Strukture Podataka i Algoritmi

Lekcija 3

leto 2019/2020

Prof. dr Branimir M. Trenkić

Linearne strukture podataka

- ***Počinjemo sa*** izučavanjem ***najfundamentalnijih struktura podataka*** u programiranju – predstavljaju ***sekvencu (linearno uređenu) elemenata***

1. ***Nizovi***
2. ***Liste***
   - Stekovi
   - Redovi za čekanje

- Za svaku od ovih struktura podataka ***opisaćemo nekoliko konkretnih realizacija***

SP - Osnovne napomene

- ***Svaki*** viši ***programski jezik*** obezbeđuje neke ***osnovne tipove podataka***
  - Njihov repertoar se ***razlikuje od jezika do jezika***
  - Celobrojni, realni, logički, znakovni i pokazivački tip podataka
- ***Svaki jezik*** propisuje ***pravila za konstruisanje složenih tipova podataka*** polazeći od osnovnih
- Agregacioni način za struktuiranje podataka:
  - Niz (najjednostavniji)
  - Struktura
  - Datoteka

SP - Osnovne napomene

- ***Realizacija ATP*** u računaru – ***prevođenje logičkog opisa*** podatka u ***strukturu podataka***
  1. ***Upotrebom osnovnih tipova podataka*** kao gradivnih elemenata
  2. Korišćenjem raspoloživih ***agregacionih mehanizama***
- Struktura podataka:

1. ***Kolekcija elemenata*** povezanih na određeni način
2. ***Skup operacija*** dozvoljenih nad tom kolekcijom podataka

Operacije - Osnovne napomene

- ***Operacije*** se ***razlikuju*** od strukture do strukture – postoje ***bar dve zajedničke*** za sve strukture:
  1. ***Dodavanje novog elementa*** kolekciji postojećih elemenata
  2. ***Uklanjanje postojećih elemenata*** iz kolekcije elemenata
- Operacija za ***konstruisanje potpuno nove*** "prazne" strukture podataka

Operacije - Osnovne napomene

- ***Sve operacije*** se mogu podeliti u ***dve kategorije***
- ***Upiti***
- Kao rezultat daju neku informaciju o strukturi podataka ***ne menjajući je***
- ***Modifikujuće operacije***
- ***Menjaju sadržaj*** strukture podataka

ATP - Linearna lista

- ***Osnovno svojstvo*** - ***postojanje*** određenog ***poredka između elemenata strukture***
- Ova klasa podataka se može konceptualno formalizovati uvođenjem opšteg termina ***linearna lista***
- Linearna lista se definiše kao ATP (***apstraktni tip podataka***)
- Linearna lista $(a_1, ..., a_n)$ je skup od n elemenata $(n \geq 0)$ - ***linearno uređeni*** na osnovu svoje ***pozicije u listi***

ATP - Linearna lista

- Broj elemenata ***n*** - ***dužina linearne liste***
- Ako je ***n = 0*** – ***prazna linearna lista***
- Ako je ***n > 0***,

  a_1 - ***prvi element***

  a_n - ***poslednji element*** liste
- Za element a_i se kaže da se nalazi na i-toj poziciji u listi
- ***Prvi element*** liste nema svog prethodnika
- ***Poslednji element*** nema svog sledbenika

ATP - Linearna lista

- Na linearnu listu se može primeniti više ***različitih operacija***:
  - ***Pristup elementima liste*** u njihovom linearnom poretku 1, ..., n
  - ***Pretraživanje liste*** i vraćanje pozicije elementa koji ima traženu vrednost
  - ***Čitanje*** vrednosti proizvoljnog elementa liste ***ili upis*** u njega
  - ***Dodavanje novog elementa*** u proizvoljnu poziciju u listi, pri čemu se svi sledeći elementi pomeraju za po jednu poziciju naniže

ATP - Linearna lista

- Na linearnu listu se može primeniti više ***različitih operacija***:
  - ***Nalaženje prethodnika i sledbenika*** za proizvoljan element liste
  - ***Određivanje dužine*** liste
  - ***Spajanje dve liste*** u jednu, itd....

