Pokazivač na pokazivač

- Učitavanje niza u funkciji

```
void Unos(int *a, int *n)      /* pogresno */
{
     int i;
     scanf("%d",n);
     a = (int*)malloc(*n * sizeof(int));
     for(i=0; i<*n; i++)
          scanf("%d",&a[i]);
}

...
int *a, n;
Unos(a, &n)
...
```

Pokazivač na pokazivač

- Učitavanje niza u funkciji

```
void Unos(int **a, int *n)  /* version 1 */
{
     int i;
     scanf("%d",n);
     *a = (int*)malloc(*n * sizeof(int));
     for(i=0; i<*n; i++)
          scanf("%d",&(*a)[i]);
}

...
int *a, n;
Unos(&a, &n)
...
```

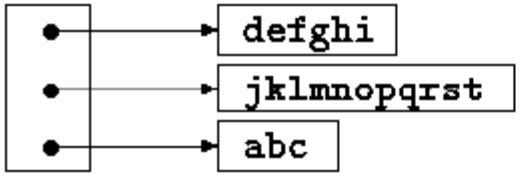
Pokazivač na pokazivač

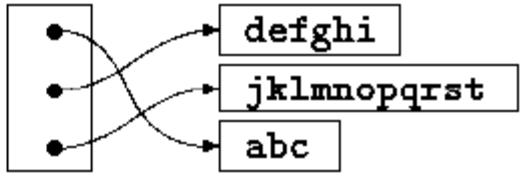
- Učitavanje niza u funkciji

```
int* Unos(int *n)  /* version 2 */
{
     int i, *a;
     scanf("%d",n);
     a = (int*)malloc(* n *sizeof(int));
     for(i=0; i<*n; i++)
          scanf("%d",&a[i]);
     return a;
}

...
int *a, n;
a = Unos(&n)
...
```

Nizovi pokazivača – Pokazivači na pokazivače

- Pokazivači su promenljive, pa se i oni mogu čuvati u nizovima.
- Primer: Sortiranje redova teksta po abecednom redu



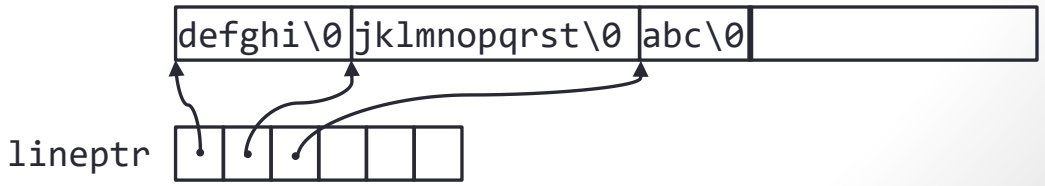


  - pročitati sve redove ulaznog teksta
  - sortirati ih
  - prikazati ih po redu

```c
#include <stdio.h>
#include <string.h>

#define MAXLINES 5000 /* maksimalan broj linija koje se mogu sortirati*/

char *lineptr[MAXLINES]; /* pokazivači na linije teksta */

int readlines(char *lineptr[], int nlines);
void writelines(char *lineptr[], int nlines);
void sort(char *lineptr[], int nlines);

/* sortiranje linija sa ulaza */
main()
{
	int nlines; /* broj učitanih linija */

	if ((nlines = readlines(lineptr, MAXLINES)) >= 0)
	{
		sort(lineptr, nlines);
		writelines(lineptr, nlines);
		return 0;
	}
	else
	{
		printf("greska: suvise veliki broj linija\n");
		return 1;
	}
}
```

```
#define MAXLEN 1000 /* maksimalna duzina linije */

int getline(char *, int);
char *alloc(int);

/* readlines: ucitava redove sa ulaza */
int readlines(char *lineptr[], int maxlines)
{
	int len, nlines;
	char *p, line[MAXLEN];

	nlines = 0;
	while ((len = getline(line, MAXLEN)) > 0)
		if (nlines >= maxlines || p = alloc(len) == NULL)
			return -1;
		else
		{
			line[len-1] = '\0'; /* brisanje znaka za novi red */
			strcpy(p, line);
			lineptr[nlines++] = p;
		}

	return nlines;
}
```

```
#define MAXLEN 1000 /* maksimalna duzina linije */

int getline(char *, int);
char *alloc(int);

/* readlines: ucitava redove sa ulaza */
int readlines(char *lineptr[], int maxlines)
{
	int len, nlines;
	char *p, line[MAXLEN];

	nlines = 0;
	while ((len = getline(line, MAXLEN)) > 0)
		if (nlines >= maxlines || p = alloc(len) == NULL)
			return -1;
		else
		{
			line[len-1] = '\0'; /* brisanje znaka za novi red */
			strcpy(p, line);
			lineptr[nlines++] = p;
		}

	return nlines;
}
```



