Napredni I/O (Ulaz/Izlaz)

- **Biblioteke**
  - **<stdio.h>,**
  - **<conio.h>,**
  - **<stdlib.h>.**
- **Neki od primjera**
- **Provjera da li je pritisnut taster na tastaturi (bez učitavanja tastera): conio.h**

```
{if(kbhit())
 {printf("Neki taster je pritisnut ali nije ucitan");
 }
}
```

Napredni I/O (Ulaz/Izlaz)

- **Učitavanje pritisnutog tastera sa tastature: <conio.h>**

  {**char** taster;

  taster=***getch();***

  }
- **Učitavanje stringa sa tastature: <stdio.h>**

  {**char** tekst[50];

  gets(tekst);

  }

Napredni I/O (Ulaz/Izlaz)

- **Ispisivanje znaka na ekran: <stdio.h>**

  {**char** taster='A';

  putch(taster);

  }

- **Ispisvanje stringa na ekran: <stdio.h>**

  {**char** tekst[50]="Pozdrav";

  puts(tekst);

  }

Napredni I/O (Ulaz/Izlaz)

- **Učitavanje cijelog broja sa tastature korišćenjem stringa:**

```
{char buffer[50];
int i;
gets(buffer); /* <stdio.h> */
i=atoi(buffer); /* <stdlib.h> */
}
```

- **Učitavanje velikog (long) cijelog broja sa tastature korišćenjem stringa:**

```
{char buffer[50];
long l;
gets(buffer); /* <stdio.h> */
l=atol(buffer); /* <stdlib.h> */
}
```

Napredni I/O (Ulaz/Izlaz)

- **Učitavanje realnog broja sa tastature korišćenjem stringa:**

```
{char buffer[50];
float f;
gets(buffer); /* <stdio.h> */
f=atof(buffer); /* <stdlib.h> */
}
```

- **Formatirani upis u string i ispisivanje stringa na ekran:**

```
/*<stdio.h> */
{char buffer[50];
int i=101;
float f=123.456;
sprintf(buffer," --- %d --- %f ---",i,f);
puts(buffer);
}
```

Napredni I/O (Ulaz/Izlaz)

FUNKCIJE IZ BIBLIOTEKE "CONIO.H "	
cgets	čita string sa konzole
clreol	briše tekst od tekuće pozicije kursora do kraja linije u tekstualnom prozoru
clrscr clrscr	briše prozor u tekstualnom modu
cprintf	ispisuje formatirani izlaz u tekstualni prozor
cputs	ispisuje string u tekstualni prozor
cscanf	uzima formatirani ulaz sa konzole
delline	briše liniju u tekstualnom prozoru
getch	uzima kod tastera sa konzole (ne prikazuje ga)
getche	uzima kod tastera sa konzole (prikazuje ga)
getpass	učitava password
gettext	kopira tekst iz tekstualnog prozora u memoriju

conio.h (nastavak)

gettextinfo	daje tekst video informacije
gotoxy	pozicionira kursor u tekstualnom prozoru
highvideo	postavlja visok intezitet prikaza teksta
insline	umeće liniju na mjestu kursora
kbhit	provjerava dali je taster pritisnut
lowvideo	postavlja nizak intezitet prikaza teksta
movetext	kopira tekst prozor sa jednog na drugu poziciju
normvideo	postavlja normalan intezitet prikaza teksta
putch	štampa karakter na ekran
puttext	kopira tekst iz memorije na ekran
textattr	postavlja tekst atribute za "tekst funkcije"
textbackground	postavlja novu boju pozadine
textcolor	postavlja novu boju slova
textmode	mjenja tekstualni mod prikaza
ungetch	vraća taster nazad na konzolu
wherex	vraća apsulutnu x poziciju kursora
wherey	vraća apsulutnu y poziciju kursora
window	definiše aktivni prozor u tekstualnom modu

Ulaz/ izlaz podataka u C programima

- **Dva osnovna načina:**
- **Na sistemskom (nižem)**
  - open(), close(),
  - read(), write()...
- **i na višem ulazno/izlaznog toka**
  - scanf(), printf(),
  - getchar(), putchar(),
  - gets(), puts()...