ATP - Linearna lista

- ***Fizička implementacija*** koncepta linearne liste:
  - ***Sekvencijalna***
    - Niz
  - ***Ulančana reprezentacija***
    - Povezana lista ili jednostavno lista

Nizovi

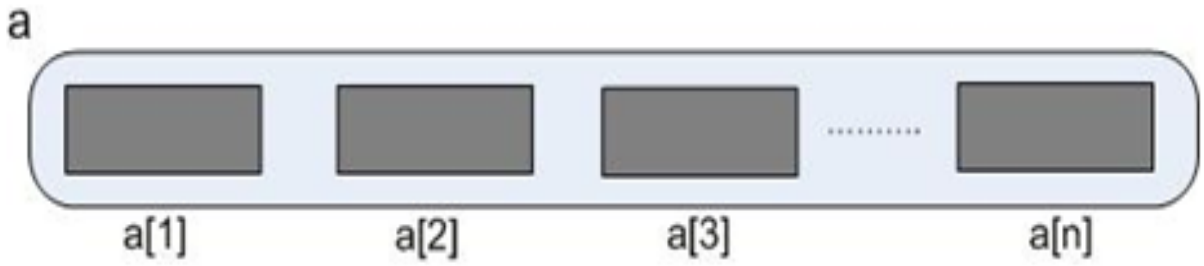
- ***Niz*** (***array***) je ***osnovni način struktuiranja podataka*** u programiranju i zato se koristi u skoro ***svim programskim jezicima***
- Niz je ***struktura podataka*** koja predstavlja ***sekvencu elemenata istog tipa***
- ***Sekvencijalna (linearna) uređenost*** niza je realizovana ***numeracijom*** njegovih elemenata, (od prvog do poslednjeg)
- ***Redni broj nekog elementa*** u nizu se naziva ***indeks elementa***

Nizovi

- ***Označavanja*** niza
- **Grupno** i
  - Koristi se ***ime niza*** i njime se predstavlja ***niz kao jedna celina***
- ***Pojedinačno*** označavanje
  - Koristi se ***poseban zapis*** koji se sastoji od ***imena niza*** i ***indeksa*** odgovarajućeg elementa niza u ***uglastim zagradama***

Nizovi

- ***Ukupan broj elemenata*** niza se naziva veličina ili ***dužina niza***
- Na primer, niz sa imenom ***a*** dužine ***n*** se sastoji od ***n elemenata istog tipa*** čiji se izgled može predstaviti na sledeći način,



Nizovi

- ***Dva glavna svojstva*** nizova su:

1. Svi elementi niza moraju biti ***istog tipa***
   - ***Homogenost*** ne uvodi ***nikakvo ograničenje po pitanju tipa***
   - Tip pojedinačnih elemenata niza može biti bilo koji, ma kako složen
2. ***Dužina niza*** mora biti ***unapred zadata i fiksna*** tj. ne može se menjati

- ***Iz ovih svojstava*** proističu ***dobre osobine*** ali i ***nedostaci*** niza kao strukture podataka

Nizovi

- Na osnovu ovih svojstava:
- Niz se odlikuje ***jedinstvenom memorijskom reprezentacijom***
- Elementi niza zauzimaju ***kontinualni niz susednih memorijskih lokacija***
- Time je ostvarena ***podudarnost*** između ***logičke forma*** niza i njegove ***fizičke forme*** (memorijske reprezentacije)

Nizovi

- Lako se može utvrditi ***veličina*** ukupnog ***memorijskog prostora*** potrebnog za smeštaj niza
- Ako je ***poznata*** – (***I***) ***dužina niza*** i (***II***) ***veličina jednog elementa*** tog niza
- Po osnovnim svojstvima niza – ***moraju se unapred naznačiti*** (I) ***dužina*** niza i (II) ***tip*** elemenata tog niza (lako se može utvrditi veličina jednog elementa)

Nizovi

- Lako je moguće proračunati ***lokaciju svakog elementa*** niza
- Ako je ***prvi element*** na mamorijskoj lokaciji ***a***
- Veličina ***jednog elementa*** je ***b*** memorijskih lokacija
- Onda se ***i-ti element niza*** nalazi na lokaciji

$$a + (i - 1)b$$

- ***Vreme pristupa*** svim elementima niza je ***konstantno*** - u algoritmima operacije čitanja ili upisa nekog elementa niza može se smatrati jediničnom instrukcijom

