-kod

```
PROCES formiranje_ras_dat_sa_lin_tr(U(serijska_dat, rasuta_dat, broj_slogova, faktor_popunjenosti,  
faktor_baketiranja), I(), UI())
```

```
POČETAK PROCESA formiranje_ras_dat_sa_lin_tr
```

```
  NASTAVAK
```

```
    INAČE
```

```
      AKO JE ind_psl == 1 // nedostatak slobodnih lokacija
```

```
        POSTAVI status_čitanja <- 1
```

```
      INAČE // slog sa datom vrednošću ključa već postoji
```

```
        ČITAJ_SLOG(serijska_dat, trenutni_slog_serijska, status_čitanja)
```

```
      KRAJ AKO
```

```
    KRAJ AKO
```

```
  KRAJ RADI formiranje
```

```
  ZATVORI_DATOTEKU(rasuta_dat)
```

```
  ZATVORI_DATOTEKU(serijska_dat)
```

```
KRAJ PROCESA formiranje_ras_dat_sa_lin_tr
```

Traženje sloga

Traženje sloga - Koraci

1. Određivanje početne pozicije
 - a. Na osnovu ključa sloga, izračunati adresu matičnog baketa u kojem bi slog trebalo da se nalazi
2. Traženje u matičnom baketu
 - a. Učitati odgovarajući baket i pregledati slogove
 - b. Ako se traženi slog nalazi u matičnom baketu, traženje je uspešno
 - c. Ako matični baket ne sadrži traženi slog, a sadrži slobodnu lokaciju, traženje je neuspešno

Traženje sloga - Koraci

3. Prelazak na sledeći baket ako slog nije pronađen
 - a. Ako slog nije pronađen, prelaziti na sledeće bakete prema pravilima linearne metode (pomeramo se na sledeći baket koji je udaljen od trenutnog za korak k)
 - b. Kad se naiđe na slog sa traženom vrednošću ključa, traženje je uspešno
 - c. Ako se dođe do baketa sa slobodnom lokacijom ili se saglasno cirkularnom postupku određivanja adrese baketa dođe do matičnog baketa, traženje je neuspešno
4. Završetak traženja - vratiti rezultat pretrage:
 - a. status da slog nije ili jeste pronađen
 - b. status da li u datoteci ima slobodnih lokacija ili nema
 - c. lokaciju baketa i lokaciju sloga u baketu
 - d. baket

Traženje sloga - Pseudo-kod

```
PROCES trazenje_sloga(U(rasuta_dat, kljuc_sloga), I(ind_usp_tr, ind_psl, (idx_tr_baketa, idx_tr_sloga), baket) UI())
```

```
POČETAK PROCESA trazenje_sloga
```

```
  POZOVI baket_idx <- transformacija_kljuca(kljuc_sloga, idx_tr_baketa)
```

```
  POSTAVI init_idx <- baket_idx
```

```
  POSTAVI ind_usp_tr <- 99           // indikator uspešnosti traženja
```

```
  POSTAVI ind_psl <- 0              // indikator postojanja slobodnih lokacija
```

```
  RADI trazenje DOK JE ind_usp_tr == 99
```

```
    ČITAJ_BAKET(rasuta_dat, idx_tr_baketa, baket, status_čitanja)
```

```
    POSTAVI idx_tr_sloga <- 1
```

```
    RADI baket DOK JE (idx_tr_sloga <= f_baketiranja) I (ind_usp_tr == 99)
```

```
      AKO JE kljuc_sloga = kljuc_tr_sloga
```

```
        POSTAVI ind_usp_tr <- 0 // uspešno traženje
```

```
      INAČE
```

```
        AKO JE kljuc_tr_sloga = *
```

```
          POSTAVI ind_usp_tr <- 1 // neuspešno traženje
```

```
        INAČE
```

```
          POSTAVI idx_tr_sloga <- (idx_tr_sloga + 1)
```

NASTAVAK NA SLEDEĆEM SLAJDU

KRAJ PROCESA trazenje_sloga

Traženje sloga - Pseudo-kod

PROCES trazenje_sloga(*U(rasuta_dat, kljuc_sloga)*, *I(ind_usp_tr, ind_psl, (idx_tr_baketa, idx_tr_sloga)*, *baket*) *UI()*)