Ulaz/ izlaz podataka u C programima

- **Ulaz / izlaz na sistemskom nivou**
  - vezan za detalje ulazno / izlaznog uređaja,
  - nema baferovanja ni formatiranja,
  - poziva direktno ulazno / izlazne mogućnosti operativnog sistema,
  - nije standardizovan način u jeziku C,
  - **ne preporučuje se.**

Ulaz/ izlaz podataka u C programima

- **Ulaz / izlaz na nivou ulazno / izlaznog toka**
  - ne zavisi od detalja ulazno / izlaznog uređaja,
  - zbog veće efikasnosti prenosa podataka u oba smera,automatski se realizuje baferovanje,
  - omogućava da program bude prenosiv iz jednog operativnog sistema u drugi,
  - standardizovan je način u jeziku C,
  - **koristi se.**

Ulaz/ izlaz podataka u C programima

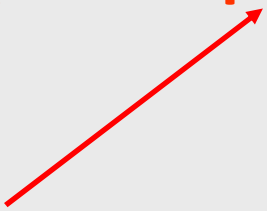
- **Ulazno / izlazni tokovi (streamovi)**
- Sve funkcije višeg nivoa ulaza / izlaza koriste ulazno / izlazne tokove.
- Ulazno / izlazni tok (stream) je sekvenca bajtova:
  - ulazni tok (input stream) – sa ulaznog uređaja.
  - izlazni tok (output stream) – na izlazni uređaj.
- Ulazno / izlazni tok može se smatrati komunikacionim kanalom između ulazno / izlaznog uređaja i ulaza / izlaza procesa programa.

Ulaz/ izlaz podataka u C programima

- **Funkcije koje traže definisanje streamova, stdio.h.**
- **Prefiks f (file) u nazivu.**
- **fscanf()**
- **fprintf()**
- **fgetc()**
- **fputc()**
- **fgets()**
- **fputs()**
- **Uglavnom za rad sa datotekama**

Ulaz/ izlaz podataka u C programima

- **Funkcija za formatirani ulaz**
- **int scanf(const char *format, void *ptr, . . .);**
- **int fscanf(FILE *fptr, const char *format, void *ptr, . . .);**



- **Razlika samo u jednom argumentu.**

Ulaz/ izlaz podataka u C programima

- **Funkcija za formatirani izlaz**

- **int printf(const char *format, . . .);**
- **int fprintf(FILE *fptr, const char *format, . . .);**

- **Razlika samo u jednom argumentu.**

Ulaz/ izlaz podataka u C programima

- **Funkcije za ulaz karakter po karakter**
- int getchar(void);

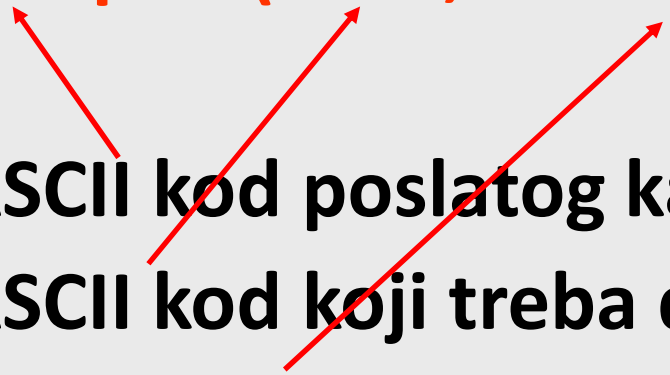
- **int fgetc(FILE *fptr);**

- **ASCII kod procitanog kar. ili EOF(-1).**
- **Dodat argument.**

Ulaz/ izlaz podataka u C programima

- **Funkcija za izlaz karakter po karakter**
- int putchar(int c);

