Strukture Podataka i Algoritmi

Lekcija 5

leto 2019/2020

Prof. dr Branimir M. Trenkić

Stekovi

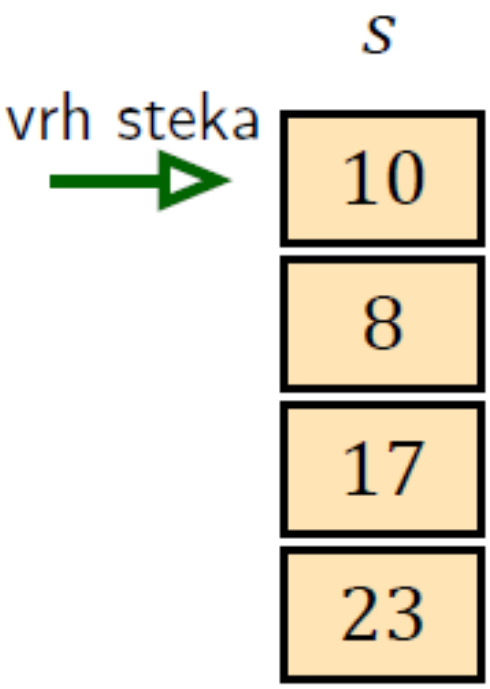
- ***Stek*** (engl. ***stack***)
- Specijalna ***vrsta liste***
- Operacije ***dodavanja*** ili ***uklanjanja*** elemenata obavlja se samo ***na jednom kraju*** - koji se naziva ***vrh (top) steka***
- Drugi kraj steka se naziva – ***dno (bottom)***
- Intuitivni model strukture podataka steka:
  - Naslagani čisti ***tanjiri u restoranu*** ili
  - Komadići mesa poređani na ***ražnjiću***

Stekovi

- Struktura podataka organizovana po ***principu LIFO*** (*Last-In-First-Out*)- ***poredak*** elemenata po kojem se oni ***uklanjaju*** sa steka je ***obrnut*** onom po kojem su oni ***dodati*** na stek
- Pogodno ukloniti samo onaj element (***tanjir*** ili ***komadić mesa***) koji se nalazi na vrhu steka
- Operacija ***dodavanja*** elemenata na stek – ***push***
- Operacija ***uklanjanja*** elemenata iz steka zove se ***pop***

