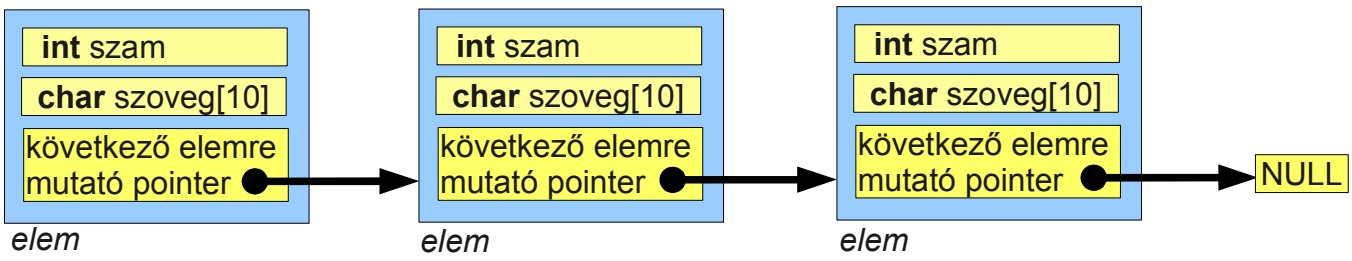
Láncolt lista



A láncolt lista egymás után egy irányba felfűzött elemek halmaza.

Hogy készül a láncolt lista?

Veszünk egy struct-ot, ami a lista elemének prototípusa lesz.

```
typedef struct lista_struct
{
     int szam;
     char szoveg[10];
     struct lista_struct *kovetkezo;
} lista_tipus;
```

az itt adott nevet csak a struct deklaráción belül használjuk

a következő ugyanilyen típusú listaelemre mutató pointer

ezzel a névvel hivatkozunk majd valójában a típusra

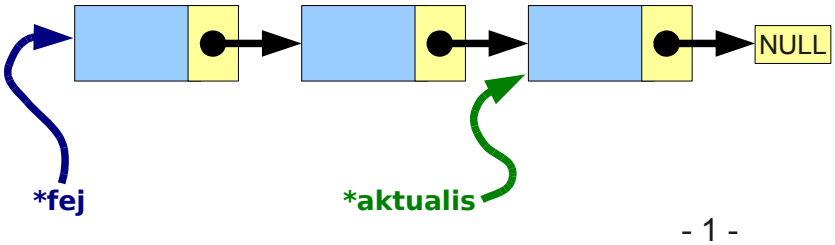
Mondjuk van egy *adat.txt* szöveg fájlunk, ami a következő módon néz ki:

```
12 macska
23 kutya
53 zsiraf
66 elefant
10 nyul
6 pocok
```

Ebből ki akarjuk olvasni az adatokat és bele akarjuk rakni a fenti listába. Hogyan tegyük ezt meg?

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

int main()
{
     lista_tipus *fej;
     lista_tipus *aktualis;
```

A láncolt lista használatához szükséges két mutató:
- a lista **feje**, ez az első elemre mutat.
- az **aktuális** elemre mutató pointer



A fájból való olvasáshoz a következő sorokat kell hozzáadnunk a programhoz:

```
int beolvasott_szam;
char beolvasott_szoveg[10];
FILE *f;

f=fopen("adat.txt", "r");

while(fscanf(f,"%d %s",&beolvasott_szam,beolvasott_szoveg) !=  EOF)
{

     printf("szam: %d, szoveg: %s\n",
          beolvasott_szam, beolvasott_szoveg);

}
```

Ez idáig annyit csinál, hogy a végéig olvassa az *adat.txt*-t. Az *adat.txt* sorai a *beolvasott_szam* és *beolvasott_szoveg* változókba kerülnek. Ezeket kellene betenni a láncolt listába.

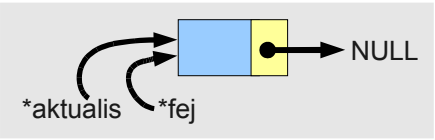
Új kód:

a fej NULL értéke jelzi, hogy még nincs egy eleme sem a láncolt listának

az *fscanf* kiolvas egy sort a fájlból a megadott formátum szerint. Ha elfogyott a fájl, ún. *EOF* értéket ad vissza.

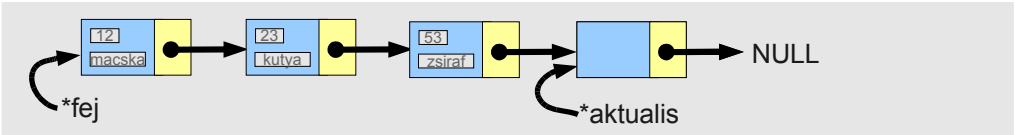
```
fej=NULL;
while(fscanf(f,"%d %s",&beolvasott_szam,beolvasott_szoveg) !=  EOF)
{
    if(fej==NULL)
    {
        fej=(lista_tipus*)malloc(sizeof(lista_tipus));
        aktualis=fej;
        aktualis->kovetkezo=NULL;
```

Ha üres a listánk:
1.) lefoglalunk helyet az első elemnek
2.) az első elemet tesszük aktuálissá.
3.) A láncolt lista végére mindig a NULL kerül.



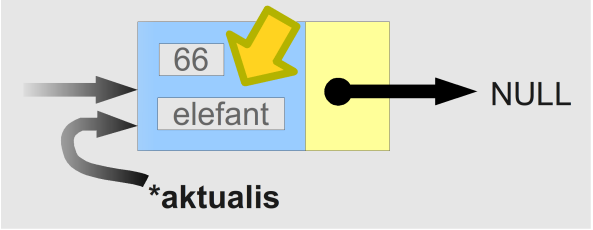