Nizovi

- Niz kao struktura podataka ima i određene ***nedostatke***

1. Nepraktična je kada se radi o ***kolekciji elemenata različitih tipova***
2. Njihov ***ukupan broj*** ni približno ***nije poznat***
   - Ovaj ***problem se prevazilazi*** zadavanjem neke vrednosti koja predstavlja ***maksimalnu dozvoljenu dužinu*** niza
   - Negativan efekat na efikasno ***iskorišćavanje memorijskog prostora***

Nizovi

- Niz kao struktura podataka ima i određene ***nedostatke***

3. Operacije ***dodavanja*** ili ***uklanjanja elemenata*** sa pozicija ***koje nisu prva i poslednja*** – ***nisu efikasne***, jer dovode do ***pomeranja svih elemenata*** u nizu od date lokacije do kraja

Realizacija polinoma

- ***Polinom p(x)*** je, matematički gledano, složeni ***izraz oblika***:

$$p(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3 + \ldots\ldots\ldots + p_n x^n, \quad n \neq 0$$

- izrazi $p_i x^i$, (i=0,…, n) se nazivaju ***monomi polnoma***,
- p_i su ***koeficijenti polinoma***, (***i*** – ***indeks*** koefic.)
- ***n*** je ***stepen polinoma***

Polinom kao ATP

- ***Polinom kao ATP:***
  - Linearna lista ***monoma***
  - Osnovne ***operacije*** nad polinomima
  - Monom ($p_i x^i$) – šta ga ***jednoznačno*** određuje?
    - ***Koeficijent***
    - ***Stepen***
      - Definiše i ***uređenost*** kolekcije monoma

Polinom kao ATP

- ***Koeficijenti***:
- Polinom se naziva ***normalizovanim*** ako je ***koeficijent*** monoma sa najvišim stepenom jednak jedinici (***$p_n = 1$***)
- ***Koeficijenti*** ***polinoma*** su obično ***brojevi*** - najčešće ***celi***, ***prirodni***, ***realni***, …

Polinom kao ATP

- ***Skup osnovnih operacija*** sa polinomima (tj. ***polinomski račun***):
  - Konstruisati polinom, tj. staviti da je polinom jednak nuli,
  - Naći stepen polinoma, (***operacija upita***)
  - Izračunati vrednosti polinoma za neku vrednost promenljive,
  - Štampati polinom,
  - Sabrati (oduzeti) dva polinoma,
  - Pomnožiti polinom konstantom, (***modifikujuća operacija***)
  - Pomnožiti dva polinoma i
  - Podeliti dva polinoma

Realizacija polinoma

- Polinom se kao ***struktura podataka*** može realizovati na ***više načina***
- Realizacija može biti u
- ***Statičkoj*** formi
  - ***Dva pristupa***:
  1. Jednim objektom – podrazumeva korišćenje ***samo nizova***
  2. Kolekcijom objekata - koristi se ***i niz i struktura***, odnosno, ***niz struktura***
- ***Dinamičkoj*** formi
  - Zasniva se pre svega ***na listama*** (pokazivačima)

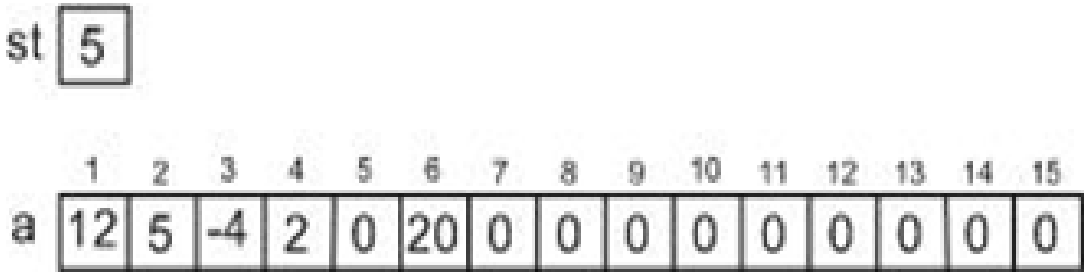
Implementacija polinoma nizom

- ***Najčešći*** vid realizacije polinoma nizom - rešenje u kome se ***ceo polinom predstavlja jednim objektom***
- ***Struktura*** se sastoji od ***dva polja***

