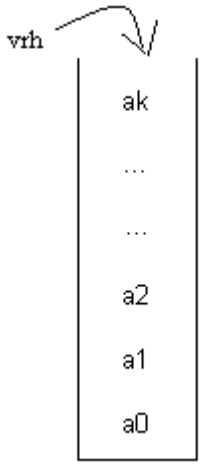
ALGORITMI I STRUKTURE PODATAKA – GRUPA 9
DODACI UZ DEVETI, DESETI I JEDANAESTI TJEDAN NASTAVE
Robert Manger, 24. travnja 2007.

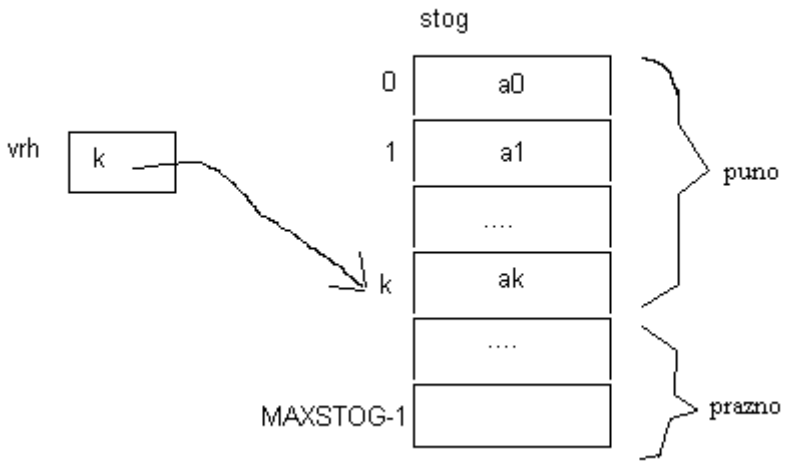
STOG (STACK)

Stog je struktura sastavljena od podataka istog tipa, nad kojom se obavljaju dvije karakteristične operacije:
- dodaj (push): ubacuje novi podatak u strukturu,
- skini (pop): izbacuje onaj podatak koji je zadnji bio ubačen.

Stog obično zamišljamo kao «stupac» (uspravno postavljeni niz) podataka, koji ima svoje podnožje i vrh. Dodavanje i skidanje podataka obavlja se na vrhu.



Prikaz stoga pomoću polja

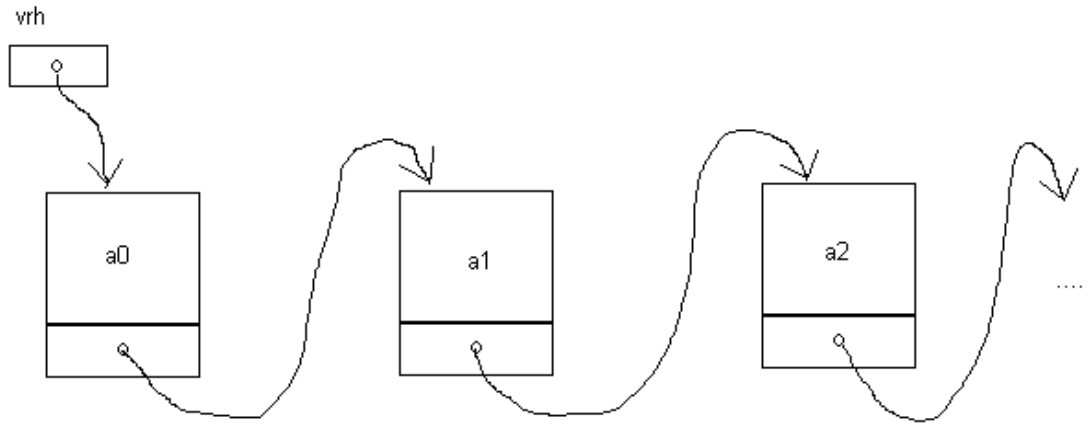


Prazan stog prepoznajemo po tome što kazaljka vrh ima vrijednost –1.

Operacija dodaj obavlja se tako da se kazaljka vrh poveća za 1, te da se na mjesto u polju koje kazaljka vrh pokazuje nakon povećavanja upiše novi podatak. Pritom treba paziti na iznimni slučaj kad je polje već popunjeno.

Operacija skini obavlja se tako da se s mjesta u polju kojeg pokazuje kazaljka vrh pročita podatak, te da se nakon toga kazaljka vrh smanji za 1. Pritom treba paziti na iznimni slučaj kad je stog prazan.

Prikaz stoga pomoću vezane liste



Prazan stog prepoznajemo po tome što je pokazivač vrh jednak NULL.

Operacija dodaj obavlja se dinamičkom alokacijom novog zapisa, upisom novog podatka u taj novi zapis, te uključivanjem novoalociranog zapisa na početak vezane liste (prespajanje pokazivača).

Operacija skini obavlja se tako da se pročita podatak iz prvog zapisa u vezanoj listi, da se prespoje pokazivači tako da se dotadašnji prvi zapis isključi iz vezane liste, te da se dealocira (oslobodi) memorija koju je zauzimao dotadašnji prvi zapis.

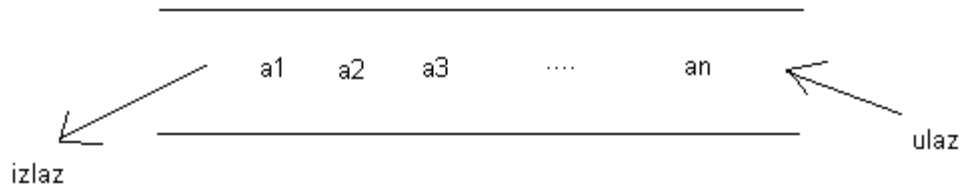
Prednosti i mane dvaju prikaza stoga

- Oba omogućuju izvršavanje operacija dodaj i skini u konstantnom vremenu.
- Kod prikaza pomoću polja može doći do prepunjenja polja, dakle pojavljuje se umjetno i neželjeno ograničenje na broj podataka koji mogu biti u stogu.
- Prikaz pomoću vezane liste troši više memorije, budući da se uz svaki podatak pohranjuje i pokazivač na idući podatak.