- **int fputc(int c, FILE *fptr);**



- **ASCII kod poslatog karaktera ili EOF(-1).**
- **ASCII kod koji treba da se pošalje.**
- **Dodat argument.**

Ulaz/ izlaz podataka u C programima

- **Funkcija za ulaz red po red**
- **char *gets(char *s);**

- **char *fgets(char *s, int MAX+1 , FILE *fptr);**

- **Adresa stringa koji čuva učitani red ili NULL.**
- **Adresa stringa koji treba da čuva učitani red.**
- **Dodati argumenti, može biti ograničen broj znakova koji čita iz reda na MAX .**
- **Dodatni argument.**

Ulaz/ izlaz podataka u C programima

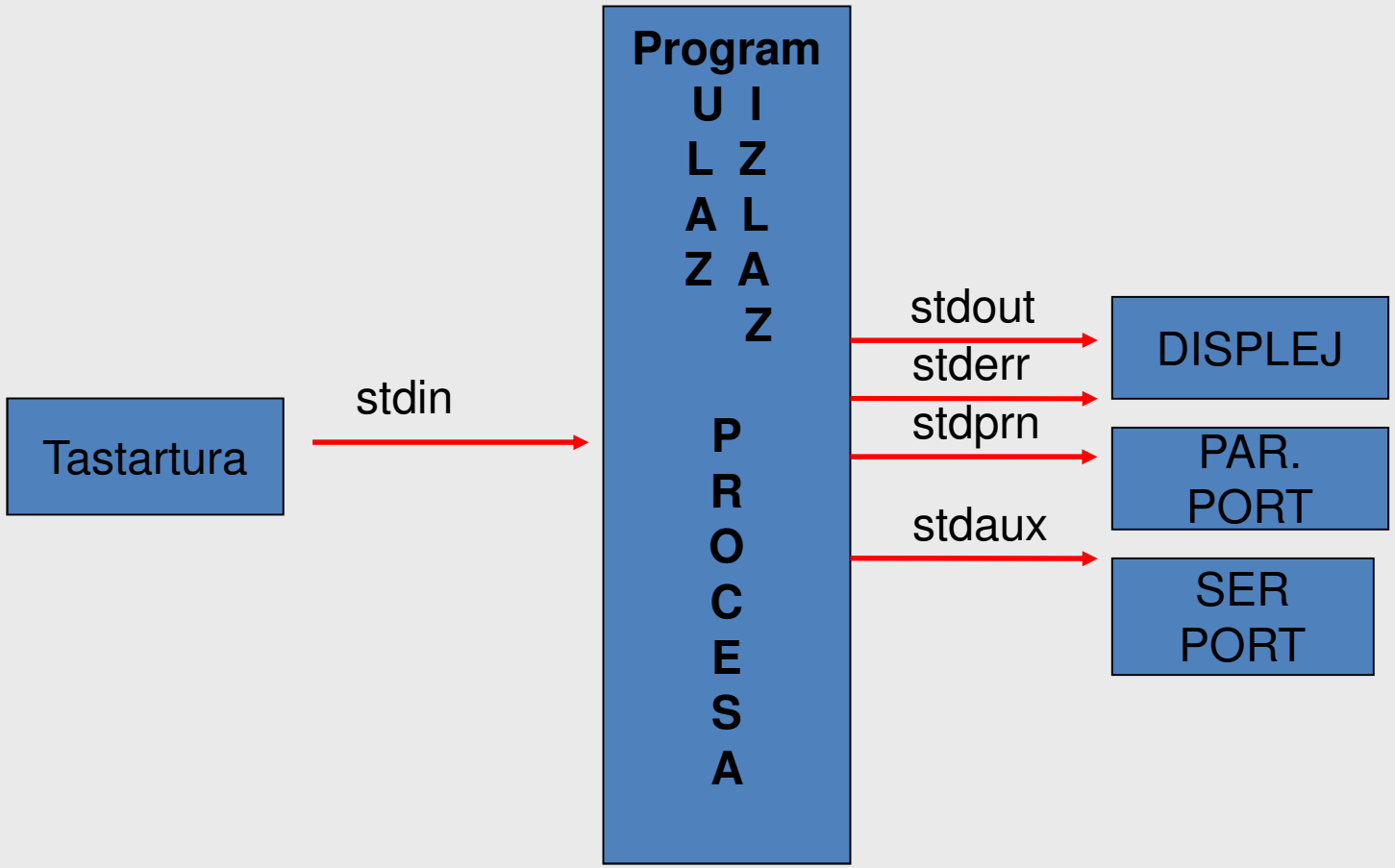
- **Funkcija za izlaz red po red**
- int puts(char *s);