1. ***Prvo polje*** (označeno sa ***st***) - sadrži vrednost koja predstavlja ***najveći stepen*** u datom polinomu
   - Vrednost ***n*** iz opšteg izraza za polinom
2. ***Drugo polje*** je ***predstavljeno nizom*** (označeno sa ***a***), vrednosti elemenata - ***koeficijenti*** datog polinoma

Implementacija polinoma nizom

- Na primer, reprezentacija polinoma
  $p(x) = 12 + 5x - 4x^2 + 2x^3 + 20x^5$
  korišćenjem niza:



- Pitanje ***dimenzionisanja niza***
  - Niz je ***statička struktura*** podataka – ***dužina niza*** mora biti definisana ***pre njegovog korišćenja***
  - ***Maksimalan broj monoma***, odnosno koeficijenata koji polinom može da sadrži (na slici = ***15***)

Implementacija polinoma nizom

- Pitanje ***usaglašavanja indeksiranja***
  - ***koeficijenti polinoma*** (odnosno stepena) i
  - načina indeksiranja ***elemenata u nizu*** (u pseudo jeziku koji koristimo)
- Za polinom koji sadrži koeficijente od p_0 do p_n, potrebno je polje niza u strukturi dimenzionisati kao niz sa ***n + 1*** elemenata
- U primeru sa slike, ***maksimalan indeks koeficijenata polinoma*** (***= stepenu polinoma***) koji se može realizovati pomoću ovog objekta je ***14***

Implementacija polinoma nizom

- ***Predpostavke*** u predstojećim algoritmima:
- ***Stepen polinoma*** (maksimalni indeks koeficijenata) tretiramo kao ***globalnu promenljivu*** i označavaćemo je sa ***n***
- Vrednost koeficijenta ***jednaka 0 ako se monom*** tog stepena ***ne pojavljuje*** u polinomu
- Svi koeficijenti u ***nula- polinomu*** su jednaki 0

Implementacija polinoma nizom

- *Konstruisanje praznog polinoma*
- Konstruisanje prazne liste se sastoji od inicijalizacije polja maksimalne vrednosti stepena sa ***-1***:

```
// Ulaz: polinom p

// Izlaz: nula polinom p

algorithm poli-make(p)

        p.st = -1;

        for i = 1 to n + 1 do

            p. a[i] = 0;

        return p;
```

Implementacija polinoma nizom

- *Sabiranje i oduzimanje polinoma*
- Sabiranje i oduzimanje polinoma se može jednostavno realizovati ***jedinstvenim algoritmom*** jer je postupak vrlo sličan
- Pri sabiranju (oduzimanju) polinoma ***koeficijenti u monomima istog stepena se sabiraju*** (oduzimaju)
- ***Ključno mesto u algoritmu*** je ***određivanje stepena rezultujućeg polinoma***
- Moguće je da stepen rezultujućeg polinoma bude ***manji*** od stepena sabiraka

Implementacija polinoma nizom

- *Sabiranje i oduzimanje polinoma*
- ***Ulazni parametri***: polinom ***p1*** i polinom ***p2*** koje sabiramo (oduzimamo)
- Da bi se realizovao ***jedinstveni algoritam***

1. Uvodimo ***dodatni ulazni parametar op*** čija vrednost predstavlja ***oznaku za operaciju*** koju treba izvršiti
2. Vrednost promenljive ***op*** može biti ***plus*** ili ***minus***

Implementacija polinoma nizom

- *Sabiranje i oduzimanje polinoma*

```
// Ulaz: polinomi p1 i p2, oznaka operacije op
// Izlaz: polinom rez kao rezultat operacije op nad polinomima p1 i p2
algorithm poli-sab(p1, p2, op)

        poli-make(rez);
        if (p1.st > p2.st)  then
            rez.st = p1.st;
        else
            rez.st = p2.st;

        if (op == plus)  then
            for i = 1 to rez.st + 1 do
                rez. a[i] = p1. a[i] + p2. a[i];
         else
            for i = 1 to rez.st + 1 do
                rez. a[i] = p1. a[i] - p2. a[i];

         return rez;
```