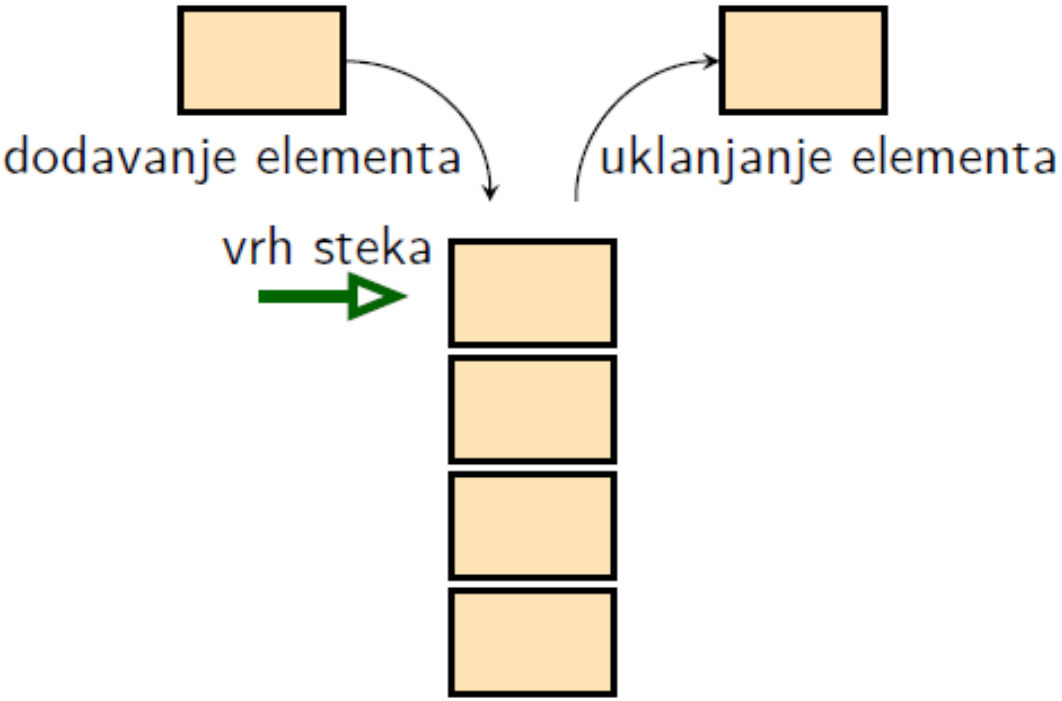
Stekovi

- Stek – niz elemenata poređanih ***"jedan na drugi"***
- Primer:



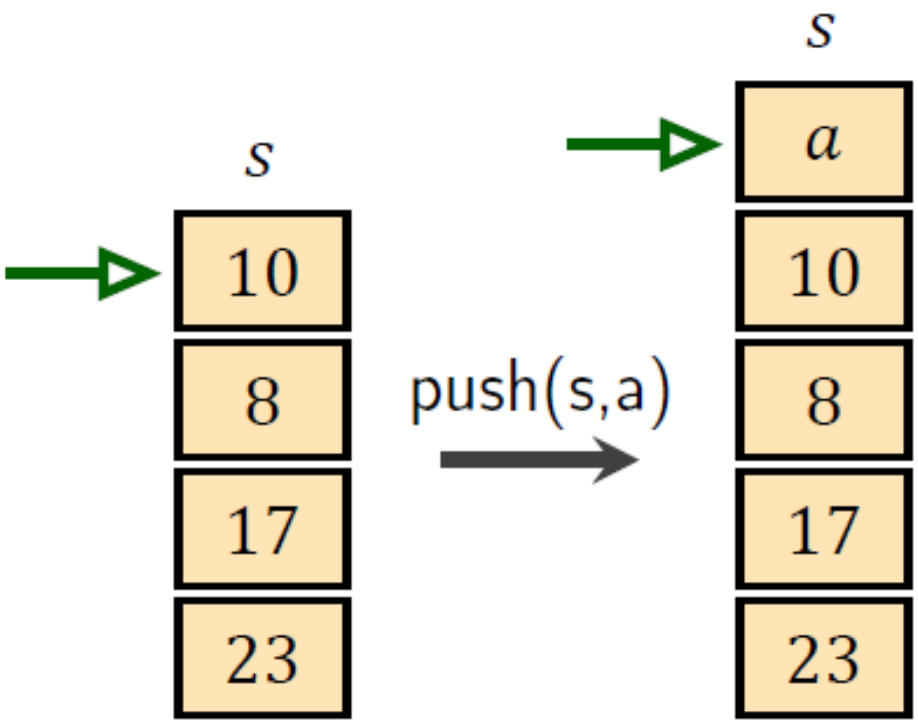
Stekovi

- Rad sa stekom je moguć samo preko jedne ***pristupne tačke*** **– *vrha steka***
- Dodavanje i uklanjanje elemenata:



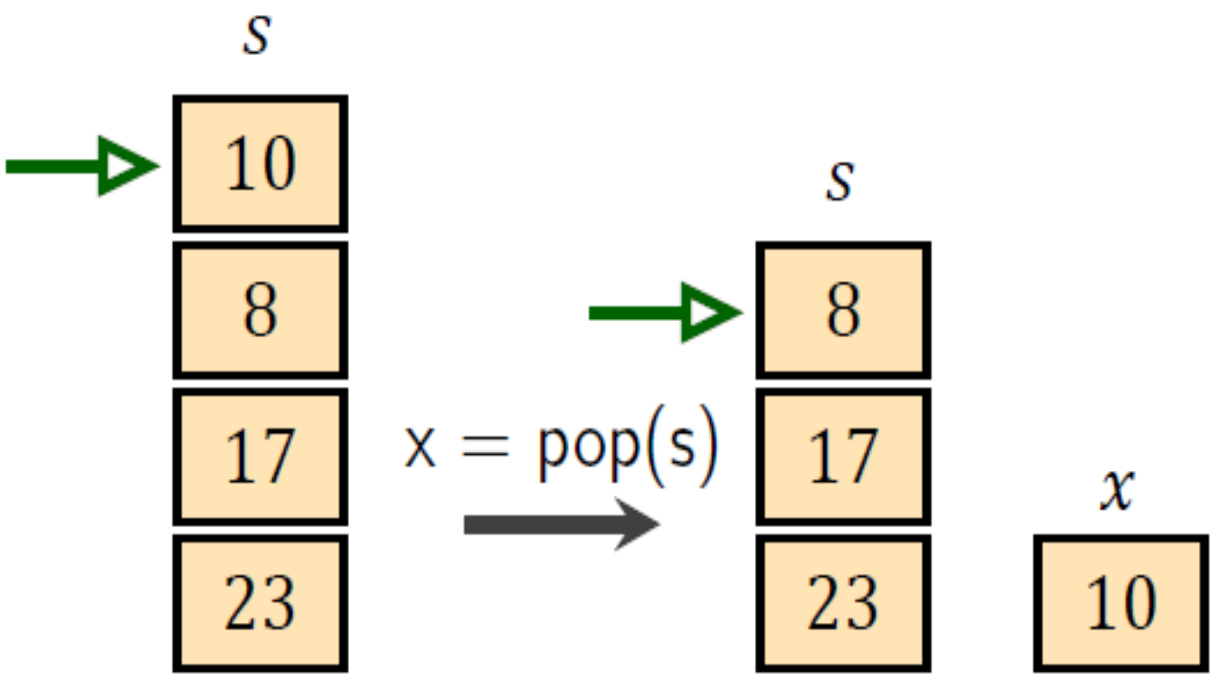
Stekovi

- ***Dodavanje*** elementa na stek (***push***)