RED (QUEUE)

Red je struktura koja sastavljena od podataka istog tipa, nad kojom se obavljaju dvije karakteristične operacije:
- dodaj (enqueue): ubacuje novi podatak u strukturu,
- skini (dequeue): izbacuje onaj podatak koji je prvi bio ubačen.

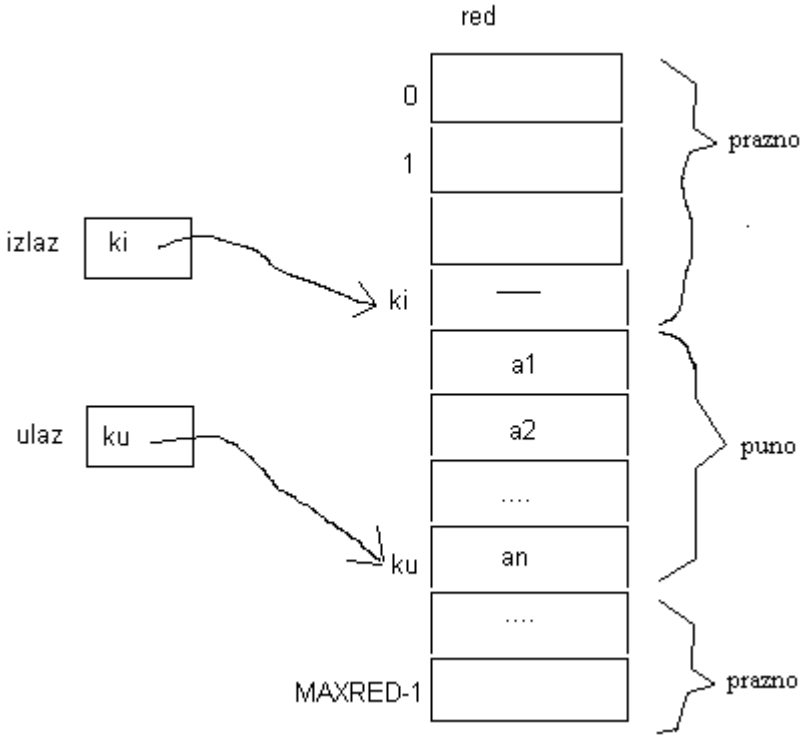
Stog obično zamišljamo kao «cijev» (vodoravno postavljeni niz) podataka, koji ima dva otvora: ulaz i izlaz. Dodavanje podataka obavlja se na ulazu, a skidanje na izlazu.



Prikaz reda pomoću polja

Oponašamo prikaz stoga pomoću polja. No sad trebamo dvije kazaljke: ulaz i izlaz. Da bi slučaj praznog reda prepoznavali po tome što obje kazaljke pokazuju na isto mjesto, dogovorit ćemo se da kazaljka izlaz pokazuje jedno mjesto ispred čelnog podatka (dakle slobodno mjesto).

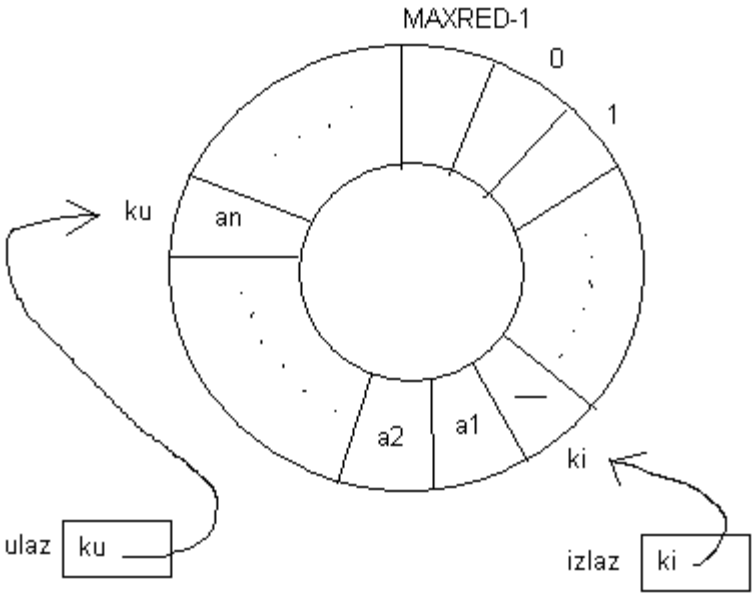
Prva (naivna) varijanta prikaza izgleda ovako:



Dodavanje odnosno skidanje podataka obavlja se na očigledan način, dakle tako da se najprije poveća odgovarajuća kazaljka ulaz odnosno izlaz, te da se zatim upiše odnosno pročita element u polju kojeg pokazuje ta kazaljka.

Ova naivna varijanta prikaza zapravo nije dobra zato jer zbog dodavanja i skidanja podataka red «putuje» prema dolje i «udara» u donji rub polja, makar u početnom dijelu polja ima slobodnih mjesta.

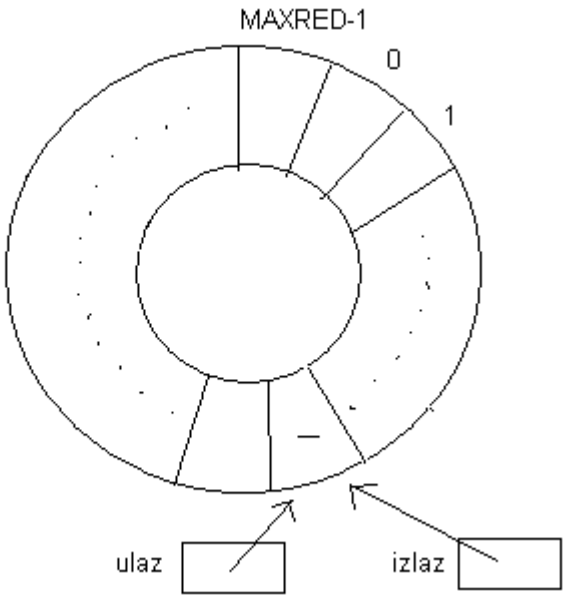
Bolja varijanta prikaza je cirkularno polje:



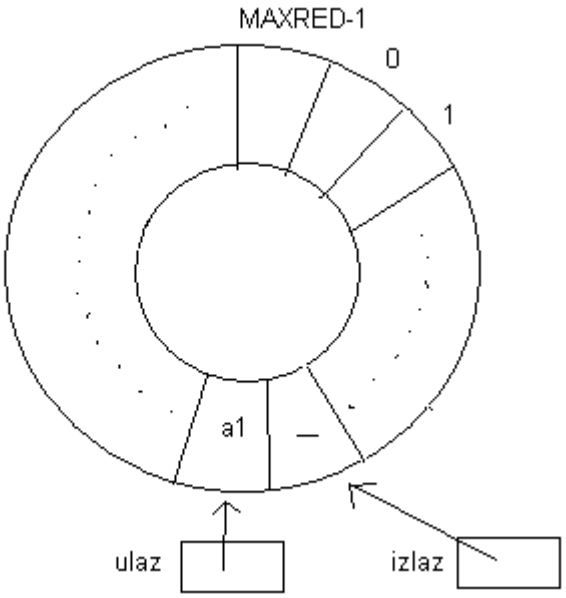
Znači smatramo da iza zadnjeg elementa u polju ponovo slijedi početni element. Dodavanje i skidanje podataka obavlja se na otprilike isti način kao u naivnoj varijanti, dakle pomicanjem kazaljki i pisanjem/čitanjem na pokazanom mjestu u polju. Vidimo da dodavanjem i skidanjem podataka red «kruži» po polju u smjeru kazaljke na satu. «Cirkularnost» se realizira tako da se indeksi elemenata polja računaju u modulo aritmetici (operator %). Preciznije, uvećavanje indeksa i za 1 izgleda ovako: `i = (i+1) % MAXRED`; Uvećavanjem zadnjeg indeksa `MAXRED-1` dobiva se početni indeks 0.

Promotrimo kako u prikazu reda pomoću cirkularnog polja izgledaju neki specijalni slučajevi.

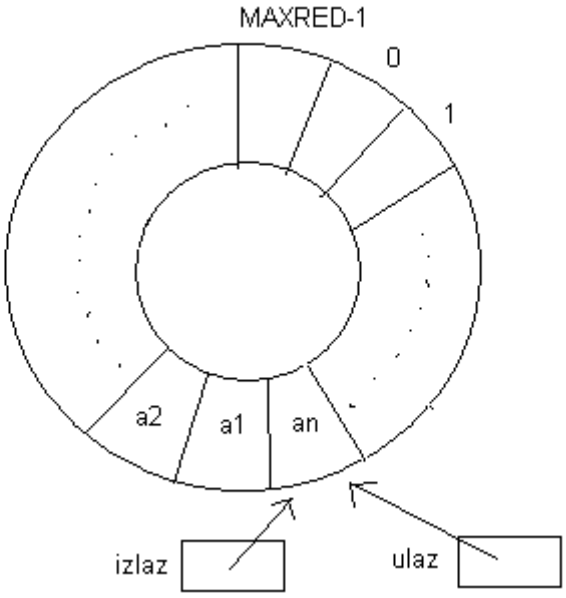
Prazni red:



Red s točno jednim podatkom:

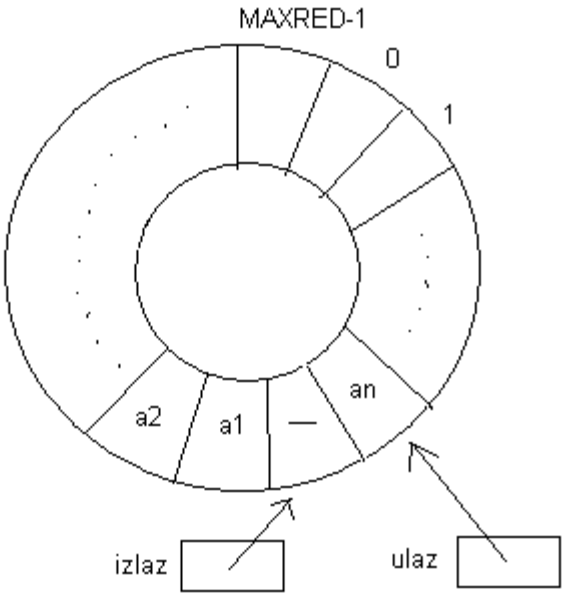


Potpuno puni red s MAXRED podataka:



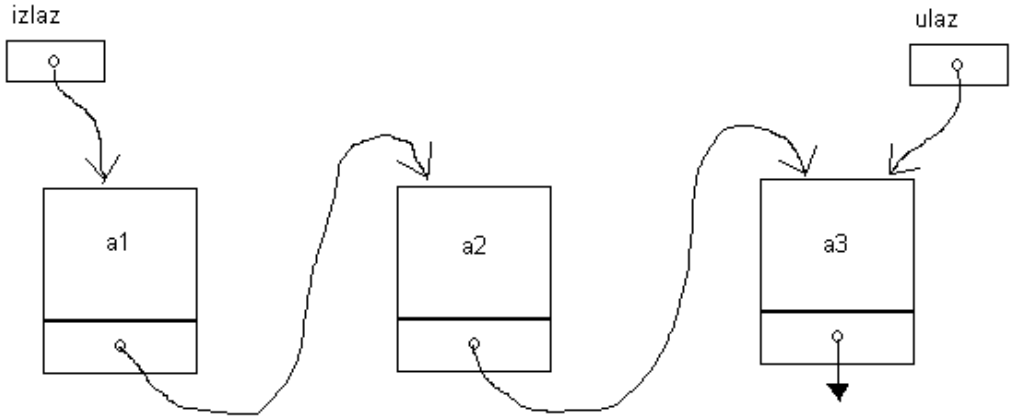
Vidimo da u prikazu postoji dvoznačnost – prazni i potpuno puni red izgledaju isto. Da bi izbjegli tu dvoznačnost, moramo ustrajati na tome da postoji bar jedno slobodno mjesto u polju na kojeg pokazuje kazaljka izlaz. Dakle ne smijemo dopustiti da u polju bude više od MAXRED-1 podataka. Drugim riječima, s dodavanjem moramo prestati ako je (ulaz+1) % MAXRED == izlaz .