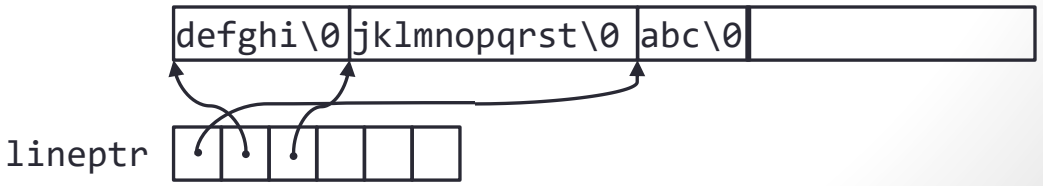
```
/* getline: get line into s, return length */
int getline(char s[], int lim)
{
	int c, i;

	i = 0;
	while (--lim > 0 && (c=getchar()) != EOF && c != '\n')
		s[i++] = c;

	if (c == '\n')
		s[i++] = c;

	s[i] = '\0';

	return i;
}
```

```
/* writelines: ispisivanje linija teksta */
void writelines(char *lineptr[], int nlines)
{
	int i;

	for (i = 0; i < nlines; i++)
		printf("%s\n", lineptr[i]);
}
```

ili ovako

```
/* writelines: ispisivanje linija teksta */
void writelines(char *lineptr[], int nlines)
{
	while (nlines-- > 0)
		printf("%s\n", *lineptr++);
}
```

```
/* sort: sortira niz v u rastucem redosledu */
void sort(char *v[], int nlines)
{
        int i, j;
        char *temp;

        for(i = 0; i < n-1; i++)
                for(j = i+1; j < n; j++)
                        if( strcmp(v[i],v[j]) > 0 )
                        {
                                temp = v[i];
                                v[i] = v[j];
                                v[j] = temp;
                        }
}
```



Učitavanje niza karaktera

- Funkcija

  `int scanf(const char *format, ...)`

  učitava formatirani ulaz sa standardnog ulaza do prvog znaka beline (blanko, tab, enter) i kao rezultat vraća broj učitanih podataka

```
char *ps, ns[100];
int d;
d=scanf("%s",ns);
ps=(char*)malloc(strlen(ns)+1);
strcpy(ps,ns);

printf("%d %s\n",d,ps);
printf("%d %d\n",strlen(ns),strlen(ps));
printf("%d %d\n",sizeof(ns),sizeof(ps));
```

Učitavanje niza karaktera

- Funkcija

  `int scanf(const char *format, ...)`

  učitava formatirani ulaz sa standardnog ulaza do prvog znaka beline (blanko, tab, enter) i kao rezultat vraća broj učitanih podataka

```
char *ps, ns[100];
int d;
d=scanf("%s",ns);                          ← strukture podataka
ps=(char*)malloc(strlen(ns)+1);
strcpy(ps,ns);

printf("%d %s\n",d,ps);                          → 1 strukture
printf("%d %d\n",strlen(ns),strlen(ps));         → 9 9
printf("%d %d\n",sizeof(ns),sizeof(ps));         → 100 4
```

Učitavanje niza karaktera

- Funkcija

  `char *gets(char *str)`

  učitava karaktere sa standardnog ulaza do oznake za kraj reda.
- U slučaja uspešnog učitavanja vrača `str`, a u slučaju greške ili oznake za kraj datoteke (ukoliko nema učitanih karaketra) vraća `NULL`

```
char *ps, ns[100];
gets(ns);                          ← strukture podataka
ps=(char*)malloc(strlen(ns)+1);
strcpy(ps,ns);

printf("%s\n",ps);                              → strukture podataka
printf("%d %d\n",strlen(ns),strlen(ps));        → 18 18
```