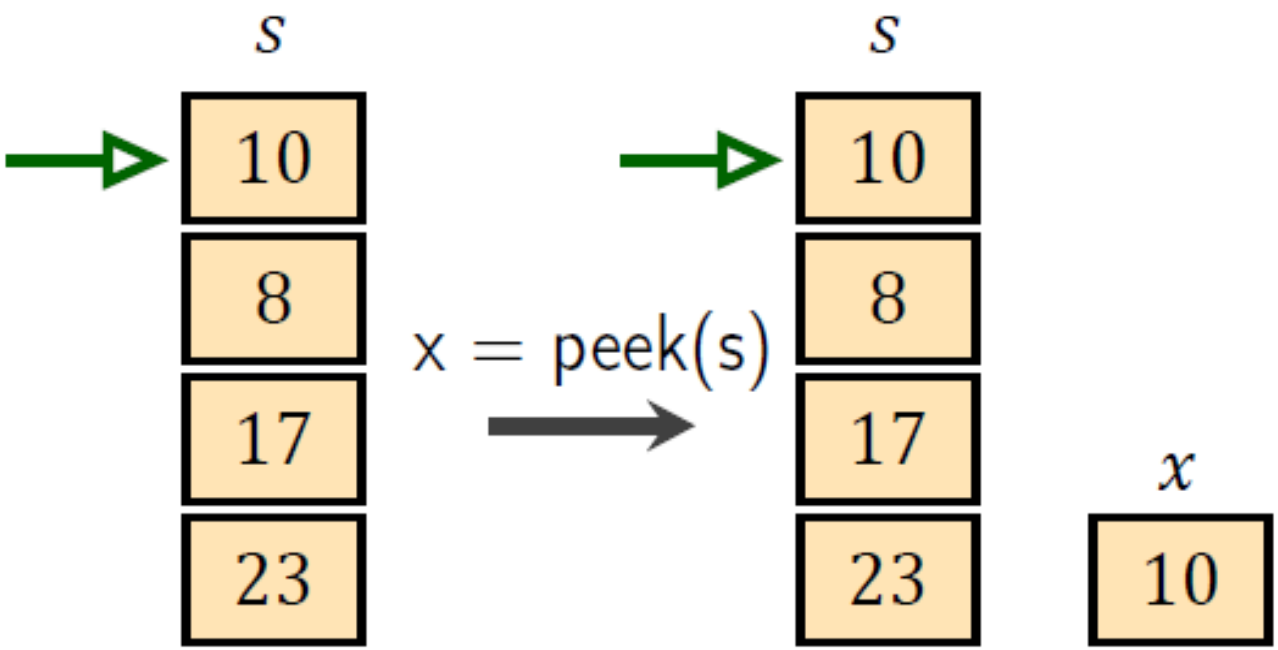
Stekovi

- ***Uklanjanje*** elementa iz steka (***pop***)



Stekovi

- ***Vrednost*** elementa na vrhu steka (***peek***)



Stekovi

- ***Primena*** stekova:
  - Operacija “undo” u programima,
  - Nalaženje izlaza iz lavirinta u igrama
  - Izračunavanje izraza
  - .....

Stekovi

- Izračunavanje izraza u ***infiks*** notaciji (zapisu):

operand operator operand operator operand

4 + 5 * 2

- Binarni operator ***između*** operanada
- ***Pravila prioriteta*** za operatore
- ***Zagradama*** se menja prioritet
- 4 + (5 * 2) = 14
- (4 + 5) * 2 = 18

Stekovi

- Ekvivalentni ***postfiks*** (***obrnuta poljska notacija***) zapis izraza:

operand	operand	operand	operator	operator
4	***5***	***2***	*******	***+***

a)Pravila prioriteta za operatore ***nisu potrebna***

b)Zagrade nisu potrebne

Stekovi

- ***Efikasan algoritam*** za izračunavanje ***postfiks izraza***:
- Analizirati elemente izraza redom ***sleva na desno***:

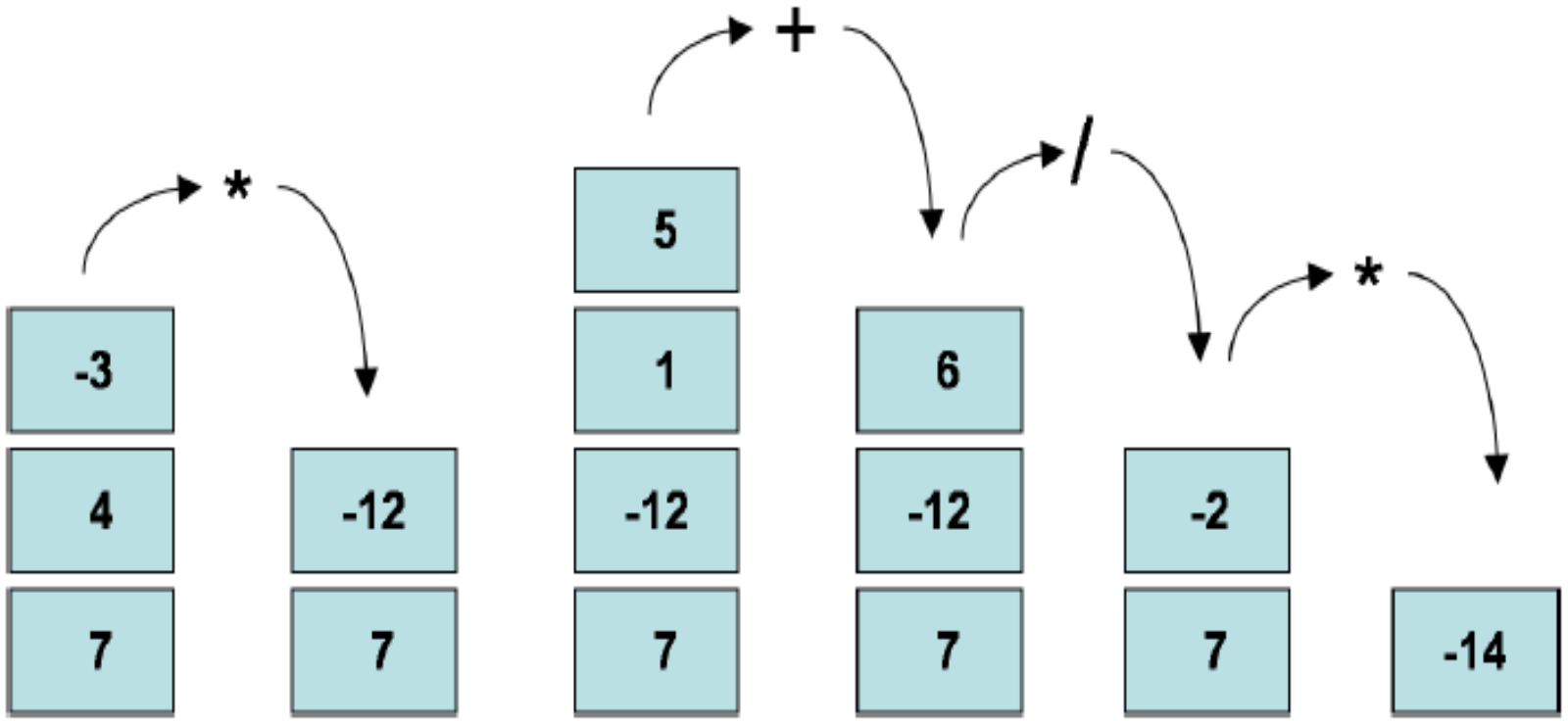
A. Svaki operand na koji se naiđe - ***stavlja se na stek***

