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;
 }
```

Prepoznavanje obrazaca u tekstu

- Ako dozvolimo dva opciona argumenta:
  - Prikazati sve redove osim onih u kojima se nalazi obrazac -x
  - Ispred svakog prikazanog reda navesti njegov redni broj -n
- Opcioni argumenti mogu biti dozvoljeni u proizvoljnom redosledu i mogu se kombinovati

```
find –nx obrazac
```

```
#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];
    long lineno=0;
    int c, except=0, number=0, found=0;
    ...
```

Propoznavanje obrazaca u tekstu

```
...
   while(--argc > 0 && (*++argv)[0] == '-')
       while(c = *++argv[0])
           switch(c)
           {
               case 'x':
                     except = 1;
                     break;
               case 'n':
                     number = 1;
                     break;
               default:
                     printf(" find : illegal option %c\n", c);
                     argc = 0;
                     found = -1;
                     break;
           }
   ...
```

Propoznavanje obrazaca u tekstu

```
...
   while(--argc > 0 && (*++argv)[0] == '-')
       while(c = *++argv[0])
           switch(c)
           {
               case 'x':
                     except = 1;
                     break;
               case 'n':
                     number = 1;
                     break;
               default:
                     printf(" find : illegal option %c\n", c);
                     argc = 0;
                     found = -1;
                     break;
           }
   ...
```

**++argv

Propoznavanje obrazaca u tekstu

```
...
   while(--argc > 0 && (*++argv)[0] == '-')
       while(c = *++argv[0])
           switch(c)
           {
               case 'x':
                     except = 1;
                     break;
               case 'n':
                     number = 1;
                     break;
               default:
                     printf(" find : illegal option %c\n", c);
                     argc = 0;
                     found = -1;
                     break;
           }
   ...
```

Uvećava pokazivač `argv[0]`, tj. prelazi na sledeći karakter u stringu

Propoznavanje obrazaca u tekstu

```
    ...
    if(argc!=1)
        printf("Usage : find -x -n pattern\n");
    else
        while(fgets(line, MAXLINE,stdin) > 0)
        {
            lineno++;
            if((strstr(line, *argv)!=NULL)!=except)
            {
                if(number)
                    printf("%ld:", lineno);
                printf("%s", line);
                found++;
            }
        }
    return found;
}
```

Propoznavanje obrazaca u tekstu

```
if((strstr(line, *argv)!=NULL)!=except)
```

except	strstr	!=NULL	!=except	Štampa
0	pronašao	1	1	DA
0	nije pronašao	0	0	NE
1	pronašao	1	0	NE
1	nije pronašao	0	1	DA

Pokazivači na funkcije

- Moguće je definisati pokazivače na funkcije, iako funkcija nije promenljiva
- Ovim pokazivačima se mogu dodeljivati vrednosti, mogu se čuvati u nizovima, prenositi funkcijama, vraćati kao rezultat funkcija ...

- U programu za leksikografsko sortiranje linija teksta, dozvolićemo opcioni parametar **-n** koji označava da treba izvršiti numeričko sortiranje, tj. sortiranje učitanih brojeva
- Algoritam za sortiranje ne zavisi od tipa vrednosti koje se sortiraju
- Funkcija `strcmp` će i dalje biti korišćenja za sortiranje teksta, definisaćemo funkciju `numcmp` koja će dva reda porediti na osnovu numeričke vrednosti i vraćati isti rezltat kao `strcmp`.

Pokazivači na funkcije

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

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

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

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

void sort(void *lineptr[], int nlines, int (*comp)(void *,void *));
int numcmp(char *,char *);
void swap(void **, int, int);
```

Pokazivači na funkcije

```
main(int argc, char *argv[])
{
     int nlines;        /* broj ucitanih linija */
     int numeric = 0;   /* 1 ako je numericko sortiranje */

     if(argc > 1 && strcmp(argv[1],"-n")==0)
       numeric = 1;

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

Pokazivači na funkcije