Cirkularno polje s MAXRED-1 podataka izgleda ovako:



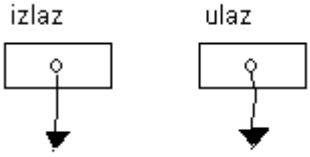
Prikaz reda pomoću vezane liste

Oponašamo prikaz stoga pomoću vezane liste. No sad trebamo dva pokazivača.

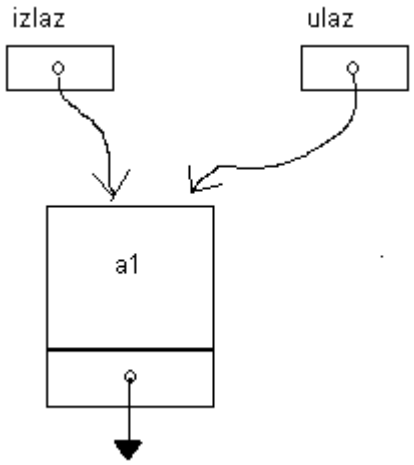


Promotrimo opet kako izgledaju neki specijalni slučajevi.

Prazni red:



Red s točno jednim podatkom:



Operacija dodaj obavlja se dinamičkom alokacijom novog zapisa, upisom novog podatka u taj novi zapis, te uključivanjem novo-alociranog zapisa na završni kraj vezane liste. U slučaju da vezana lista nije bila prazna, to uključivanje obavlja se tako da se pokazivač u dotadašnjem zadnjem zapisu preusmjeri na novo-alocirani zapis. U slučaju prazne vezane liste uključivanje se obavlja preusmjeravanjem pokazivača izlaz. U oba slučaja također se preusmjerava i pokazivač ulaz tako da i on pokaže na novo-alocirani zapis.

Operacija skini obavlja se tako da se pročita podatak iz prvog zapisa u vezanoj listi, da se prespoje pokazivači tako da se dotadašnji prvi zapis isključi iz vezane liste (promjena pokazivača izlaz), te da se dealocira (oslobodi) memorija koju je zauzimao dotadašnji prvi zapis. Sve se to obavlja pod uvjetom da vezana lista nije prazna. Još jedan iznimni slučaj na koji moramo paziti je skidanje iz liste koja se sastoji od samo jednog zapisa – tada nakon skidanja lista postaje prazna, pa pokazivač ulaz, kao i izlaz, također mora dobiti vrijednost NULL.

Prednosti i mane dvaju prikaza reda

- Analogno kao kod stoga.
- Oba prikaza omogućuju izvršavanje operacija dodaj i skini u konstantnom vremenu.
- Kod prikaza pomoću polja može doći do prepunjenja polja, dakle pojavljuje se umjetno i neželjeno ograničenje na broj podataka koji mogu biti u redu.
- Prikaz pomoću vezane liste troši više memorije, budući da se uz svaki podatak pohranjuje i pokazivač na idući podatak.

LISTA

Lista je struktura sastavljena od podataka istog tipa. Podaci su organizirani u niz, tj znamo koji od njih je prvi, koji je drugi itd. Obavljaju dvije karakteristične operacije:
- dodaj (insert): ubacuje novi podatak u strukturu,
- skini (delete): izbacuje podatak iz strukture.

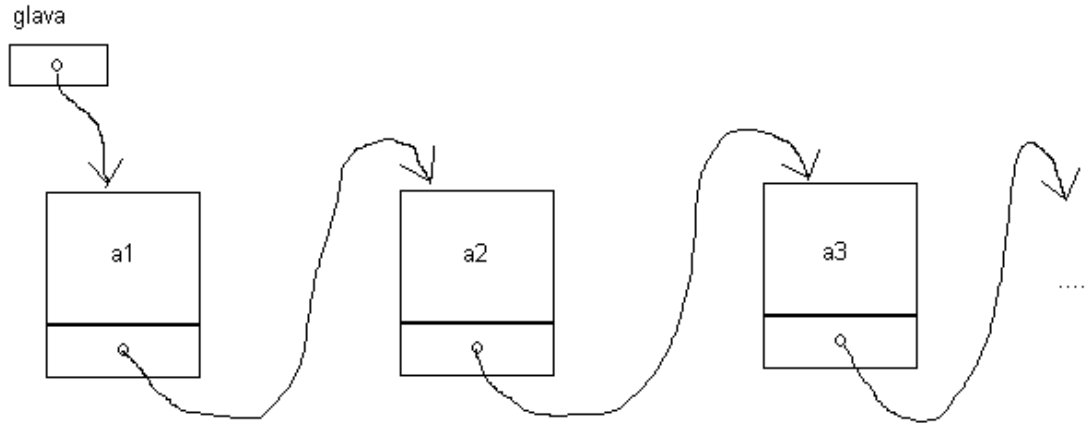
Za razliku od stoga i reda, gdje se dodavanje i skidanje obavljalo na točno određenim mjestima, sada se ono može obavljati na bilo kojem mjestu u nizu.

Prikaz liste pomoću polja

Korištenje polja za prikaz liste je moguće i bilo bi slično kao kod stoga. Ipak, takav prikaz ne bi bio baš efikasan jer bi zbog dodavanja podataka na proizvoljna mjesta dolazilo do fizičkog prepisivanja prethodno dodanih podataka. Dakle, vrijeme izvršavanja osnovnih operacija ovisilo bi o duljini liste.

Prikaz liste pomoću vezane liste

Slično kao kod stoga ili reda. Dovoljna nam je jednostruka vezana lista s pokazivačem na početak.

