

Programiranje 2

Liste

Zadatak 1 Napisati biblioteku (`liste.h` i `liste.c`) za rad sa jednostruko povezanom listom koja sadrži cele brojeve:

- a) Napraviti strukturu `_cvor` kojom se predstavlja čvor liste i sadrži ceo broj (*vrednost*) i pokazivač na sledeći element liste (*sledeci*).
- b) Napisati funkciju `_cvor* novi_cvor(int broj)` koja pravi novi čvor liste, inicijalizuje njegove vrednosti i vraća adresu tog čvora.
- c) Napisati funkciju `_cvor* dodaj_na_pocetak(_cvor* glava, int br)` koja dodaje element na početak liste i vraća glavu nove liste.
- d) Napisati funkciju `void dodaj_na_pocetak2(_cvor** adresa_glave, int br)` koja dodaje element na početak liste.
- e) Napisati funkciju `void ispis_liste2(_cvor* glava, FILE* f)` koja ispisuje elemente liste u datoteku `f`. Funkciju implementirati iterativno.
- f) Prethodnu funkciju implementirati rekursivno – `void ispis_liste(_cvor* glava, FILE* f)`.
- g) Napisati funkciju `oslobodi2(_cvor* lista)` koja oslobađa memoriju koju zauzimaju čvorovi liste. Funkciju implementirati iterativno.
- h) Prethodnu funkciju implementirati rekursivno – `oslobodi(_cvor* lista)`.
- i) Napisati funkciju `_cvor* dodaj_na_kraj2(_cvor* lista, int br)` koja dodaje element na kraj liste. Funkciju implementirati iterativno.
- j) Prethodnu funkciju implementirati rekursivno – `_cvor* dodaj_na_kraj(_cvor* lista, int br)`.
- k) Napisati funkciju `void dodaj_na_kraj3(_cvor** lista, int br)` koja dodaje element na kraj liste.
- l) Napisati funkciju `_cvor* napravi_listu2(FILE* f)` koja pravi listu celih brojeva. Brojevi se učitavaju iz datoteke `f` dok se ne dođe do kraja datoteke (maksimalan i ukupan broj brojeva u datoteci nije poznat). Svaki učitani broj dodavati na **kraj** liste. Funkcija vraća glavu dobijene liste. Funkciju implementirati iterativno.
- m) Prethodnu funkciju implementirati rekursivno – `_cvor* napravi_listu(FILE* f)`.
- n) Napisati funkciju `_cvor* napravi_obrnutu_listu2(FILE* f)` koja pravi listu celih brojeva. Brojevi se učitavaju iz datoteke `f` dok se ne dođe do kraja datoteke (maksimalan i ukupan broj brojeva u datoteci nije poznat). Svaki učitani broj dodavati na **početak** liste. Funkcija vraća glavu dobijene liste. Funkciju implementirati iterativno.
- o) Prethodnu funkciju implementirati rekursivno – `_cvor* napravi_obrnutu_listu(FILE* f)`.

Napisati program koji testira rad ovih funkcija. Ime datoteke se zadaje kao argument komandne linije. Ime datoteke može biti i `stdin` i u tom slučaju kraj unosa se završava sa CTRL+D.

Primer 1: ./a.out lista.txt lista.txt: 1 5 6 8 9 0 23 -16 18 dodavanje na kraj liste: 1 5 6 8 9 0 23 -16 18 dodavanje na pocetak liste: 18 -16 23 0 9 8 6 5 1	Primer 2: ./a.out stdin 897 65 0 -34 5 6 dodavanje na kraj liste: 897 65 -34 5 6 dodavanje na pocetak liste: 6 5 -34 0 65 897
---	--

Zadatak 2 Napisati funkciju `_cvor* dodaj_element(_cvor* lista, int br)` koja dodaje element u neopadajuće sortiranu listu tako da lista ostane sortirana. Testirati funkciju pozivom u `main-u`.

Primer 1: ./a.out lista.txt lista.txt: -5 1 5 6 8 9 18 Unesi element: 12 Nova lista: -5 1 5 6 8 9 12 18	Primer 2: ./a.out stdin 200 300 400 500 Unesi element: -100 Nova lista: -100 200 300 400 500
---	---

Zadatak 3 Napisati funkciju `_cvor* spoji(_cvor* lista1, _cvor* lista2)` koja spaja dve neopadajuće sortirane liste u treću, takođe neopadajuće sortiranu listu. Funkcija ne sme da kreira nove čvorove (već samo da preraspodeli postojeće). Testirati funkciju pozivom u `main-u`. Prva lista se nalazi u datoteci čije ime je prvi argument komandne linije, a druga lista se nalazi u datoteci čije ime je drugi argument komandne linije.