```
void sort(void *v[], int n, int (*comp)(void*, void*))
{
     int i, j;

     for(i = 0; i < n-1; i++)
       for(j = i+1; j < n; j++)
            if( (*comp)(v[i],v[j]) > 0 )
                   swap(v,i,j);
}
```

Pokazivači na funkcije

```
int numcmp(char *s1, char *s2)
{
    double v1, v2;

    v1=atof(s1);
    v2=atof(s2);
    if(v1<v2) return -1;
    else if(v1>v2) return 1;
    else return 0;
}

void swap(void *v[], int i, int j)
{
     void *temp;

     temp = v[i];
     v[i] = v[j];
     v[j] = temp;
}
```

Komplikovane deklaracija

- `int *f();`
  - f : funkcija koja vraća pokazivač na int
- `int (*pf)();`
  - pf : pokazivač na funkciju koja vraća int
- `char **argv`
  - argv : pokazivač na pokazivač na char
- `int (*daytab)[13]`
  - daytab : pokazivač na niz[13] od int
- `int *daytab[13]`
  - daytab : niz[13] pokazivača na int

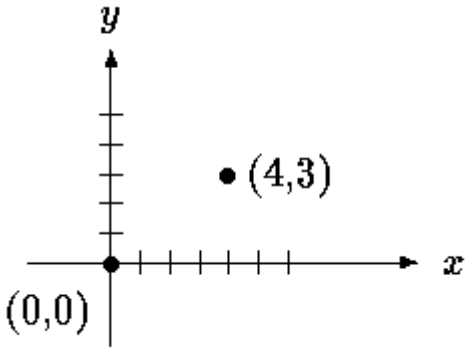
Komplikovane deklaracija

- `void *comp()`
  - comp : funkcija koja vraća pokazivač na void
- `void (*comp)()`
  - comp : pokazivač na funkciju koja vraća void
- `char (*(*x())[])()`
  - x : funkcija koja vraća pokazivač na niz[] pokazivača na funkciju koja vraća char
- `char (*(*x[3]))()[5]`
  - x : niz[3] pokazivača na funkciju koja vraća pokazivač na niz[5] od char

STRUKTURE I UNIJE

Osnove struktura

```
struct point
{
        int x;
        int y;
};
```



- Strukture se deklarišu pomoću ključne reči `struct`
- Naziv strukture može se pisati iza reči `struct`
- Promenljive članice strukture se navoda između vitičastih zagrada
- Promenljive članice mogu imati isti naziv kao i obične promenljive ili kao članice neke druge strukture

Osnove struktura

- Deklaracija `struct` definiše tip
- Iza deklaracije strukture može se nalaziti lista promenljivih tog tipa

```
struct { ... } x, y, z;
```

potpuno analogno kao

```
int x, y, z;
```

- Deklaracije strukture iza koje se ne nalazi lista promenljivih ne izaziva rezervisanje memorije. Ona predstavlja šablon strukture.
- Naziv neke strukture se može iskoristiti za deklarisanje primeraka te strukture
- Ako je prethodno deklarisana struktura `point`, onda

```
struct point pt;
```

deklariše promenljivu pt koja je struktura tipa struct point

- Struktura se može inicijalizovati navođenjem inicijalnih vrednosti

```
struct point maxpt = { 320, 200 };
```

Osnove struktura

- Član određene strukture se može upotrebiti korišćenjem oblika

  ime_strukture.član

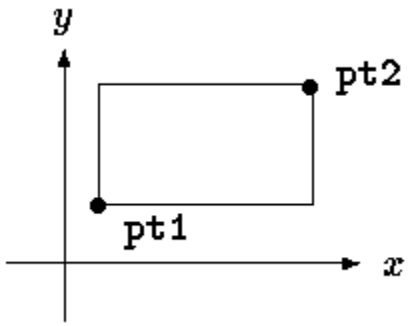
- Na primer, koordinate tačke `pt` možemo prikazati pomoću

```
printf("%d,%d", pt.x, pt.y);
```

- Slično, rastojanje od koordinatnog početka do tačke pt možemo izračunati kao

```
dist = sqrt((double)pt.x * pt.x + (double)pt.y * pt.y);
```

Osnove struktura



- Članice struktura mogu biti takođe strukture