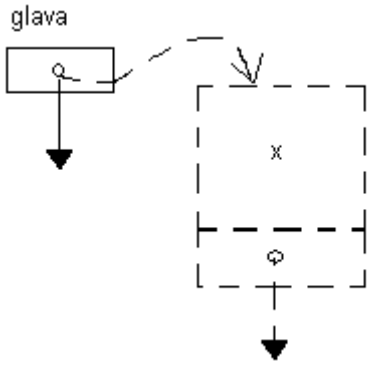


Sortiranje pomoću vezane liste

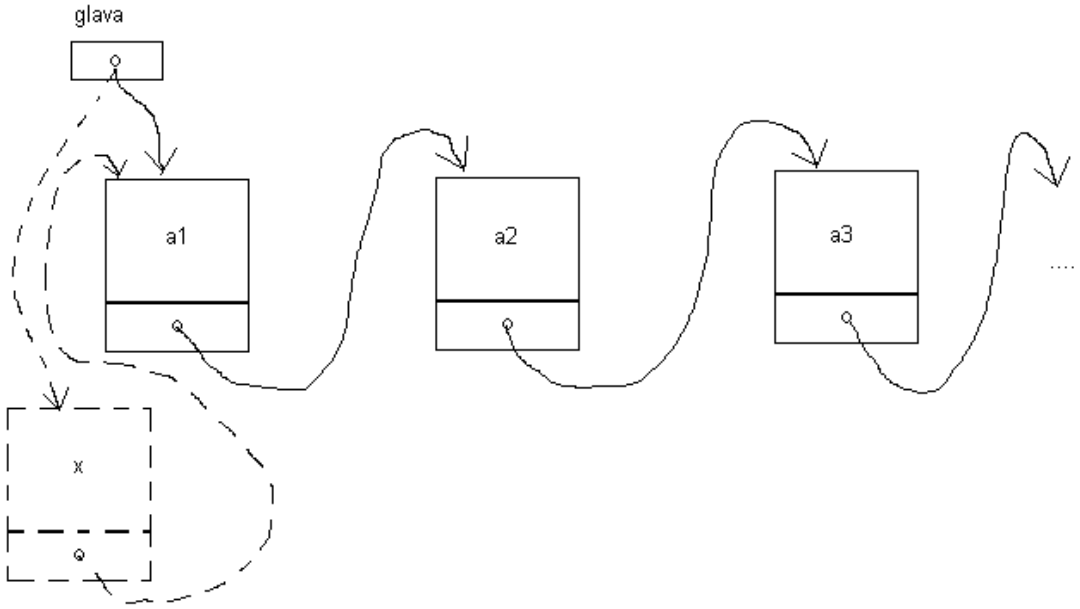
Jedna primjena (jednostruko vezane) liste ilustrirana je programom Lista. Taj program čita podatke iz datoteke i redom ih dodaje u listu. Pritom se svaki novi podatak ubacuje u već stvoreni dio liste na «pravo mjesto» u smislu sortiranog redoslijeda. Nakon što dodamo sve podatke, čitanjem liste od početka prema kraju dobit ćemo podatke u sortiranom redoslijedu. Cijeli postupak može se shvatiti kao algoritam za sortiranje (varijanta insertion sort-a bez fizičkog pomicanja podataka).

Kod dodavanja novog podatka x u već sortiranu listu (a_1, a_2, ... a_n) moguća su četiri slučaja:

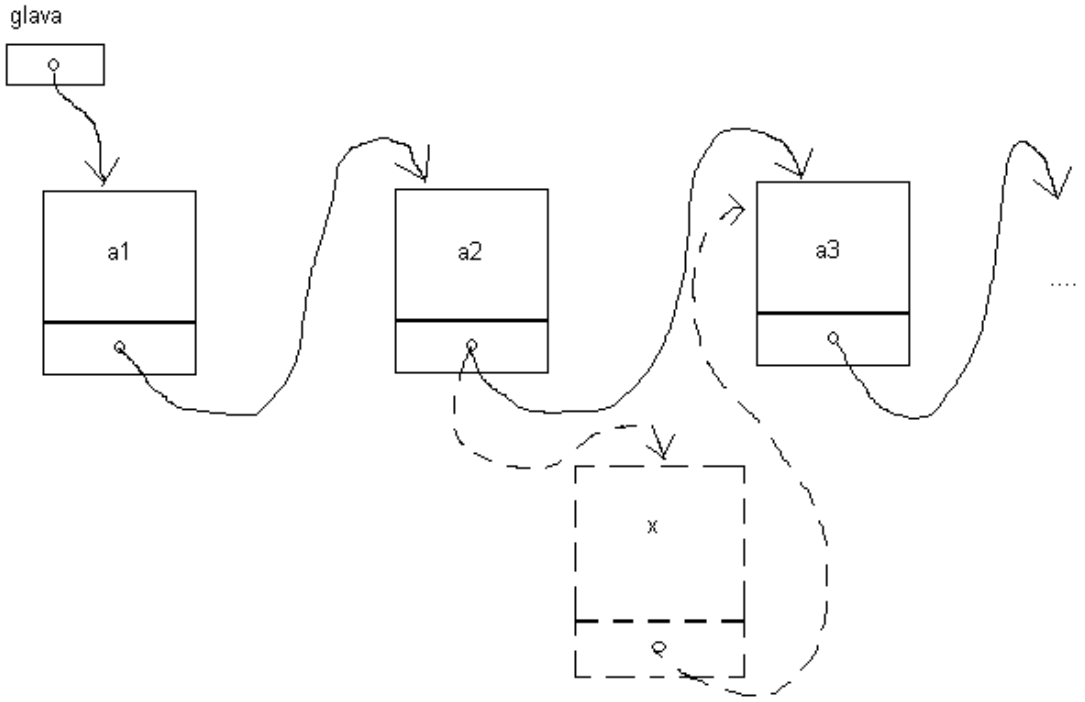
- Lista je prazna,



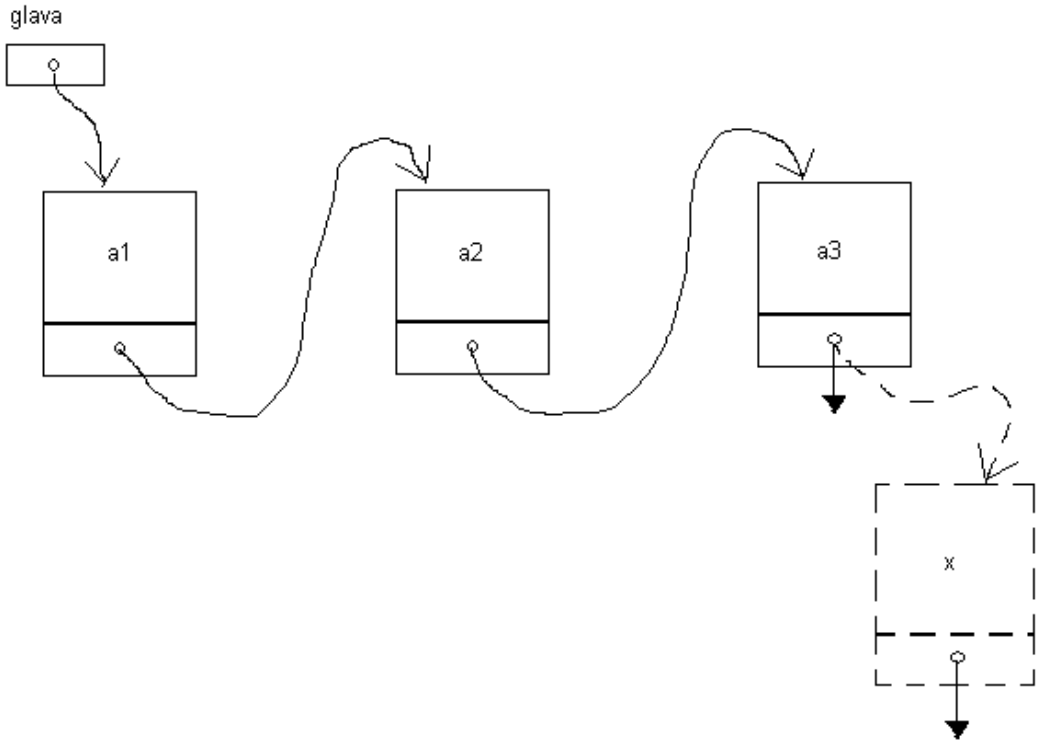
- Lista je neprazna, $x <= a_1$,



- Lista je neprazna, $a_i < x <= a_{i+1}$ za neki i,



- Lista je neprazna, $x > a_n$.

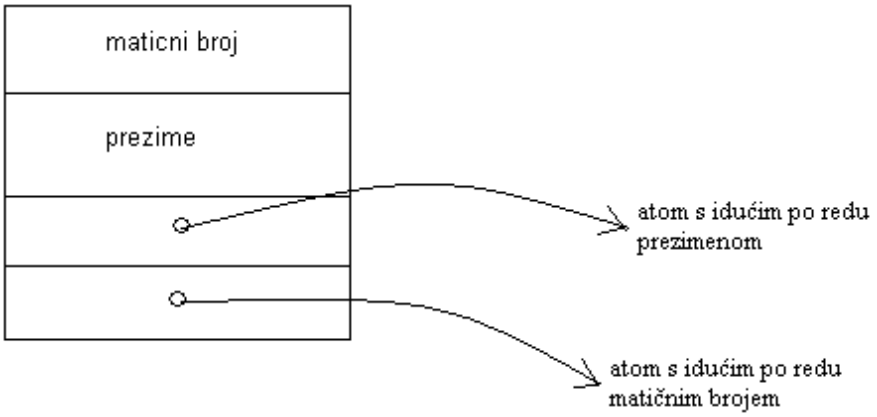


Svaki od ova četiri slučaja rješava se dinamičkim alociranjem novog zapisa, upisom novog podatka x u taj novi zapis, te uključivanjem novog zapisa na odgovarajuće mjesto u vezanu listu promjenom odgovarajućih pokazivača. Na gornjim slikama crtkane strelice označavaju promjene u podacima.

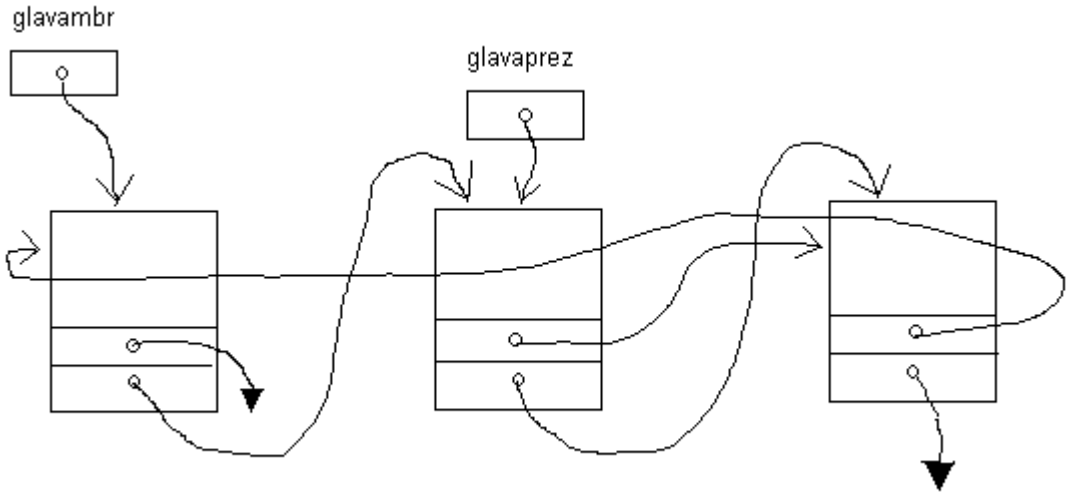
Spretnim kodiranjem moguće je gornja četiri slučaja svesti na dva:
- Dodavanje na početak liste, mijenja se pokazivač glava,
- Dodavanje iza postojećeg zapisa, mijenja se pokazivač u tom postojećem zapisu.

Lista s više ključeva

Riječ je o strukturi koja se sastoji od atoma koji sadrže više podataka (npr. matični broj i prezime studenta). Umjesto jednog pokazivača, u atomu postoji više pokazivača od kojih se svaki odnosi na jedan od kriterija za sortiranje (npr. sortiranje po matičnom broju, sortiranje po prezimenu).