Primer 1: ./a.out jedan.txt dva.txt jedan.txt: 98 200 203 400 dva.txt: 78 300 301 402 izlaz: 78 98 200 203 300 301 400 402	Primer 2: ./a.out jedan.txt dva.txt jedan.txt: 18 21 65 78 94 102 203 dva.txt: 23 78 100 izlaz: 18 21 23 65 78 78 94 100 102 203
--	--

Zadatak 4 Date su dve jednostruko povezane liste L_1 i L_2 . Napisati funkciju koja od ovih listi formira novu listu L koja sadrži naizmenično raspoređene čvorove listi L_1 i L_2 : prvi čvor iz L_1 , prvi čvor iz L_2 , drugi čvor L_1 , drugi čvor L_2 , itd. Ne formirati nove čvorove, već samo postojeće rasporediti u jednu listu. Prva lista se učitava iz datoteke čije se ime zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz. Ne praviti nove čvorove liste, već ispremeštati postojeće.

Primer 1: ./a.out dat1.txt dat2.txt dat1.txt: 98 -200 203 400 dat2.txt: 78 300 -301 402 izlaz: 98 78 -200 300 203 -301 400 402	Primer 2: ./a.out jedan.txt dva.txt jedan.txt: 65 21 18 78 102 203 94 dva.txt: 100 23 78 izlaz: 65 100 21 23 18 78 78 102 203 94
--	--

Zadatak 5 Napisati funkcije za izbacivanje elementa iz liste:

- a) `_cvor* izbaci1(_cvor* lista, int br)` koja izbacuje prvo pojavljivanje datog elementa u listi i vraća glavu nove liste. (prokomentarisati zašto bi moralo da se piše `void izbaci1(_cvor** lista, int br)` u slučaju kada bi pravili funkciju koja nema povratnu vrednost).

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
1 56 23 7 9 -90
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880 45
Unesi element:
880

izlaz:
-4 57 880 3 45 880 45
```

- b) `_cvor* izbaci2(_cvor* lista, int br)` koja izbacuje poslednje pojavljivanje datog elementa u listi i vraća glavu nove liste.

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
1 56 7 23 9 -90
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880
Unesi element:
880

izlaz:
880 -4 57 880 3 45
```

- c) `_cvor* izbaci3(_cvor* lista, int br)` koja izbacuje svako pojavljivanje datog elementa u listi i vraća glavu nove liste.

```
Primer 1:
./a.out dat1.txt
dat1.txt:
1 56 7 23 7 9 -90
Unesi element:
7

izlaz:
1 56 23 9 -90
```

```
Primer 2:
./a.out dat1.txt
dat1.txt:
880 -4 57 880 3 45 880 45
Unesi element:
880

izlaz:
-4 57 3 45 45
```

Testirati funkcije pozivom u `main-u`. U funkcijama ne praviti novu listu već brisati elemente postojeće. Voditi računa o oslobađanju memorije.

Zadatak 6 Sadržaj datoteke je aritmetički izraz koji može sadržati zagrade `{`, `[` i `(`. Napisati program koji učitava sadržaj datoteke `izraz.txt` i korišćenjem steka utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz.

```
Primer 1:
izraz.txt:
{[23 + 5344] * (24 - 234)} - 23

izlaz:
Zagrade su dobro uparene
-----
```

```
Primer 2:
izraz.txt:
{[23 + 5] * (9 * 2)} - {23}

izlaz:
Zagrade su dobro uparene
-----
```

```
Primer 3:
izlaz.txt:
{[2 + 54] / (24 * 87)} + (234 + 23)

izlaz:
Zagrade nisu ispravno uparene
-----
```

```
Primer 4:
izlaz.txt:
{(2 - 14) / (23 + 11)} * (2 + 13)

izlaz:
Zagrade nisu ispravno uparene.
```

```
Primer 5:
izlaz.txt:

izlaz:
Zagrade jesu ispravno uparene.
```