```
struct rect
{
        struct point pt1;
        struct point pt2;
};
```

- Sada možemo deklarisati promenljivu `screen`

```
struct rect screen;
```

pa bi onda koordinata x prve tačke pravougaonika bila

```
screen.pt1.x;
```

Strukture i funkcije

- Jedine dozvoljene operacije sa strukturama su kopiranje, dobijanje njihove adrese operatorom & i pristup njihovim članovima
- Operacije kopiranja obuhvataju prenošenje struktura kao argumenata funkcijama i vraćanje vrednosti iz funkcija
- Strukture se ne mogu porediti
- Strukture se mogu inicijalizovati listom konstantnih vrednosti njhovih članova
- Kako slati podatke iz strukture funkciji
  - poslati svaki podatak pojedinačno
  - poslati celu strukturu
  - poslati pokazivač na tu strukturu

Strukture i funkcije

- Funkcija makepoint uzima dve koordinate (celobrojne vrednosti) i vraća strukturu

```
/* makepoint: make a point from x and y components */
struct point makepoint(int x, int y)
{
    struct point temp;

    temp.x = x;
    temp.y = y;

    return temp;
}
```

- Obratite pažnju da ne postoji konflikt između argumenata funkcije i članica strukture

Strukture i funkcije

- Primer korišćenja

```
struct rect screen;
struct point middle;
struct point makepoint(int, int);

screen.pt1 = makepoint(0,0);
screen.pt2 = makepoint(XMAX, YMAX);

middle = makepoint((screen.pt1.x + screen.pt2.x)/2,
                   (screen.pt1.y + screen.pt2.y)/2);
```

Strukture i funkcije

- Funkcija za sabiranje dva vektora

```
/* addpoints: add two points */
struct point addpoint(struct point p1, struct point p2)
{
    p1.x += p2.x;
    p1.y += p2.y;
    return p1;
}
```

- Strukturni elementi se prenose po vrednosti kao i svaki drugi

- Funkcija koja proverava da li je tačka unutar pravougaonika

```
/* ptinrect: return 1 if p in r, 0 if not */
int ptinrect(struct point p, struct rect r)
{
    return  p.x >= r.pt1.x && p.x < r.pt2.x &&
            p.y >= r.pt1.y && p.y < r.pt2.y;
}
```

Strukture i funkcije

- Najčešće je pogodnije funkciji poslati pokazivač na strukturu
- Deklaracija

```
struct point *pp;
```

označava da je pp pokazivač na strukturu tipa struct point

- Ako pp pokazuje na strukturu `point`, onda se njenim članicama može pristupiti sa `(*pp).x` i `(*pp).y`

```
struct point pt, *pp;
pp = &pt;
printf("Point is (%d,%d)\n", (*pp).x, (*pp).y);
```

- Ako je p pokazivač na neku strukturu, pristupanje članicama strukture može se kraće izvršiti korišćenjem

p->član_strukture

tako da bi poslednji red prethodnog primera glasio

```
printf("Point is (%d,%d)\n", pp->x, pp->y);
```

Strukture i funkcije

- Ako imamo

```
struct rect r, *rp = &r;
```

  onda su sledeći izrazi ekvivalentni

```
r.pt1.x        rp->pt1.x      (r.pt1).x      (rp->pt1).x
```

- Strukturni operatori "." i "->" su na vrhu hijerarhije prioriteta

```
struct {
   int len;
   char *str;} *p;
```

- `++p->len` - uvećava `len`, a ne `p`
- `(++p)->len` - uvećava `p`, pre pristupanja članu `len`
- `(p++)->len`, tj. `p++->len` - uvećava `p`, nakon pristupanja članu `len`
- `*p->str` - predstavlja karakter na koji `str` pokazuje
- `*p->str++` - uvećava pokazivač `str` (isto kao `*s++`)
- `(*p->str)++` - uvećava ono na šta `str` pokazuje
- `*p++->str` - uvećava `p` nakon pristupa onome na šta `str` pokazuje