Učitavanje niza karaktera

- Funkcija

  `char *fgets(char *str, int n, FILE *stream)`

  učitava karaktere iz ulaza definisanog sa `stream`.
- Učitavanje se završava ili nakon učitane oznake za kraj reda ili nakon učtavanja n-1 karaktera

```
char *ps, ns[100];
fgets(ns,100,stdin);          ← strukture podataka
ps=(char*)malloc(strlen(ns)+1);
strcpy(ps,ns);

printf("%s",ps);                          → strukture podataka
printf("%d %d\n",strlen(ns),strlen(ps));        → 19 19
```

```
#define MAXLEN 1000 /* maksimalna duzina linije */

int getline(char *, int);
char *alloc(int);

/* readlines: ucitava redove sa ulaza */
int readlines(char *lineptr[], int maxlines)
{
        int len, nlines;
        char *p, line[MAXLEN];

        nlines = 0;
        while ((len = getline(line, MAXLEN)) > 0)
                if (nlines >= maxlines || p = alloc(len) == NULL)
                        return -1;
                else
                {
                        line[len-1] = '\0'; /* brisanje znaka za novi red */
                        strcpy(p, line);
                        lineptr[nlines++] = p;
                }

        return nlines;
}
```

```
while(fgets(line,MAXLEN,stdin)) {
        len=strlen(line);
        if (nlines>=maxlines  || p=(char*)malloc(len)==NULL)
```

Višedimenzionalni nizovi

- Primer: Konverzija dana u mesecu u dan u godini i obratno
  - funkcija `day_of_year` – pretvara mesec i dan u dan u godini
  - funkcija `month_day` – pretvara dan u u godini u mesec i dan

```
static char daytab[2][13] = {
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
        {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};

/* day_of_year: set day of year from month & day */
int day_of_year(int year, int month, int day)
{
        int i, leap;
        leap = year%4 == 0 && year%100 != 0 || year%400 == 0;
        for (i = 1; i < month; i++)
                day += daytab[leap][i];
        return day;
}
```

Višedimenzionalni nizovi

```
/* month_day: set month, day from day of year */
void month_day(int year, int yearday, int *pmonth, int *pday)
{
        int i, leap;
        leap = year%4 == 0 && year%100 != 0 || year%400 == 0;
        for (i = 1; yearday > daytab[leap][i]; i++)
                yearday -= daytab[leap][i];
        *pmonth = i;
        *pday = yearday;
}
```

Višedimenzionalni nizovi

- Pisanje indeksa

```
daytab[i][j]      /* ispravno */
daytab[i,j]       /* pogresno */
```

- Prenošenje višedimenzionalnih nizova funkciji

```
f(int daytab[2][13]) { ... }
```

Ili

```
f(int daytab[][13]) { ... }
```

a pošto je broj redova nebitan, može se napisati i

```
f(int (*daytab)[13]) { ... }
```

što znači da je parametar pokazivač na niz od 13 celih brojeva.

- Uglaste zagrade [] imaju veći prioritet od operatora *.
- Bez zagrada bi se dobila sledeća deklaracija

```
int *daytab[13]
```

što bi predstavljalo niz od 13 pokazivača na tip `int`.