B. Operator se izvršava nad dva elementa (ili jedan) na vrhu steka i ***rezultat operacije se stavlja na stek***

Stekovi

- ***Primer***:

7 4 -3 * 1 5 + / * **[7*(4*(-3))/(1+5)]**



Implementacija Steka

A. ***Dinamička*** (pomoću ***povezane liste***)
B. ***Statička*** (pomoću ***niza***)

Implementacija Steka

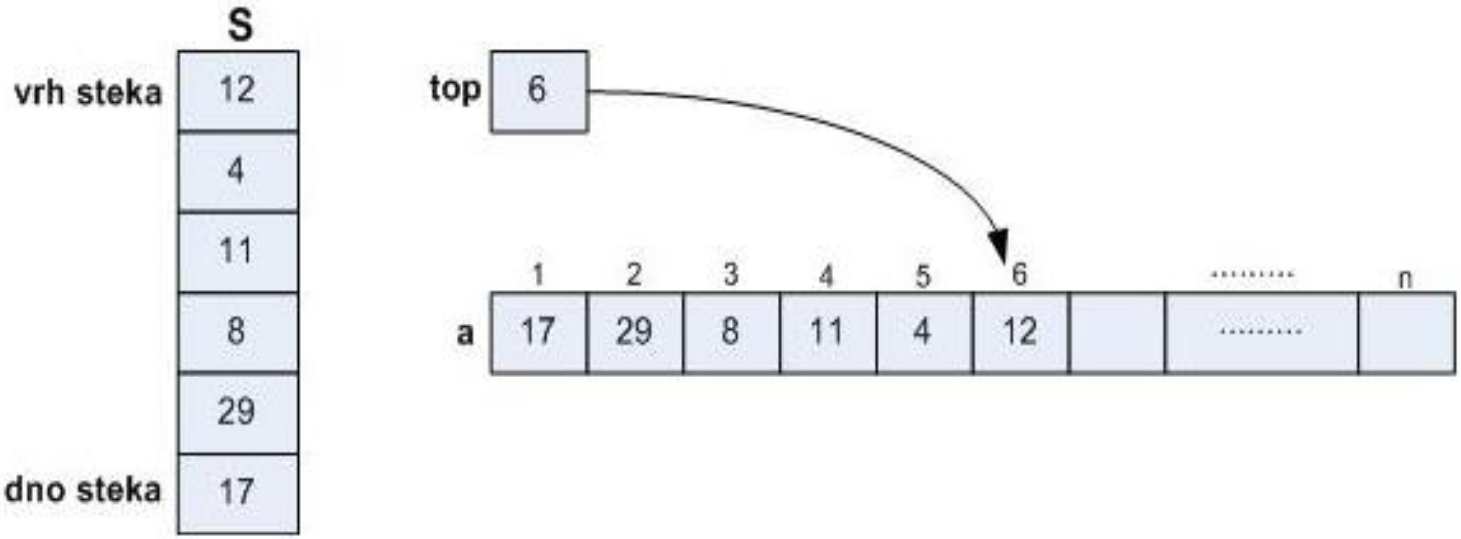
- Dinamička (pomoću liste)
- Sve ***implementacije liste*** o kojima smo govorili - ***mogu se iskoristiti*** i za stek
- Ako je omogućeno korišćenje ***pokazivača*** - implementacija liste praktično ***ne mora da se menja*** za implementaciju steka
  - Operacija za ***dodavanje*** elemenata na ***početak liste*** (***push***) i
  - Operacija ***uklanjanja*** elemenata sa ***početka liste*** (***pop***)

