Datum: 8.10.2018.

Linearne strukture podataka – stek

Linearne strukture podataka su kolekcije sačinjene od pravolinijski (linearno) aranžiranih elemenata. Kada linearnim strukturama dodajemo ili iz njih uklanjamo podatak, strukture rastu ili se smanjuju. Možete ih zamisliti kao da su u nekom ovitku i elementi u njima nikako ne mogu mijenjati poredak. Dvije važne linearne strukture su stek i red.

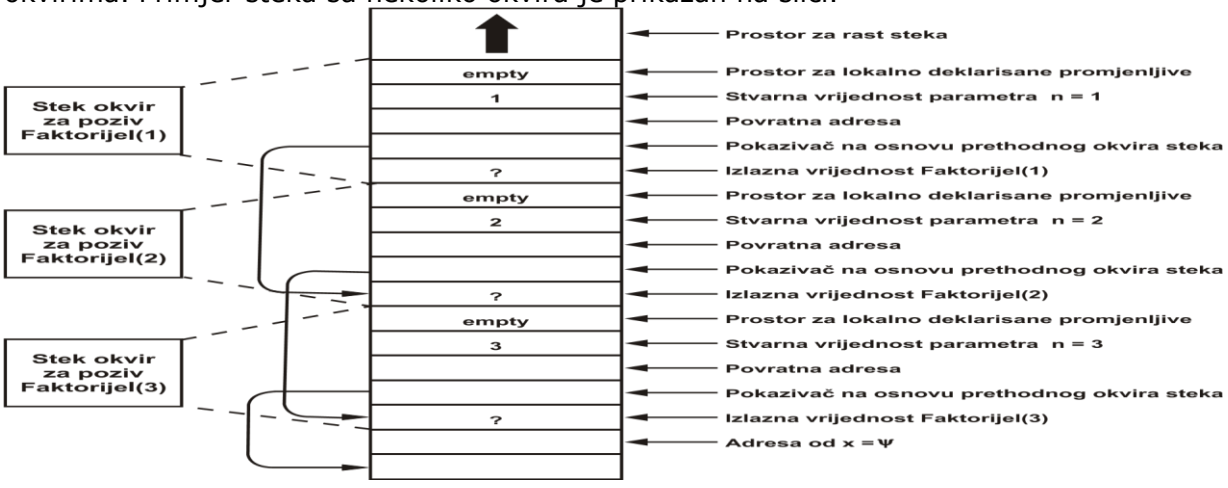
Ukoliko se ograničimo pravilom da se podaci i dodaju i izbacuju samo na jednom kraju, onda takvu strukturu nazivamo stek (*stack*). Stek se još naziva i LIFO struktura (Last-InFirst-Out) jer se prilikom uklanjanja elementa it steka uklanja element koji je posljednji dodan.

Primjer korištenja steka je UNDO operacija u MSExcel-u, koja uvijek poništava samo posljednju izvršenu radnju.

Stekovi se u različitim algoritmima često koriste za čuvanje informacija o odloženim obavezama koje se tek trebaju procesirati. Stekovi se koriste za praćenje niza poziva funkcija tokom izvršavanja programa (stek aktivacijskih slogova). Svaki put kada se pozove funkcija, u stek se umetne aktivacijski slog. Slog sadrži prostor za informacije koje su potrebne tokom izvršavanja poziva funkcije i informacije o nastavku pozivnog programa nakon što se završi sa izvršavanjem funkcije. Stekovi su idealna struktura podataka za ovakve namjene jer su pozivi funkcija uvijek dinamički ugniježđeni (slučaj kada se poziva funkcija u drugoj funkciji, tada je vrijeme pozivanog programa ugniježđeno unutar vrijemena poziva pozivnog programa).

Neki raniji programski jezici, nisu dozvoljavali rekurzivne pozive funkcija. U **C**-u je ovaj problem riješen upotrebom steka. Pošto su pozivi rekurzivnih funkcija ugniježđeni jedan unutar drugog i pošto procesiranje pozivane prekida procesiranje funkcije koja poziva, prirodno je da se za ove ugniježđene rekurzivne pozive koriste stekovi sa okvirima rekurzivnih funkcija.

Pri pozivu funkcije F(a_1, a_2, …, a_n) sa stvarnim argumentima a_i, i=1, … ,n, u **C**-u se koristi takozvani run-time stek. Pripremi se odgovarajući skup informacija za poziv funkcije, takozvani okvir steka, i smjesti se na vrh steka sa drugim, prethodno generiranim, okvirima. Primjer steka sa nekoliko okvira je prikazan na slici.



ATP stek

Apstraktni tip podataka (ATP) se sastoji od kolekcije struktura podataka i skupa operacija definiranih na njima.

Strukture podataka su obično složene, dobijene primjenom metoda struktuiranja na kolekciju komponenti.