- **int fputs(char *s, FILE *fptr);**

- **Broj uspješno poslatih znakova ili EOF(-1).**
- **Adresa stringa čiji sadržaj treba da pošalje.**
- **Dodat argument.**

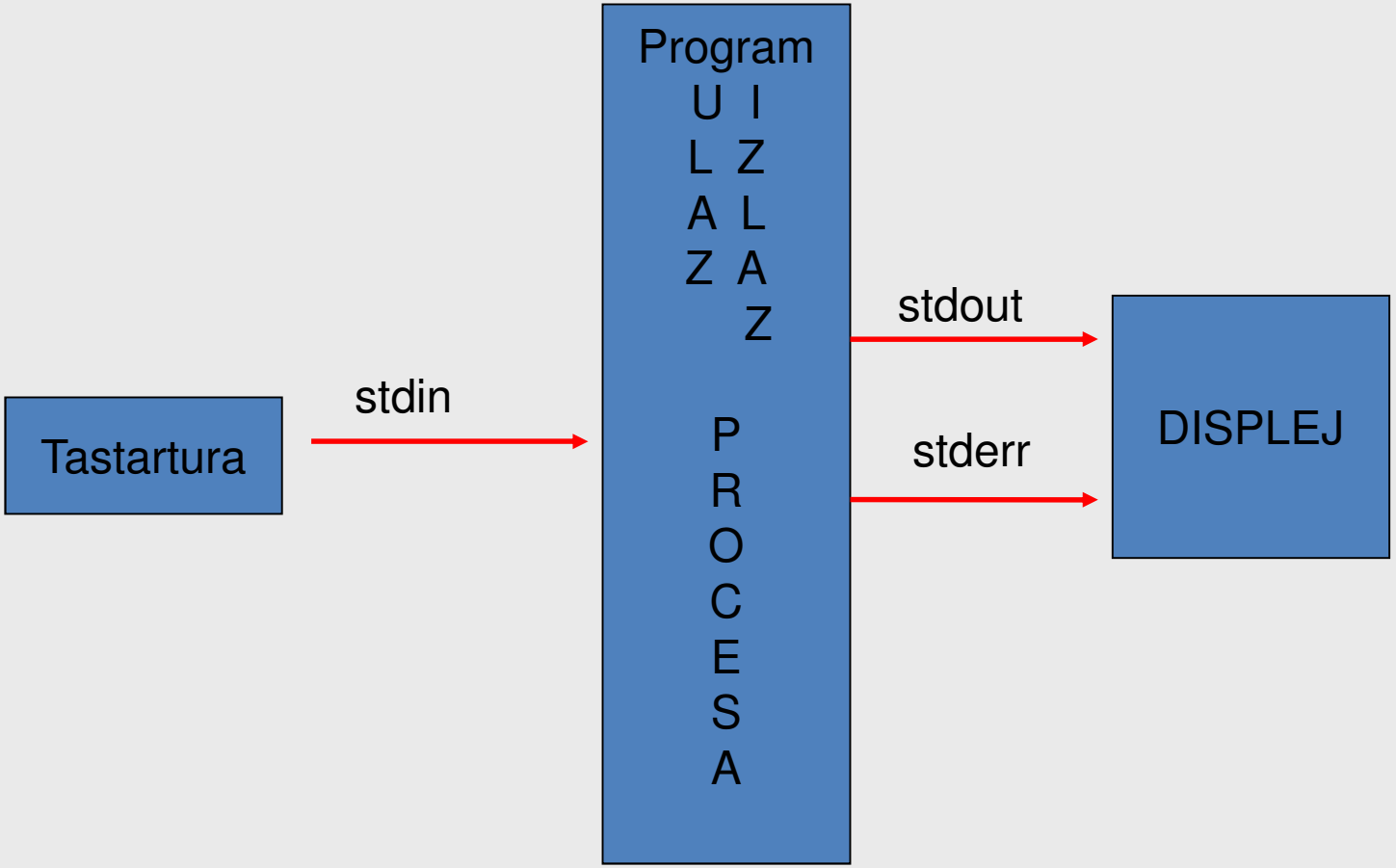
Ulaz/ izlaz podataka u C programima

- **DOS model ulazno / izlaznih tokova**



Ulaz/ izlaz podataka u C programima

- **UNIX / LINUX model ulazno / izlaznih tokova**



Ulaz/ izlaz podataka u C programima

- **Glavni standardni ulazni tok**
- **stdin (pokazivač na strukturu I/O tipa)**
  - oznaka za glavni standardni ulazni tok.
  - povezuje tastaturu i ulaz procesa programa.
  - moguće preusmeravanje na datoteku ako komanda operativnog sistema to kaže.
- **Po ANSI standardu, automatski se otvara na početku izvršavanja programa.**

Ulaz/ izlaz podataka u C programima

- **Glavni standardni izlazni tok**
- **stdout (pokazivač na strukturu I/O tipa)**
  - oznaka za glavni standardni izlazni tok.
  - povezuje izlaz procesa programa i ekran.
  - moguće preusmeravanje na datoteku ako komanda operativnog sistema to kaže.
- **Po ANSI standardu, automatski se otvara na početku izvršavanja programa.**

Ulaz/ izlaz podataka u C programima

- **Standardni izlazni tok za greške.**
- **stderr (pokazivač na strukturu I/O tipa)**
  - oznaka za standardni izlazni tok.
  - povezuje izlaz procesa programa i ekran.
  - nije moguće preusmeravanje na datoteku i zbog toga se koristi za poruke o greškama i statuse.
- **Po ANSI standardu, automatski se otvara na početku izvršavanja programa.**

Ulaz/ izlaz podataka u C programima

- **Karakteristike standardnih tokova**