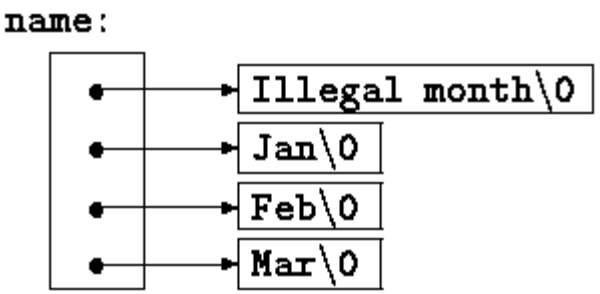
Inicijalizacija nizova pokazivača

- Primer: Funkcija koja vraća naziv meseca u godini

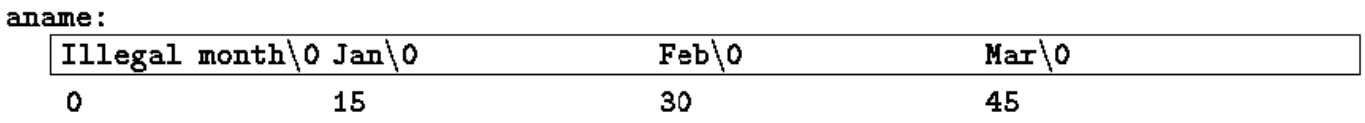
```
/* month_name: return name of n-th month */
char *month_name(int n)
{
    static char *name[] = {
            "Illegal month",
            "January", "February", "March",
            "April", "May", "June",
            "July", "August", "September",
            "October", "November", "December"
    };

    return (n < 1 || n > 12) ? name[0] : name[n];
}
```

Pokazivači ili višedimenzionalni nizovi?

```
int a[10][20];
int *b[10];
int (*c)[20];

char *name[] = { "Illegal month", "Jan", "Feb", "Mar" };
```




```
char aname[][15] = { "Illegal month", "Jan", "Feb", "Mar" };
```



Pokazivači ili višedimenzionalni nizovi?

- Dinamičko deklarisanje matrice celih brojeva

```
int **a;
int m, n, i, j;
scanf("%d%d",&m,&n);
a = (int**)malloc(m * sizeof(int*));
for(i=0; i<m; i++)
{
    a[i] = (int*)malloc(n * sizeof(int));
    for(j=0; j<n; j++)
        scanf("%d",&a[i][j]);
}
```

Argumenti komandnog reda

- Agumente (parametre) komandnog reda moguće je proslediti programu na početku njegovog izvršavanja

- `main` funkcija se poziva sa dva argumenta
  - `argc` (*argument count*) – broj argumenata u komandnom redu
  - `argv` (*argument vector*) – pokazivač na niz stringova koji sadrži te argumente

- Po konvenciji `argv[0]` je ime programa, tako da je argc bar 1

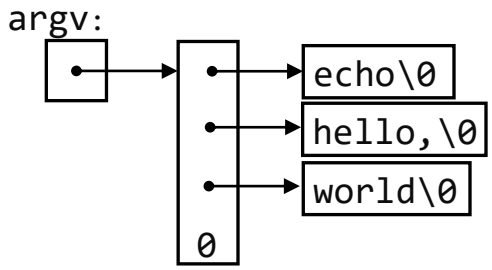
ECHO

- Program koji svoje argumente razdvojene razmakom ispisuje na izlazu
- Pozivom

  `echo hello, world`

  na izlazu se ispisuje

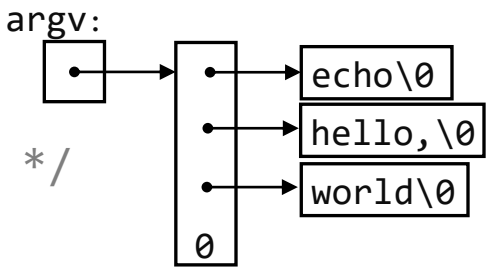
  `hello, world`




```
/* echo command-line arguments; 1st version */
main(int argc, char *argv[])
{
      int i;
      for(i=1; i < argc; i++)
           printf("%s%s",argv[i],(i<argc-1)? " ":"");
      printf("\n");
      return 0;
}
```

ECHO

- argv je pokazivač, pa možemo manipulisati pokazivačem, a ne indeksom niza




```
/* echo command-line arguments; 2st version */
main(int argc, char *argv[])
{
     while(--argc > 0)
          printf("%s%s",*++argv, (argc > 1)? " ":"");
     printf("\n");
     return 0;
}
```

- Komanda `printf` može da se zapiše i na sledeći način

```
printf((argc > 1)? "%s ": "%s",*++argv);
```

Prepoznavanje obrazaca u tekstu

- Obrazac koji se traži je prvi argument pri pokretanju

```
#include <stdio.h>
#include <string.h>
#define MAXLINE 1000

/* find : print lines that match pattern from 1st arg */
main(int argc, char *argv[])
{
     char line[MAXLINE];
     int found=0;
     if(argc != 2)
         printf("Usage : find patern\n");
     else
         while(fgets(line, MAXLINE,stdin) != 0)
             if(strstr(line, argv[1]) != NULL)
             {
                  printf("%s", line);
                  found++;
             }
      return found;
 }
```