Ukoliko krenemo od niza, kao osnovne strukture podataka, onda stekove i redove za čekanje možemo definirati pomoću adekvatnih operacija.

Stek, S, sastavljen od elemenata tipa T, je niz elemenata tipa T na kome su definirane sljedeće operacije:

1. **Iniciranje** steka S da bude prazan stek,
2. Utvrđivanje da li je stek S **prazan**,
3. Umetanje novog elementa na vrh steka (**push**),
4. Ako je S neprazan, uklanjanje elementa sa vrha steka (**pop**),

Dodatno, mogu se definisati i druge funkcije, kao

1. Pozicioniranje vrha steka (**top**),
2. Utvrđivanje da li je stek S **pun**.

Funkcija PunStek je imala smisla dok je doalociranje memorije za sekvencijalnu reprezentaciju bilo „skupo", ali je implementacijom koncepta dinamičkih nizova postala suvišna. Funkcija Top nije obavezna u literaturi, te ćemo je smatrati opcionom.

Primjene ATP steka

Provjera ispravnosti zagrada

Prva primjena ATP stekova je program za provjeru jesu li sve zagrade u jednom izrazu u paru. Ispravnost zagrada se provjerava i prilikom pisanja programskog koda u novim okruženjima.

Da bi izraz bio valjan, sve otvorene zagrade u njemu moraju biti adekvatno zatvorene. Na primjer, u izrazu

$$\{ a^2 - [(b+c)^2 - (d+e)^2] * [\sin (x - y)] \} - \cos (x+y) ,$$

pojavljuju se sve tri vrste zagrada i to u sljedećem rasporedu $\{[( )( )] [( )]\}( )$.
Da bi provjerili ispravnost zagrada u ovakvom izrazu:

1. počinjemo od praznog steka,
2. čitamo izraz kao znakovnu nisku s lijeva na desno.
3. Kada god naiđemo na simbol otvorene zagrade, umetnemo je na vrh steka.
4. Kada naiđemo na simbol zatvorene zagrade,
    a. Ispitujemo da li je stek prazan,
        i. ako jeste, javljamo grešku i izlazimo iz programa,
        ii. ako nije,
            1. uzimamo element sa vrha steka i
            2. provjeravamo je li par zagrada (otvorene i zatvorene zagrade) adekvatan,

   a. ako nije, javljamo grešku i izlazimo iz programa,
   b. ako jeste, idemo na 5.
5. Vraćamo se na tačku 2. sve dok ne pročitamo čitavu nisku.
6. Kada završimo čitanje izraza, provjeravamo je li stek prazan,
   a. Ako jeste, onda su zagrade ispravne;
   b. ukoliko nije, postoji neka zagrada "viška".

Postoje tri vrste greške. Prva, ukoliko u toku čitanja niske stek isprazni, a učita se simbol zatvorene zagrade (tada u izrazu postoji neadekvatna zatvorena zagrada i izdaje se greška). Druga, kada se pročita čitava niska, a stek nije prazan, tada u izrazu postoji otvorena zagrada 'viška'. Treća, kada se prilikom provjere ustanovi da zagrade nisu pravilno uparene.

Još jedna primjena steka – izračunavanje algebarskih izraza

Primjena ATP stekova za izračunavanje postfiks izraza. Algebarski izrazi se mogu prikazati na dva načina. Prvi, češći, je kada se binarni operator β nalazi između lijevog i desnog operanda, L i D, kao u izrazu (L β D). Zagrade se koriste za isticanje prioriteta operacija. Drugi je takozvani postfiks[1] način, kada se prvo pišu operandi pa iza njih operator, L D β. Na ovaj način se izbjegava upotreba zagrada.
Stek se za izračunavanje postfiks izraza, P, koristi na sljedeći način:

1. Čitamo izraz P s lijeva na desno.
2. Kada naiđemo na operand, ubacujemo ga u stek za računanje, S.
3. Kada prepoznamo operator, β
   a. iz S uzimamo posljednji operand i
   b. smjestimo ga u promjenljivu D,
   c. uzmemo posljednji operand i smjestimo ga u promjenljivu L i
   d. izračunamo izraz L β D i
   e. ubacimo ovu vrijednost na vrh steka.
4. Kada završimo čitanje izraza P, jedina vrijednost u steku je upravo vrijednost izraza P.

Implementacija ATP steka – sekvencijalna reprezentacija

Sekvencijalna reprezentacija steka znači da se za implementaciju apstraktog tipa podatka – steka koristi niz za sve funkcije kojima se ATP definiše. Dakle, Treba se napisati 6 funkcija (InicirajS, PrazanS, PunS, PushS, PopS, TopS), pri čemu se pretpostavlja da je stek S definisan kao niz podataka traženog tipa, sa maksimalnom veličinom MaxVelS.