Implementacija Steka

- Implementacija steka pomoću niza
- Predstavljanje steka pomoću ***jednog objekta*** koji se sastoji od ***dva polja***
- ***Prvo polje*** (***top***) - sadrži indeks elementa niza koji ***trenutno*** predstavlja ***vrh steka***
  - Ovo polje se još naziva i ***pokazivač steka*** (*stack pointer*)
- ***Drugo polje*** predstavlja niz (***a***) dužine ***n***, u kome se nalazi sadržaj steka

Implementacija Steka

- Implementacija steka pomoću niza



- ***Dno steka*** je ***vezano*** za ***početnu adresu*** niza ***a*** i stek raste ka višim adresama
- Stek se sastoji od elemenata ***a[1], ...,a[top]***

Implementacija Steka

- Implementacija steka pomoću niza
- Ukoliko je ***top = 0*** - kaže se da je ***stek prazan***
- Ako se greškom uklanja element iz praznog steka - ***potkoračenje steka*** (engl. ***underflow***)
- Ukoliko je ***top = n***, ***stek je pun***
- Ako se greškom dodaje novi element u puni stek - ***prekoračenje steka*** (engl. ***overflow***)

Operacije nad stekom

- ***Kreiranje steka***. Podrazumeva ***rezervisanje prostora za niz*** date ***veličine n***, pokazivač steka se inicijalizuje na 0 (***stack-make()***)

```
// Ulaz: stek S

// Izlaz: početno stanje steka S

algorithm stack-make(S)

    S.top = 0; return;
```

Operacije nad stekom

- ***Provera da li je stek prazan***. Vraća logičku vrednost “tačno“ ako je pokazivač steka na nuli, inače vraća “netačno“ (***stack-empty()***)

```
// Ulaz: stek S

// Izlaz: tačno ako je S prazan, inače netačno

algorithm stack-empty(S)

    if (S.top == 0)

        return true;

    else

        return false;
```

Operacije nad stekom

- ***Provera da li je stek pun***. Vraća logičku vrednost “tačno“ ako je pokazivač steka jednak n, inače vraća “netačno“ (***stack-full()***)

```
// Ulaz: stek S

// Izlaz: tačno ako je s pun, inače netačno

algorithm stack-full(S)

    if (S.top == n)

        return true;

    else

        return false;
```

Operacije nad stekom

- ***Dodavanje novog elementa***. Operacija dodavanja novog elementa sa vrednošću ***x*** na stek *S* (***push()***)

```
// Ulaz: novi element x, stek S
// Izlaz: stek S sa elementom x na vrhu
algorithm push(x, S)
    if (stack-full(S) == true) then
        return "overflow"    // Stek je pun – prekoračenje kapaciteta steak
    else
        S.top = S.top + 1;    // Stek nije pun
        S.a[S.top] = x;
        return;
```

Operacije nad stekom

- ***Uklanjanje elementa***. Algoritam ***pop()*** uklanja i vraća vrednost elementa sa steka *S*

```
// Ulaz: stek S

// Izlaz: element x uklonjen sa vrha steak S

algorithm pop( S)

    if (stack-empty(S) == true) then

        return “underflow”     // Stek je prazan – potkoračenje steka

    else

        x = S.a[S.top];

        S.top = S.top - 1;

        return x;
```