A karakterláncokat kezelő függvényeket tartalmazó könyvtár használata

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

A leírásokban használt változók

```
int i, j, n;
char s[200], z[200], *p;
char c;
```

Egy karakterlánc végét mindig a nullkarakter jelzi (0 vagy '\0' a jele).

Az **s** és **z** változók esetén a gép lefoglal egyenként 200 karakter tárolására alkalmas összefüggő memóriaterületet, a terület kezdőcímét beírja az **s** illetve a **z** változókba.
Ezeket a kezdőcímeket többet nem lehet módosítani.

A **p** változó esetén csak egy memóriacím tárolására alkalmas területet foglal le, bármilyen karakterláncot tartalmazó memóriacímet be lehet ide írni. A **p** tartalma tehát módosítható.

`n=strlen(s);`	az **n** változóba bemásolja az **s** karakterlánc hosszát (megkeresi a '\0' karakter helyét) **`strlen("alma") --> 4`**
`p=strupr(s)`	nagybetűsíti az **s** karakterláncot (az **s**-ben végzi el a módosításokat) **`strupr("alma") --> ALMA`**
`p=strlwr(s);`	kisbetűsíti az **s** karakterláncot (az **s**-ben végzi el a módosításokat) **`strlwr("B.U.E.K") --> b.u.e.k.`**
`p=strrev(s);`	megfordítja az **s** karakterlánc betűit (az **s**-be az eredeti karakterlánc tükörképe kerül) **`strrev("rét") --> tér`**
`p=strcpy(z, s);`	a **z** karakterláncba bemásolja az **s** karakterláncot (a **z** eredeti tartalma eltűnik) **`z="fa"`** **`strcpy(z,"alma") --> alma`**
`p=strncpy(z, s, n);`	a **z** karakterláncba a 0. pozíciótól kezdődően bemásolja az **s** karakterláncból az első **n** darab betűt ha az első **n** betűben nem volt ott a sorvégjel és a **z** eredetileg hosszabb volt, akkor átírja az első **n** betűt **`z="bioinformatika"`** **`strncpy(z,"matematika", 5) --> matemformatika`**
`p=strcat(z, s);`	a **z** karakterlánc végéhez illeszti az **s** karakterláncot **`z="alma"`** **`strcat(z,"fa") --> almafa`**
`p=strncat(z, s, n);`	a **z** karakterlánc végéhez illeszti az **s** karakterlánc első **n** darab betűjét **`z="alma"`** **`strncat(z,"farkas",2) --> almafa`**
`p=strcmp(s, z);`	karakterenként összehasonlítja az **s** és **z** karakterláncot, különbséget tesz kis- és nagybetűk között - ha az eredmény < 0 akkor ábécé sorrendben az **s** karakterlánc a **z**

	karakterlánc előtt kell legyen - ha az eredmény nulla akkor az **s** és **z** egyforma - ha az eredmény > 0 akkor ábécé sorendben az **s** karakterlánc a **z** karakterlánc után kell legyen
p=stricmp(s, z);	az **s** és **z** karakterláncokat hasonlítja össze, de nem tesz különbséget a kis- és nagybetűk között
p=strncmp(s, z, n);	az **s** és **z** karakterláncot hasonlítja össze, de csak az első **n** pozíciót veszi figyelembe, különbség van kis- és nagybetűk között
p=strnicmp(s,z,n);	az **s** és **z** karakterláncot hasonlítja össze, de csak az első **n** pozíciót veszi figyelembe, nincs különbség kis- és nagybetűk között
p=strchr(s, c);	balról kezdve megkeresi az **s** karakterláncban a **c** karakter első előfordulását és visszaadja annak a karakternek a memóriacímét, ahol megtalálta. Ha nem szerepelt benne a **c** karakter, akkor a NULL értéket téríti vissza. **p=strchr("almafa",'m');** **puts(p); --> mafa**
p=strrchr(s, c);	jobbról kezdve keresi az **s** karakterláncban a **c** karaktert, jobbról az első előfordulás pozícióját adja vissza **p=strrchr("matematika",'m');** **puts(p); --> matika**
p=strstr(s, z);	balról kezdve megkeresi az **s**-ben az első olyan pozíciót, ahol a **z** karakterlánc kezdődik **p=strstr("barbara","ar");** **puts(p); --> arbara**
n=strspn(s, z);	balról kezdve megkeresi az **s**-ben az első olyan pozíciónak a sorszámát, ahol olyan karakter van, ami nem szerepel a **z**-ben **n=strspn("tam-tam","matek") --> n==3**
n=strcspn(s, z);	balról kezdve megkeresi az **s**-ben az első olyan pozíciónak a sorszámát, ahol olyan karakter van, ami szerepel a **z**-ben **n=strcspn("informatika","forma") --> n==2**
p=strpbrk(s, z);	balról kezdve megkeresi az **s**-ben az első olyan karaktert, ami szerepel a **z**-ben és visszaadja a karakter memóriacímét **p=strpbrk("informatika","forma");** **puts(p); --> formatika**
p=strset(s, c);	az **s** karakterlánc betűit lecseréli a **c** karakterre **p=strset("almafa", 'x');** **puts(p); --> xxxxxx**
p=strnset(s, c, n);	az **s** karakterlánc első **n** betűjét lecseréli a **c** karakterre **p=strnset("informatika", 'x', 4);** **puts(p); --> xxxxrmatika**
p=strdup(s);	helyet foglal a memóriában a **p** számára és az **s** karakterlánc tartalmát átmásolja a **p**-be ***p** nem jelent helyfoglalást, ez csak memóriacím lesz!!!
p=strtok(s, z);	az **s** karakterláncot szétdarabolja a **z**-ben megadott jelek szerint

	Szavakra tördelésre lehet nagyon jól használni. A következő programrészletben az **s**-ben megadott mondatot szavakra tördeljük. **char s[200]="egy, ketto..., szaz?!? ezer...";** **char elv[]=", ?!\n";** **char *p;** **p=strtok(s,elv);** **while(p)** **{** **puts(p);** **p=strtok(NULL,elv);** **}**