- **stdin - baferovan ulazni tok**
  - **proces ne prima ništa sve dok se ne potvrdi unos preko,tastature pomoću tastera Enter.**
- **stdout – baferovan izlazni tok**
  - izlazni podaci baferuju se u redove, zbog boljih performansi.
- **stderr - nije baferovan izlazni tok**
  - da bi se izbeglo zadržavanje poruka o greškama.

Još o podacima, konstantama i mehanizmima

- **Nabrajanje**

Primjenom ključne riječi *enum* može se definisati ime za skup nabrojanih konstanti.
Na primjer:
enum logicki {NETACNI, TACNO};
enum bool {FALSE, TRUE};
NETACNO i FALSE automatski dobija vrijednost 0 a TACNO i TRUE 1
Nakon ovoga može se deklarisari promjenljiva tipa logički i funkcije koje vraćaju ovakve vrijednosti,
na primjer:
enum logicki prekidac_otvoren;
enum logicki prekidac_zatvoren(int);
Promjenljivoj prekidac_otvoren može se dodijeliti vrijednost
NETACNO (0)ili TACNO (1).
Tako|e finkcija prekidac_zatvoren(int) mo`e da vrati jednu od ove dvije vrijednosti.
Kombinovanjem enum i typedef mogu se napraviti tipovi podataka sa `eljenim imenom.
Na primjer:

Nabrajanje

Nakon ovoga može se deklarisari promjenljiva tipa LOGIC i funkcije koje vraćaju ovakve vrijednosti, na primjer:

```
LOGIC prekidac_otvoren;
LOGIC prekidac_zatvoren(int);
```

Članovi liste nabrojanih konstanti dobijaju rastuće cjelobrojne vrijednosti počevši od 0. Medjutim, moguće je za bilo koju od nabrojanih konstanti specificirati vrijednost, uključujući i negativnu. Ostale konstante za koje vrijednost nije specificirana u listi ima vrijednost za jedan veću od poslednje specificirane vrijednosti.