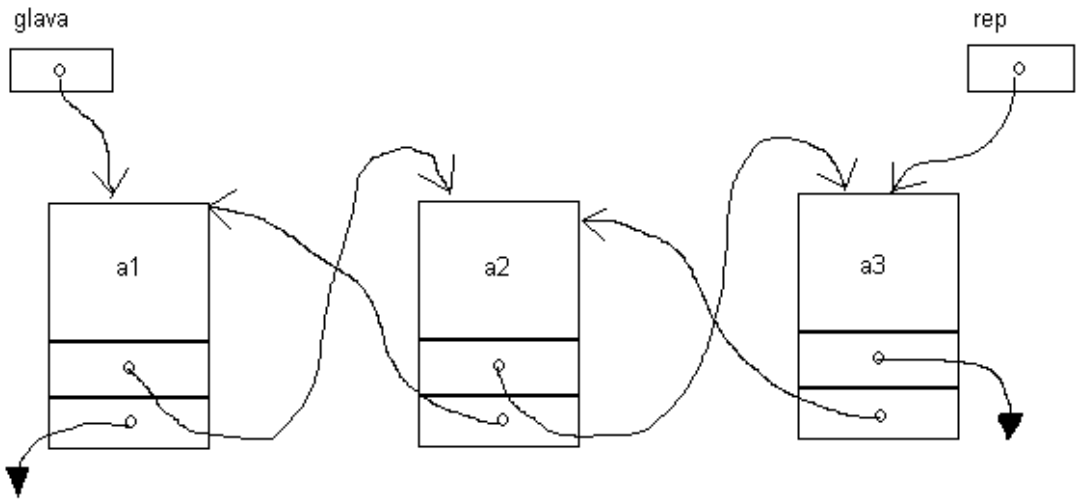
Unutar jedne liste s više ključeva, jedan atom istovremeno je uključen u više vezanih listi. Svaka vezana lista mora imati svoju glavu. Time je omogućeno da isti (samo jednom upisani) podaci budu u istovremeno sortirani na više načina.



Da bi podatke pročitali sortirano po određenom kriteriju, krećemo od odgovarajuće glave i putujemo od atoma do atoma tako da slijedimo odgovarajući pokazivač unutar atoma.

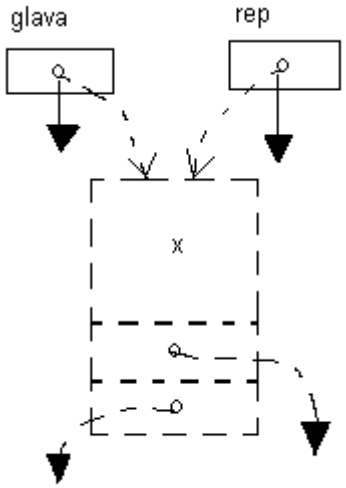
Dvostruko vezana lista

Riječ je o strukturi gdje svaki atom osim pokazivača na sljedeći atom sadrži i pokazivač na prethodni atom. Sukladno tome, osim pokazivača na početak liste (glava) imamo i pokazivač na kraj liste (rep). Moguće je manevrirati po listi u oba smjera, dakle možemo krenuti od glave i slijediti pokazivače od prvog atoma prema zadnjem, ili možemo krenuti od repa i slijediti pokazivače od zadnjeg atoma prema prvom. Ovakve mogućnosti manevriranja korisne su u aplikacijama gdje povremeno trebamo mijenjati «smjer kretanja» kroz listu. No bolja efikasnost kod manevriranja ima i svoju cijenu: operacije dodavanja i skidanja podataka postaju kompliciranije jer moramo ažurirati obje vrste pokazivača.

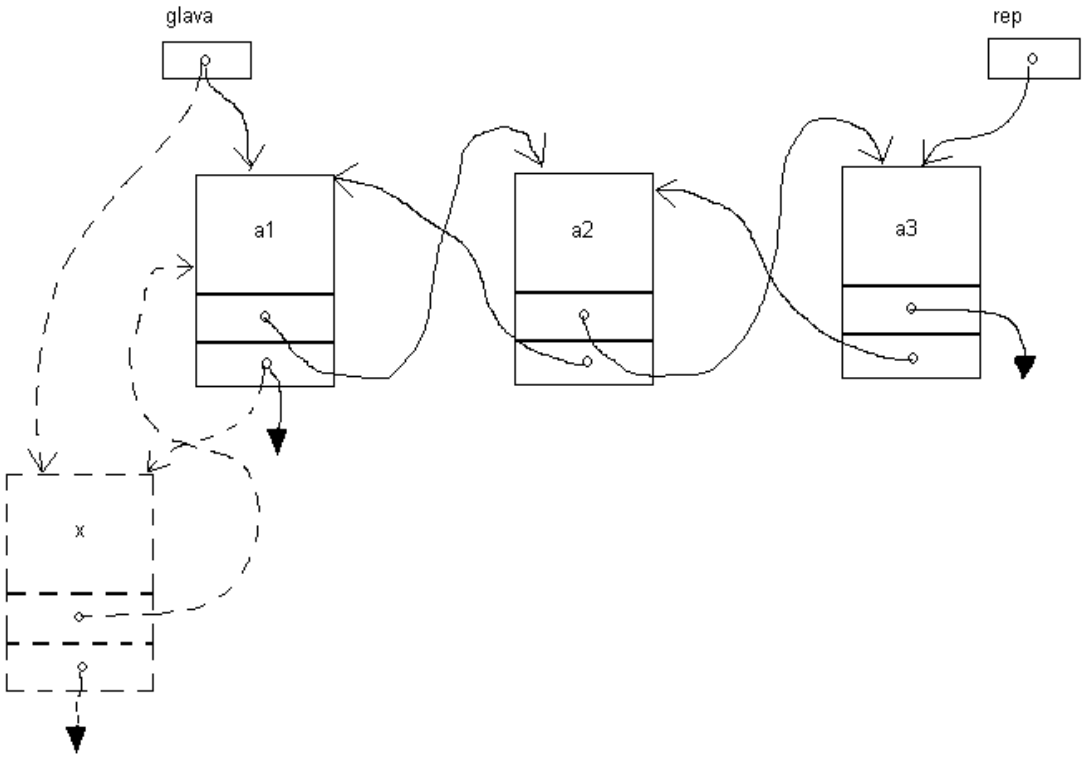


Detaljno analiziramo dodavanje novog atoma u dvostruko vezanu listu. Postoje četiri slučaja.