Implementacija polinoma nizom

- *Evaluacija polinoma*
- ***Izračunavanje vrednosti*** polinoma za neku vrednost promenljive ***x***
- Posmatrajmo sledeći polinom,

$$-10x^7 + 4x^5 + 3x^2$$

za vrednost $x = 1$,

$$= -10*1^7 + 4*1^5 + 3*1^2$$

$$= -10 + 4 + 3 = -3$$

što predstavlja rezultat evaluacije polinoma

Implementacija polinoma nizom

- *Evaluacija polinoma*
- U opštem slučaju,
- Postoji ***više načina*** kako doći do rezultata evaluacije polinoma
- Najjednostavniji način izračunavanja vrednosti polinoma je pomoću ***Horner-ove šeme***:

$$p(k) = (\ldots((p_n k + p_{n-1})k + p_{n-2} \ldots + p_2)k + p_1)k + p_0$$

Implementacija polinoma nizom

- *Evaluacija polinoma*

```
// Ulaz: vrednost promenljive x, polinom p

// Izlaz: rezultat evaluacije polinoma p

algorithm eval(x, p)

    if (p.st == -1)  then

          return  0;        // Polinom je prazan

     else

          rezultat = p.a[p.st + 1];

          for i = p.st downto 1 do

               rezultat = rezultat * x + p.a[i];

     return rezultat;
```

Implementacija polinoma nizom struktura

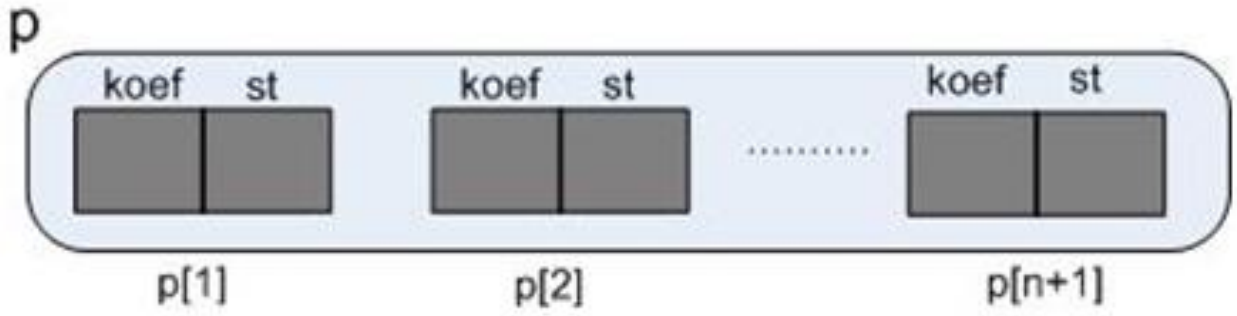
- U prethodnoj realizaciji - ***polinom*** se implementira ***kroz jedan objekat***
- U ovoj realizaciji - ***polinom*** se implementira ***nizom objekata***
- ***Svaki element*** niza predstavlja se ***objektom*** sa ***dva polja***
- Ovaj ***objekat*** (struktura) se ***naziva monom*** jer se njime implementiraju monomi datog polinoma

Implementacija polinoma nizom struktura

- ***Objekat*** se naziva monom
- ***Prvo polje*** (sa oznakom ***koef***) predstavlja ***koeficijent monoma***
- ***Drugo polje*** (označeno sa ***st***) predstavlja ***stepen monoma***
- ***Dimenzionisanje*** ovog ***niza*** struktura - važe identične napomene ***kao i u predhodnoj realizaciji*** polinoma
- ***Vrednost polja*** u strukturi koja po redosledu odgovara nepostojećem monomu jednaka su ***nuli***

Implementacija polinoma nizom struktura

- Implementacija polinoma u opštem obliku



- ***Sve operacije*** koje čine polinomski račun mogu se ***vrlo jednostavno algoritamski rešiti*** i u ovoj implementaciji polinoma

Implementacija polinoma pokazivačima

- ***Glavni nedostaci*** izloženih realizacija polinoma baziranih na nizovima ***proizilaze iz*** same strukture i ***svajstava nizova*** kao strukture podataka
- Šta ako se pokaže ***potreba*** za polinomom koji je višeg stepena ***od maksimalno dozvoljene*** vrednosti?
- Šta će se dešavati u slučalevima primene ***„retkih“ polinoma*** – polinoma sa velikim stepenom i malim brojem koeficijenata različitih od nule?