Na primjer:

```
typedef enum {MANJE=-1, JEDNAKO, VECE} UPOREDI;
```

Nabrajanje

```
enum {Ponedjeljak=1,Utorak,Srijeda,Cetvrtak,Petak,Subota,Nedjelja};
int main()
{int dan;
 printf("Unesi redni broj dana u sedmici: "); scanf("%d",&dan);
switch(dan)
{
case Ponedjeljak : printf("Prvi dan u sedmici");break;
case Utorak      : printf("Drugi dan u sedmici");break;
case Srijeda     : printf("Treci dan u sedmici");break;
case Cetvrtak    : printf("Cetvrti dan u sedmici");break;
case Petak       : printf("Peti dan u sedmici");break;
case Subota      : printf("Sesti dan u sedmici");break;
case Nedjelja    : printf("Sedmi dan u sedmici");break;
default          : printf("Nemoguce...");break;
}
}
```

Nabrajanje

ili

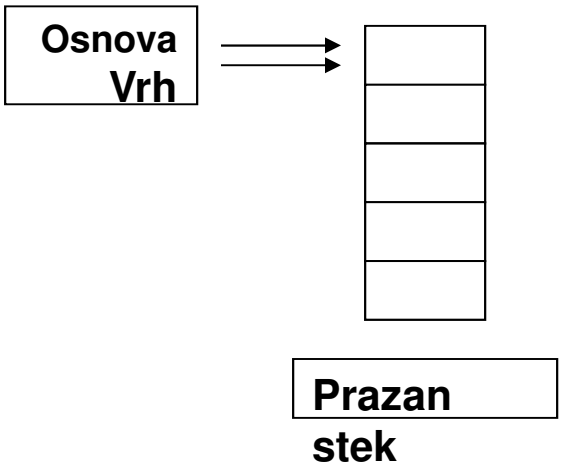
```
enum {Ponedjeljak=1,Utorak,Srijeda,Cetvrtak,Petak,Subota,Nedjelja};
int main()
{int dan;
 printf("Unesi redni broj dana u sedmici: "); scanf("%d",&dan);
  switch(dan)
    {
      case 1 : printf("Prvi dan u sedmici je Ponedeljak");break;
      case 2 : printf("Drugi dan u sedmici je Utorak");break;
      case 3 : printf("Treci dan u sedmici je Srijeda");break;
      case 4 : printf("Cetvrti dan u sedmici je Cetvrtak");break;
      case 5 : printf("Peti dan u sedmici je Petak");break;
      case 6 : printf("Sesti dan u sedmici je Subota");break;
      case 7 : printf("Sedmi dan u sedmici je Nedjelja");break;
      default : printf("Greska !!!!");break;
    }
     getch();
}
```