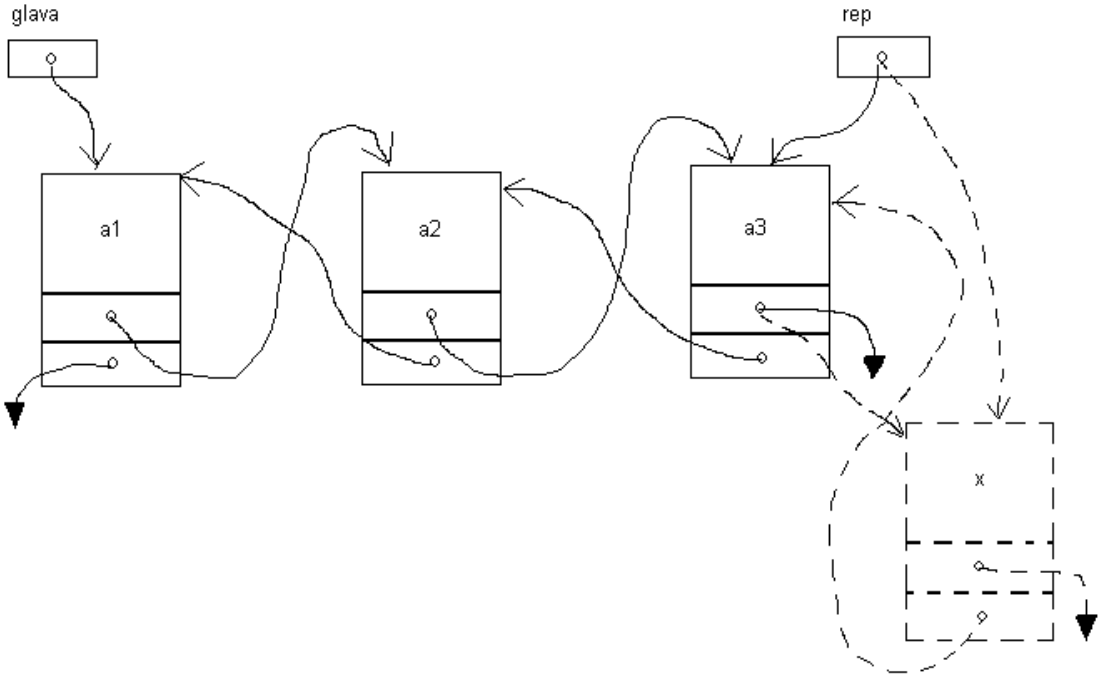
- lista je prazna:



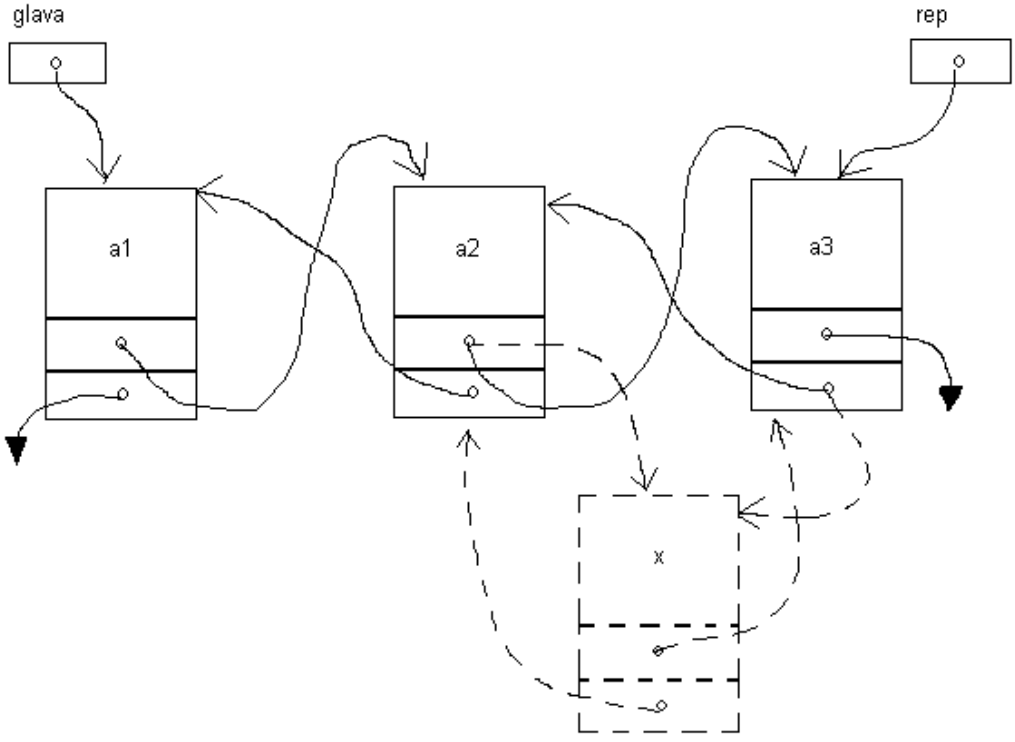
- lista je neprazna, ubacujemo na početak:



- lista je neprazna, ubacujemo na kraj:



- lista je neprazna, ubacujemo negdje u «sredinu»:



Svaki od ova četiri slučaja rješava se dinamičkim alociranjem novog zapisa, upisom novog podatka x u taj novi zapis, te uključivanjem novog zapisa na odgovarajuće mjesto u dvostruko vezanu listu promjenom odgovarajućih pokazivača. Na gornjim slikama crtkane strelice opet označavaju promjene u podacima. Vidimo da je u svakom od slučajeva potrebno promijeniti dva pokazivača u postojećoj strukturi.