Szöveg (sztring)

A sztring egy karakterekből álló tömb. Mivel a szövegek általában nem fix hosszúságúak, nem töltik ki a rendelkezésre álló tömböt, ezért jelölni kell, hogy hol a szöveg vége. Ezt bináris nullával vagy nulla kódú karakterrel (0 vagy ’\0’) jelöljük.

Deklarálása: `char szoveg[10];`

Jelentése: lefoglalunk a memóriában 10 bájt helyet a szöveg tárolására. (A záró karakternek is el kell férnie!)

’a’	’l’	’m’	’a’	’\0’					

Sztring inicializálása (kezdőértékadása):

```
char szoveg1[10] = {’a’, ’l’, ’m’, ’a’, ’\0’};
char szoveg2[10] = ”alma”;
char szoveg3[] = ”alma”;
```

Példa:

Olvassunk be két sor szöveget, majd írjuk vissza azokat. Beolvasáshoz használjuk a *getchar()* függvényt.

Megoldás:

```
#include <stdio.h>
#include <iostream>
int main(){
    setlocale (LC_ALL, "");
    char szoveg[80],sz[80], i=0;
    printf("Kérek egy sor szöveget: ");
    while((szoveg[i]=getchar())!='\n')i++;
    szoveg[i++]='\n';
    szoveg[i]='\0';
    i=0;
    printf("Kérek egy másik szöveget: ");
    while((sz[i++]=getchar())!='\n');
    sz[i]='\0';
    printf("A beírt szövegek:\n %s %s",szoveg,sz);
}
```

Példa:

Oldjuk meg az előző feladatot úgy, hogy a beolvasáshoz a *scanf()* függvényt használjuk.

Megbeszélés:

1. *scanf()* függvény, "%s" és "%[^\n]" formátum,
2. billentyűzet puffer törlés*: fflush()* függvény.

Megoldás:

```
#include <stdio.h>
#include <stdio.h>
#include <iostream>
int main(){
    setlocale (LC_ALL, "");
    char szoveg[80],sz[80], i=0;
    printf("Kérek egy sor szöveget: "); scanf("%s", szoveg);
    fflush(stdin);
    printf("Kérek egy másik szöveget: "); scanf("%[^\n]", sz);
    printf("A beírt szövegek:\n %s\n %s\n",szoveg,sz);
}
```

Példa:

Olvassunk be egy sor szöveget és határozzuk meg a hosszát (hány karaktert olvastunk be).

Megoldás:

```
#include <stdio.h>
int main(){
        char szoveg[80],i=0, hossz=0;
        printf("kerek egy sor szoveget: "); scanf("%[^\n]", szoveg);
        while(szoveg[hossz])hossz++;
        printf("\nA szoveg hossza: %d\n",hossz);
}
```

Példa:

Oldjuk meg az előző feladatot úgy, hogy a beolvasáshoz és a számoláshoz mutatót használunk.

Megoldás:

```
#include <stdio.h>
int main(){
        char szoveg[80],*p=szoveg;
        printf("kerek egy sor szoveget: ");
        while((*p=getchar())!='\n')p++;
        *p='\0';
        p=szoveg;
        while(*p)p++;
        printf("\n\"%s\" szoveg hossza = %d\n",szoveg, p-szoveg);
}
```

Példa:

A bekért szöveget másoljuk át egy másik tömbbe. Oldjuk meg a feladatot indexek, majd mutatók segítségével is.

Megoldás 1:

```
#include <stdio.h>
int main(){
        char szoveg[80], masolat[80],i=0;
        printf("kerek egy sor szöveget: "); scanf("%[^\n]", szoveg);
        while(masolat[i]=szoveg[i++]);
        printf("\nEz a masolt szoveg: %s\n",masolat);
}
```

Megoldás 2:

```
#include <stdio.h>
int main(){
        char szoveg[80], masolat[80], *p=szoveg, *m=masolat;
        printf("kerek egy sor szoveget: ");
        while((*p++=getchar())!='\n');
        *p='\0';
        p=szoveg;
        while(*m++=*p++);
        printf("\nEz a masolt szoveg: %s",masolat);
}
```

Többdimenziós tömbök

Az előző feladatokban egydimenziós tömböket használtunk. Azonban lehetőségünk van két- vagy többdimenziós tömb használatára is. Példaként vizsgáljuk meg, hogy néz ki egy 4x5 méretű kétdimenziós tömb, és hogyan használhatjuk azt.

int to[4][5]

to[0][0]	to[0][1]	to[0][2]	to[0][3]	to[0][4]
to[1][0]	to[1][1]	to[1][2]	to[1][3]	to[1][4]
to[2][0]	to[2][1]	to[2][2]	to[2][3]	to[2][4]
to[3][0]	to[3][1]	to[3][2]	to[3][3]	to[3][4]

Írjuk fel mutatókkal:

to →	to[0] →				
	to[1] →				
	to[2] →				
	to[3] →				

Példa:

Olvassunk be három sornyi szöveget, majd írjuk vissza a képernyőre. Használjuk a *scanf()* és *printf()* függvényeket.