Implementacija polinoma pokazivačima

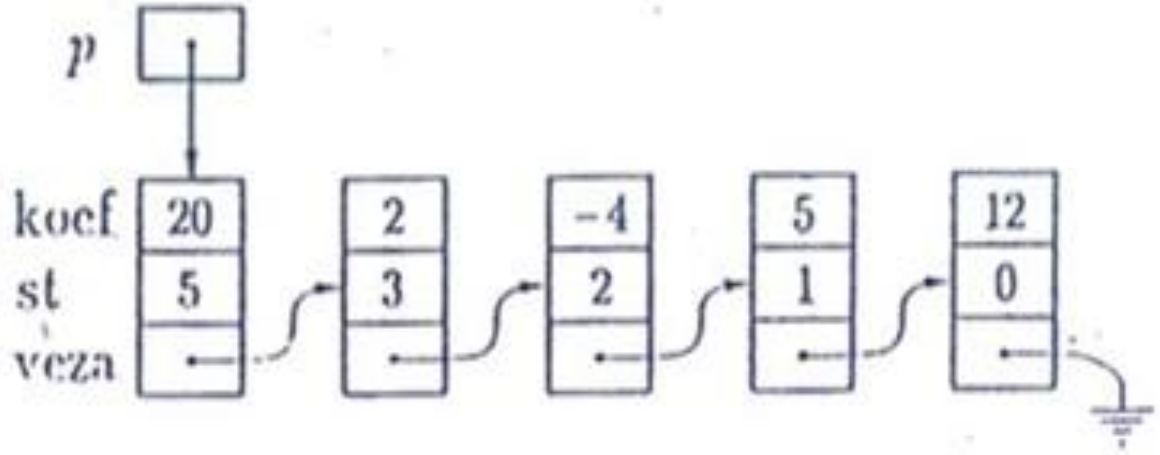
- Kako ***otkloniti*** ove ***nedostatke***?
- ***Promenom strukture monoma***
  - Uvodi se ***treće*** (pokazivačko) ***polje*** (nazvano ***veza***) - čija vrednost ***ukazuje na sledeći monom*** u redosledu monoma datog polinoma
- Time dobijamo ***dinamičku strukturu*** čime se gubi potreba za nizovima kao načinom za ***očuvanje redosleda monoma*** u polinomu

Implementacija polinoma pokazivačima

- Na primer, reprezentacija polinoma

$$p(x) = 12 + 5x - 4x^2 + 2x^3 + 20x^5$$

korišćenjem dodatnog pokazivačkog polja:



Implementacija polinoma pokazivačima

- Deklaracioni iskazi za polinom ***u C jeziku***:

```
typedef struct monom
    {
    int  koef;
    int st;
    struct monom  *veza;
     }
 struct monom  *p;
```

Implementacija polinoma pokazivačima

- U realizaciji se koristi i ***jedan spoljašnji pokazivač*** (još se naziva i ***pristupni pokazivač***) koji ukazuje na objekat – na strukturu monoma sa ***najvećim stepenom***
- Svaka ***struktura monoma ukazuje*** na ***sledeći monom*** u redosledu
- ***Poslednji monom*** (sa najmanjim stepenom) poseduje ***pokazivačko polje*** sa vrednošću ***null***

Implementacija polinoma pokazivačima

- ***Ova realizacija*** polinoma sa korišćenjem pokazivača je samo ***uvod za ono što sledi*** u sledećim predavanjima koja se bave ***dinamičkim strukturama podataka***
- Ova implementacija je ***tipičan primer korišćenja povezane liste*** o čemu će više reči biti na sledećem času