POČETAK PROCESA trazenje_sloga

NASTAVAK

KRAJ AKO

KRAJ AKO

KRAJ RADI *baket*

AKO JE *idx_tr_sloga > f_baketiranja* **TADA**

POSTAVI *idx_tren_baketa <- idx_tr_baketa MOD Q + 1*

AKO JE *idx_tren_baketa = init_idx*

POSTAVI *ind_usp_tr <- 1*

POSTAVI *ind_psl <- 1*

INAČE

KRAJ AKO

INAČE

KRAJ AKO

KRAJ RADI *trazenje*

KRAJ PROCESA trazenje_sloga

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 odgovarajuću poziciju u baketu (koristeći linearnu metodu ako je baket pun)
2. Provera rezultata traženja:
 - a. Ako je rezultat funkcije traženja neuspešan i i ima slobodnih mesta nastavljamo sa upisom (stavka 3.)
 - b. Ako je rezultat funkcije traženja uspešan ili nema slobodnih mesta postavljamo status na 1 (neuspešno) i završavamo proces dodavanja novog sloga
3. Nastavak upisa:
 - a. Pozivamo funkciju *upis_sloga*
 - b. Postavljamo status na 1 (uspešno dodavanje) i završavamo proces

Upis novog sloga - Pseudokod

```
PROCES dodavanje_sloga(U(rasuta_dat, kljuc_sloga, novi_slog), I(status), UI())
POČETAK PROCESA dodavanje_sloga
    // Pozivanje procesa traženja da se dobije pozicija za novi slog
    PROCES trazenje_sloga(rasuta_dat, kljuc_sloga, ind_usp_tr, ind_psl, (idx_tr_baketa, idx_tr_sloga),
baket)
    // Provera rezultata pretrage:
    // Ako je pretraga neuspešna i ima slobodnih lokacija
    AKO JE ind_usp_tr = 1 I ind_psl = 0
        POZOVI upis_sloga(baket, idx_tr_sloga, novi_slog)
        PIŠI_BAKET(rasuta_dat, baket_idx, status_pisanja, baket)
        POSTAVI status<- 0 // Dodavanje novog sloga uspešno
    INAČE
        POSTAVI status<- 1 // Dodavanje novog sloga neuspešno
    KRAJ AKO
KRAJ PROCESA dodavanje_sloga
```

Upis novog sloga - Pseudokod

```
PROCES upis_sloga(U(baket, idx_tr_sloga), I(), UI(novi_slog))
POČETAK PROCESA upis_sloga
    // Upisivanje novog sloga u odgovarajuću poziciju unutar baketa
    baket[idx_tr_sloga] <- novi_slog
KRAJ PROCESA upis_sloga
```

Modifikacija sloga

Modifikacija sloga - Koraci

1. Pozivanje procesa traženja:
 - a. Pozvati *trazenje_sloga* sa datim podacima i ključem slog-a
 - i. Ovaj proces će pokušati pronaći odgovarajuću poziciju u baketu (koristeći linearnu metodu ako je baket pun)
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. Inicijalizacija statusa operacije (0 = uspešno)
4. Kreiranje modifikovanog sloga
 - a. Izdvojiti postojeći ključ iz sloga koji se modifikuje
 - b. Izvršiti modifikaciju sloga
 - c. Formirati novi slog kombinovanjem postojećeg ključa sa modifikovanim podacima sloga

Modifikacija sloga - Koraci

5. Zamena sloga u baketu
 - a. Na određenoj poziciji unutar baketa, postojeći slog zameniti novim (modifikovanim) slogom
6. Upisati baket u datoteku na odgovarajuću poziciju
7. Provera uspešnosti operacije
 - a. Ako operacija nije bila uspešna (status $\neq 0$), postaviti status na 1

Modifikacija sloga - Pseudo-kod

```
PROCES modifikacija_ras_sa_lin_tr(U(rasuta_dat, slog), I(status_pisanja), UI())
POČETAK PROCESA modifikacija_ras_sa_lin_tr
    PROCES trazenje_sloga(rasuta_dat, kljuc_sloga, ind_ustp_tr, ind_psl, (idx_tr_baketa,
idx_tr_sloga), baket)
    AKO JE ind_ustp_tr = 0
    POSTAVI status <- 0 // inicijalno uspešan status
    POSTAVI novi_slog <- slog // kreiranje kopije sloga koji biva modifikovan
    POSTAVI baket[idx_tr_sloga] <- novi_slog
    PIŠI_BAKET(rasuta_dat, idx_tr_baketa, status_pisanja, baket)
    AKO JE status_pisanja != 0 TADA
        POSTAVI status <- 1
    KRAJ AKO
INAČE
    POSTAVI status <- 1
KRAJ AKO
KRAJ PROCESA modifikacija_ras_sa_lin_tr
```

Logičko brisanje sloga

Logičko brisanje sloga - Koraci

1. Pozivanje procesa traženja:
 - a. Pozvati *trazenje_sloga* sa datim podacima i ključem slog-a
 - i. Ovaj proces će pokušati pronaći odgovarajuću poziciju u baketu (koristeći linearnu metodu ako je baket pun)
2. Provera rezultata traženja
 - a. Ako slog nije pronađen, postaviti status operacije na 1 (neuspešno) i prekinuti izvršavanje procesa logičkog brisanja
 - b. Ako je slog pronađen, nastaviti sa izvršavanjem procesa logičkog brisanja