STEK

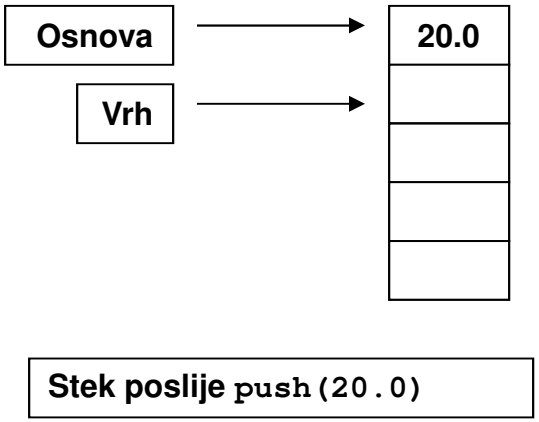
- Stek je mehanizam za prihvatabje podataka. To je linearni tip podataka koji radi na principu “poslednji u prvi iz” ili **LIFO – “Last In First Out”.** Praktično svi računari su orijentisani na rad sa stekom. Koriste se za čuvanje stanja računara za vrijeme opsluživanja prekidne rutine višeg nivoa, pri predaji argumenata funkcijama, pri aritimetičkim operacijama itd.
- Stek ima svojstvo tipa podatka. Može se primjenjivati pomoću niza i pratećih pokazivača ili indeksa koji pokazuju gdje podatak treba upisati i odakle ga treba pročirati.
- Za stek su značajni osnova steka i vrh steka. Osnova steka je pokazivač na početak niza podataka, tj. početna lokacija. Vrh steka je lokacija u koju treba upisati sledeći podatak.
- Upravljanje stekom vrši se pomoću dvije funkcije: **push()**koja upisuje podatke na stek i **pop()** koja čita podatke sa steka i pomoću indeksa koji pokazuju da li je stek prazan ili pun.

STEK

Na slici je šematski prikazano upravljanje sa stekom:

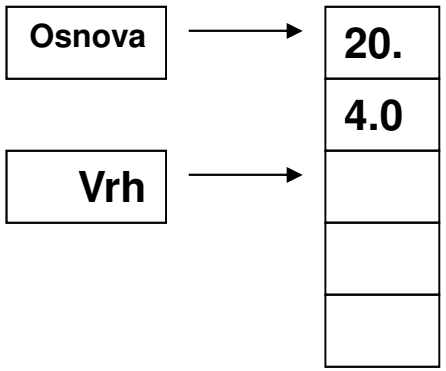


Prazan stek

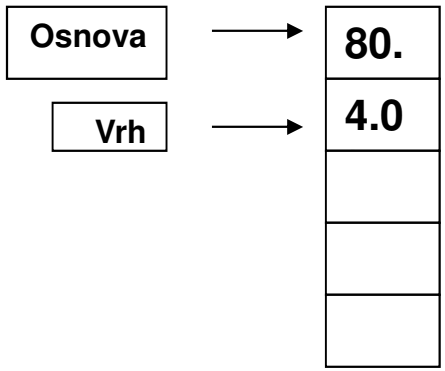


Stek poslije `push(20.0)`

STEK



Stek poslije **push(4.0)**



Stek poslije **push(pop()*pop())**

STEK

```
/*
 * stek.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define BAJTOVA 80
#define STEKVEL 10

/*  Prototipovi funkcija   */

int puch(char *);  /* Funkcija za upis na stek */
char *pop();       /* Funkcija za citanje steka */

main()
{
   char red[BAJTOVA+1];
   clrscr();
```

STEK

```
while(1)
   {
     printf("Komanda u, p ili i + return:  ");
     gets(red);
     switch(red[0])
            {
              case 'u':
              case 'U':
                printf("Upisi na stek:  ");
                gets(red);
                if(push(red)!=0)
                          printf("Stek je pun \n");
                break;
              case 'p':
              case 'P':
                printf("%s\n",pop());
                break;
              case 'i':
              case 'I':
                 return EXIT_SUCCESS;
            }
   }
```

STEK

```
static char stek[STEKVEL][BAJTOVA+1];
static int index=0;
/*
 *   Smjesta poruku na stek ako stek nije pun
 *   Ako je stek pun, prijavi gresku
 */

int push(char *poruka)
    {
      int pk=0;
      if(index<STEKVEL)
           strcpy(stek[index++],poruka);
      else
         ++pk;
      return pk;

    }
```

STEK

```
/*
 * Uzmi element sa vrha steka i vrati pokazivac na taj elem,ent
 * Ako je stek prazan, prikazi gresku
 */

char *pop()
     {
       if(index>0)
          return stek[--index];
       else
          return " Stek je prazan ";
     }
```