Megoldás:

```
#include <stdio.h>
int main(){
        char a, szoveg[3][80];
        for (char j=0;j<3;j++){
                printf("kerem az %d. sor szöveget: ",j);
                fflush(stdin);
                scanf("%[^\n]",szoveg[j]);
        }
        for (char j=0;j<3;j++)printf("\nEz a szoveg: %s\n",szoveg[j]);
}
```

Példa:

Oldjuk meg az előző feladatot de most a *getchar()* és a *putchar()* függvényekkel.

Megoldás:

```
#include <stdlib.h>
int main(){
	char szoveg[3][80],i,j;
	for (j=0; j<3; j++){
		printf("kerem az %d. sor szoveget: ",j+1);
		i = 0;
		while((szoveg[j][i++]=getchar())!='\n');
		szoveg[j][i] = '\0';
	}
	for (j=0; j<3; j++){
		printf("A %d. sor szovege: ",j+1);
		i = 0;
		while(szoveg[j][i]) putchar(szoveg[j][i++]);
	}
}
```

Példa:

Oldjuk meg az előző feladatot mutatóval.

Megoldás:

```
#include <stdlib.h>
int main(){
	char szoveg[3][80], *s;
	for (char j=0; j<3; j++){
		printf("kerem az %d. sor szoveget: ",j+1);
		s = szoveg[j];
		while((*s++=getchar())!='\n');
		*s = '\0';
	}
	for (char j=0; j<3; j++){
		printf("A %d. sor szovege: ",j+1);
		s = szoveg[j];
		while(*s) putchar(*s++);
	}
}
```

Példa:

Töltsünk fel egy kétdimenziós szövegtömböt a hónapok nevével. Kérjük be a hónap sorszámát, majd írjuk ki a sorszámnak megfelelő hónapnevet.

Megbeszélés:

1. Tömb kezdőértékadás:
   int tomb[]={1, 2, 3, 4, 5};
2. Kétdimenziós tömb kezdőértékadás:
   int tomb[][5]={{1, 2, 3, 4, 5}, {9, 8, 7, 6, 5}, {3, 5, 7, 9, 1}};
3. Szövegkonstans:
   char *s=”alma”;

Megoldás:

```
#include <stdio.h>
int main(){
        char *honev[]={
                "",
                "januar",
                "februar",
                "marcius",
                "aprilis",
                "majus",
                "junius",
                "julius",
                "augusztus",
                "szeptember",
                "oktober",
                "november",
                "december"
        };
        int ho;
        printf("Kerem a honap szamat: "); scanf("%d",&ho);
        if(ho >= 1 && ho <= 12)
                printf("%s\n",honev[ho]);
        else
                printf("Hibas honapnev");
}
```

Feladatok:

1. Egy tömbbe olvassunk be egy sor szöveget majd számoljuk meg, hogy hány kisbetű karaktert adtunk meg.

2. Egy tömbbe olvasson be egy sor szöveget, majd döntse el, hogy numerikus-e. (Numerikus, ha csak szám karaktereket tartalmaz!)

3. Olvasson be két egy-egy soros szöveget, majd hasonlítsa össze azokat. Írja ki, hogy megegyeznek, vagy különböznek.

4. Egy tömbbe olvassunk be egy sor szöveget, majd vizsgáljuk meg, hogy visszafelé olvasva is ugyanaz-e a jelentése. Az eredményt szövegesen írja ki.

5. Egy tömbbe olvassunk be több sor szöveget (EOF jelig), majd számoljuk meg, hogy hány karaktert tartalmaz.

6. Egy tömbbe olvassunk be több sor szöveget (EOF jelig), majd számoljuk meg, hogy hány sort tartalmaz

7. Töltsön fel egy kétdimenziós szövegtömböt egy-egy soros szöveggel. (Előre ne adja meg a feltöltendő sorok számát. Végjelig folytassa a feltöltést.)

8. Töltsön fel egy kétdimenziós szövegtömböt nevekkel, és egy egész tömböt a nevekhez tartozó telefonszámmal. (A tömbökben azonos indexű elemek tartoznak össze.) Majd kérjen be egy telefonszámot, és írja ki a hozzá tartozó nevet.