Logičko brisanje sloga - Koraci

3. Označiti sog kao logički obrisani
 - a. Promeniti vrednost statusa sloga sa 1 (aktivan) na 2 (logički obrisani), čime slog postaje "nevidljiv", ali fizički ostaje u datoteci
4. Pozvati funkciju za čuvanje baketa kako bi promena statusa logički obrisanih sloga ostala trajno zabeležena u datoteci
5. Na osnovu rezultata pisanja baketa, postaviti konačni rezultat operacije na 0 (uspešno) ili 1 (neuspešno)

Logičko brisanje sloga - Koraci

```
PROCES logicko_brisanje_sloga(U(rasuta_dat, kljuc_sloga), I(status), UI())
POČETAK PROCESA logicko_brisanje_sloga
    PROCES trazenje_sloga(rasuta_dat, kljuc_sloga, ind_ust_tr, ind_psl, (idx_tr_baketa, idx_tr_sloga),
baket)
    AKO JE ind_ust_tr != 0
        POSTAVI status <- -1 // slog nije pronađen - operacija neuspješna
    INAČE
        POSTAVI baket[idx_tr_sloga].status <- 2
    PIŠI_BAKET(rasuta_dat, idx_tr_baketa, status_pisanja, baket)
    AKO JE status_pisanja = 0
        POSTAVI status <- 0 // baket uspešno sačuvan - operacija uspešna
    INAČE
        POSTAVI status <- 1 // greška pri upisu baketa - operacija neuspješna
    KRAJ AKO
    KRAJ AKO
KRAJ PROCESA logicko_brisanje_sloga
```

Fizičko brisanje sloga

Fizičko brisanje sloga - Koraci

- U narednim koracima predstavimo brisanje sloga sa selektivnim pomeranjem u rasutoj datoteci sa lineranim traženjem, a kao skraćeni naziv koristićemo pojam „Fizičko brisanje sloga“.
1. Pronaći slog za brisanje u rasutoj datoteci
 2. Proveriti uspešnost pronalaska sloga
 - a. Ako traženi slog nije pronađen:
 - i. Postaviti status uspešnosti pronalaska sloga na -1
 - ii. Prekinuti proces traženja sloga
 - b. Ako je traženi slog pronađen:
 - i. Inicijalizovati vrednosti početnog indeksa sloga za brisanje, početnog baketa, indeksa baketa i koraka za traženje sloga.

Fizičko brisanje sloga - Koraci

3. Pomeranje slogova

a. Sve dok se ne dođe do kraja baketa:

i. Pomaknuti sve slogove za jedno mesto ulevo prema matičnom baketu

ii. Ako se baket isprazni, preći na sledeći baket i nastaviti pomeranje

4. Provera postojanja prekoračilaca u sledećim baketima

a. Ako postoji prekoračilac u narednom baketu, prebaciti ga u baket bliži matičnom baketu

b. Ako ne postoji prekoračilac, završiti pomeranje

5. Modifikacija podataka u datoteci

a. Upisati izmene u rasutu datoteku kako bi se sačuvala nova organizacija slogova

6. Završetak procesa.

a. Ako su svi slogovi uspešno pomereni, označiti brisanje kao uspešno

b. Ako nije bilo prekoračilaca za premeštaj, ostaviti prazno mesto u baketu

Fizičko brisanje sloga - Pseudo-kod

PROCES fizicko_brisanje_sloga(U(D), I(), UI(B, (r, q)))

POČETAK PROCESA fizicko_brisanje_sloga

```
    // 'D'   : deskriptor rasute datoteke
// 'B'   : trenutni baket
// 'r'   : indeks trenutnog baketa koji čitamo
// 'q'   : indeks sloga unutar baketa
// 'b'   : faktor baketiranja
// 'Q'   : ukupan broj baketa u datoteci (za rotaciju indeksa)
```

POSTAVI pomeranje <- 1

RADI selektivno_pomeranje **DOK JE** pomeranje = 1

// 1) Unutrašnje pomeranje unutar trenutnog baketa

RADI pomeranje_u_baketu **DOK JE** (q < b) **I** (k(B[q]) != *)

// pomeranje slogova unutar baketa (ulevo)

B[q] <- B[q + 1]

q <- q+ 1

KRAJ RADI pomeranje_u_baketu// Završetak pomeranje unutar baketa

NASTAVAK NA NAREDNOM SLAJDU

KRAJ PROCESA fizicko_brisanje_sloga

Fizičko brisanje sloga - Pseudo-kod

PROCES fizicko_brisanje_sloga(U(D), I(), UI(B, (r, q)))