Stek se, budući da ima ograničeni kapacitet, može predstaviti kao sekvencijalno uređen niz elemenata. Ne može imati više elemenata od broja određenog konstantom MaxVelS. Stek se može predstaviti strukturnim nizom koji se sastoji od:

(a) broja elemenata koji su trenutno u steku, vrijednost člana VelS i
(b) niza elemenata steka, član S.

Kažemo da je stek, po definiciji, prazan ako je vrijednost VelS jednaka nuli, a pun, kada je vrijednost VelS jednaka MaxVelS.

[1] Ovakvi izrazi su se koristili u nekim starijim verzijama džepnih kalkulatora za izračunavanje vrijednosti algebarskih izraza.

Da bi umetnuli element X na ne-pun stek S, prvo upišemo vrijednost X u niz S na poziciju određenu sa VelS, a zatim uvećamo član VelS za jedan.

Da bi uzeli element X sa vrha nepraznog steka S, prvo umanjimo član VelS za jedan, a zatim uklonimo vrijednost X iz niza S sa pozicije određene sa VelS.

Kod će biti prikazan u materijalima za vježbe.

Realociranje memorije za sekvencijalnu reprezentaciju steka

Kada se stek implementira pomoću dinamičkog niza, onda se pretpostavlja da se njegova veličina može mjenjati tokom izvršavanja programa. No, veličina se ne može mijenjati često, jer se prilikom svakog realociranja mora alocirati novi niz i u njega se kopirati sadržaj starog.

Uobičajeno je da se memorija za sekvencijalnu reprezentaciju steka doalocira na sljedeći način:

- Kada broj elemenata u steku dostigne maksimalnu veličinu niza kojim se predstavlja, njegova veličina se duplira, a elementi se kopiraju u novi niz;
- Kada se broj elemenata u steku smanji na jednu četvrtinu veličine niza, niz se smanji na polovinu veličine, elementi se kopiraju u novi niz.

Na ovaj način se uvijek u nizu nalazi između 25% i 100% elemenata.

Implementacija ATP steka – povezana reprezentacija

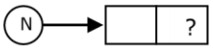
Stek se može predstaviti i pomoću povezane liste, tako da pokazivač na listu uvijek pokazuje na vrh steka. Elementi steka se smještaju u čvorove liste, a element na vrhu steka je uvijek u prvom čvoru liste.

Prazan stek se označava sa **NULL**.

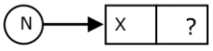
Strategija za umetanje novog prvog čvora u listu (dodavanje elementa u stek)

(deklarisati N, pokazivač na čvorove liste)

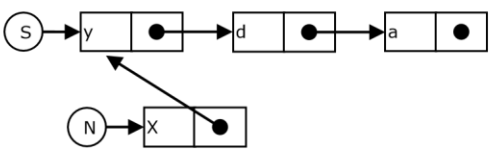
(alocirati novi čvor na koji pokazuje N)



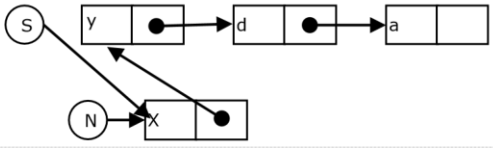
(kopirati sadržaj X u info polje novog čvora)



(postaviti N→Link da pokazuje na prvi čvor, kopiranjem vrijednosti iz S)



(postaviti S da pokazuje na novi čvor)



Da bi umetnuli element X u stek S:

1. alociramo novi čvor na koji pokazuje pokazivač N,
2. smjestimo vrijednost X u Info polje čvora na koji pokazuje N,
3. postavimo vrijednost Link polja čvora na koji pokazuje N da pokazuje na stari prvi čvor steka S
4. postavimo pokazivač S da pokazuje na novi prvi čvor, N.

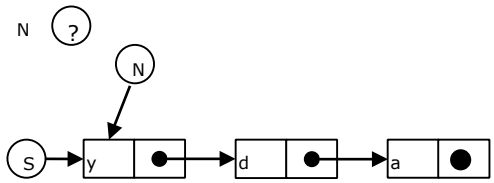
Da bi uzeli element X sa vrha nepraznog steka S,

1. smjestimo vrijednost Info polja prvog čvora InfoList u X,
2. u S smjestimo pokazivač Link polja prvog elementa S.

Strategija za uklanjanje prvog čvora iz liste (uklanjanje elementa iz steka)

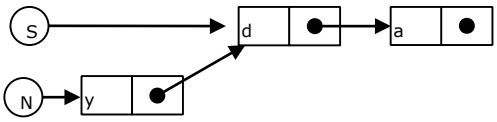
(ukoliko je stek prazan, javiti grešku, inače)

(deklarisati N, pokazivač na čvorove liste)



(postaviti N da pokazuje na prvi čvor kopiranjem sadržaja iz S)

(postaviti S da pokazuje na drugi čvor, kopiranjem vrijednosti iz N Link)



(kopirati vrijednost iz starog prvog čvora u pomoćnu promjenjivu, pa je vratiti u glavni program)