POČETAK PROCESA fizicko_brisanje_sloga

NASTAVAK

POSTAVI Bp <- B // kreirati radnu kopiju početnog baketa

POSTAVI rp <- r // čuvamo originalni indeks sloga unutar baketa

AKO JE k(B[q]) =*

POSTAVI pomeranje <- 0 // baket nije skroz popunjen → ne vršiti prebacivanje

prekoračioca

INAČE // traženje sloga među prekoračiocima

POSTAVI nađen <- 0 // indikator pronađenog sloga - prekoračioca

RADI trazenje_u_prekoračiocima **DOK JE** (nađen = 0) **I** (pomeranje = 1)

AKO JE q=b //ako se stiglo do kraja prvobitnog baketa

POSTAVI r <- 1 + (r mod Q) //prelazimo na sledeći baket

ČITAJ_BAKET(D, r, B, status_čitanja) //u B je sada novi

baket

POSTAVI q <- 0 // započinjemo od prve pozicije unutar novog

baketa

KRAJ AKO

POSTAVI q <- q + 1

NASTAVAK NA NAREDNOM SLAJDU

Fizičko brisanje sloga - Pseudo-kod

PROCES fizicko_brisanje_sloga(U(rasuta_dat, kljuc_sloga), I(status))

POČETAK PROCESA fizicko_brisanje_sloga

NASTAVAK

AKO JE (k(B[q]) != *) **I** (r != rp) // Ako je r == rp, stigli smo opet do početnog

baketa

POZOVI provera_kandidata(k(B[q]), Q, r, rp, nađen)

INACE // ako nije pronađena nijedna pozicija → obeležiti tu poziciju kao praznu

POSTAVI pomeranje <- 0

KRAJ AKO

KRAJ RADI trazenje_u_prekoračiocima

AKO JE nađen = 1

POSTAVI Bp[b-1] <- B[q]

INACE

POSTAVI Bp[b-1] <- *

KRAJ AKO

KRAJ AKO

PIŠI_DIR(D, Bp, rp, stat)

KRAJ RADI selektivno_pomeranje

KRAJ PROCESA fizicko_brisanje_sloga

Fizičko brisanje sloga - Pseudo-kod

- Pomoćna funkcija za proveru da li se određeni slog može prebaciti u ciljni baket kako bi bio što bliže matičnom baketu

```
PROCES provera_kandidata(U(slog_za_testiranje, Q, r ,rp), I(), UI(nađen))
```

```
POČETAK PROCESA provera_kandidata
```

```
    POSTAVI i <- transformacija_kljuca(k(slog_za_testiranje), Q, rm)
```

```
    AKO JE r > rp // Ako je trenutni baket r "posle" pocetnog baketa rp (u cikličnom poretku)...
```

```
        AKO JE ((rm > r) I (rm < rp)) TADA
```

```
            // Kandidat se nalazi u opsegu "izmedju" r i rp, pa ga označavamo kao
```

```
pronađenog
```

```
                POSTAVI nađen <- 1
```

```
            KRAJ AKO
```

```
    INAČE
```

```
        AKO JE ((rm < rp) I (rm > r)) TADA
```

```
            // Kandidat se nalazi u opsegu "posle" r ali pre rp, pa ga označavamo kao
```

```
pronađenog
```

```
                POSTAVI nađen <- 1
```

```
            KRAJ AKO
```

```
    KRAJ AKO
```

```
KRAJ PROCESA provera_kandidata
```

Formiranje rasute datoteke sa linearnim traženjem u dva prolaza

Formiranje rasute datoteke sa linearnim traženjem u dva prolaza - Koraci

1. Definisavanje strukture:
 - a. Odrediti veličinu baketa, broj baketa i vrstu identifikatora slogova
2. Kreiranje prazne datoteke:
 - a. Inicijalizovati rasutu datoteku sa praznim vrednostima
3. Prvi prolaz – Upis slogova:
 - a. Čitanje slogova iz ulazne serijske datoteke
 - b. Izračunavanje adrese baketa pomoću hash funkcije (metoda ostatka pri deljenju).
 - c. Upis slogova u matični baket ako postoji slobodno mesto; u suprotnom, prebacivanje u privremenu datoteku prekoračilaca. (Izmena postojeće funkcije upisa novog sloga)

Formiranje rasute datoteke sa linearnim traženjem u dva prolaza - Koraci

4. Drugi prolaz – Obrada prekoračilaca:
 - a. Čitanje slogova iz privremene datoteke
 - b. Pokušaj ponovnog upisa slogova u rasutu datoteku korišćenjem linearne metode sa korakom k .

Formiranje rasute datoteke sa linearnim traženjem u dva prolaza

- Implementirati formiranje u dva prolaza
 - iskoristiti serijsku datoteku sa prethodnih termina vežbi kao ulaznu

Literatura

- P. Mogin, Strukture podataka i organizacija datoteka, 3rd ed. Računarski fakultet i CET Computer Equipment and Trade, 2008.

Kraj!

Hvala na pažnji!