```
    }
    else
    {
        aktualis->kovetkezo=(lista_tipus*)malloc(sizeof(lista_tipus));
        aktualis->kovetkezo->kovetkezo=NULL;
        aktualis=aktualis->kovetkezo;
```

Ha már nem üres a listánk:
Mindig az aktuális elem végére fűzzük az újabb elemet.




```
    }
    aktualis->szam=beolvasott_szam;
    strcpy(aktualis->szoveg,beolvasott_szoveg);
```

A beolvasott adatok átrakása az aktuális elembe.




```
}
```

Nagyszerű! Idáig van egy teljesen kész, működő kódunk, ami egy *fájl* alapján felépít a memóriában egy *láncolt listát.*

A teljes kód így néz ki:

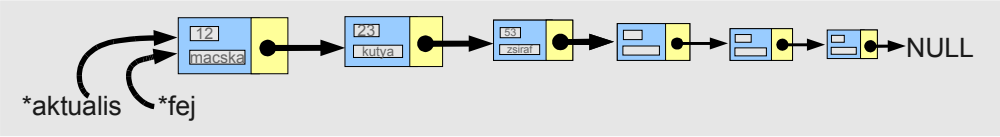
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct lista_struct
{
     int szam;
     char szoveg[10];
     struct lista_struct *kovetkezo;
} lista_tipus;

int main()
{
     lista_tipus *fej;
     lista_tipus *aktualis;
     int beolvasott_szam;
     char beolvasott_szoveg[10];

     FILE *f;
     f=fopen("adat.txt", "r"); //Fajl megnyitasa olvasasra

     fej=NULL;
     while(fscanf(f,"%d %s",&beolvasott_szam,beolvasott_szoveg) != EOF)
     {
          if(fej==NULL) //Uj elem hozzaadasa, ha meg ures volt a listank
          {
               fej=(lista_tipus*)malloc(sizeof(lista_tipus)); //Uj elem
               aktualis=fej; //A fej legyen az aktualis
               aktualis->kovetkezo=NULL; //Az uj elem legyen a lista vege
          }
          else //Uj elem hozzaadasa ha mar volt elem a listankban
          {
               aktualis->kovetkezo=
                    (lista_tipus*)malloc(sizeof(lista_tipus)); //Uj elem
               aktualis->kovetkezo->kovetkezo=NULL; //Lista vege
               aktualis=aktualis->kovetkezo; //Az uj elem legyen aktualis
          }
          //Az uj elem ertekeinek beallitasa
          //a beolvasott ertekek alapjan
          aktualis->szam=beolvasott_szam;
          strcpy(aktualis->szoveg,beolvasott_szoveg);
     }
}
```

Ha később fel akarjuk használni ezt a láncolt listát, *végig kell tudnunk menni a lista elemein*.

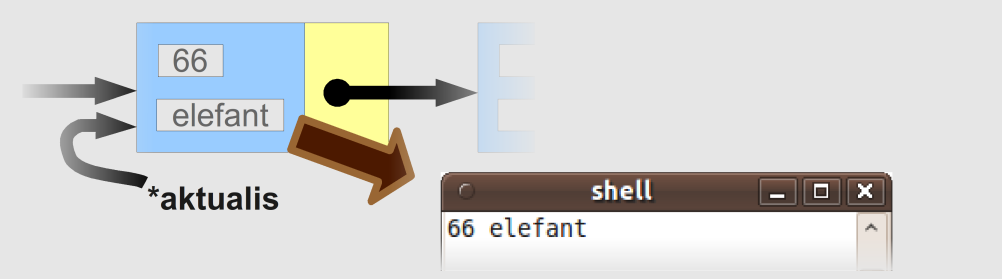
Például akkor van ere szükség, ha ki akarunk keresni a listából egy elemet, vagy ki akarjuk íratni az összes elemet a képernyőre.

A lista elemeit kiíratő kód:

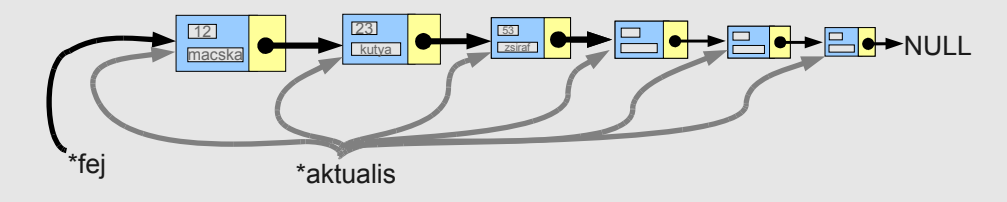
```
aktualis=fej;
```




```
while(aktualis!=NULL)
{
    printf("%d %s\n", aktualis->szam, aktualis->szoveg);
```




```
    aktualis=aktualis->kovetkezo;
```

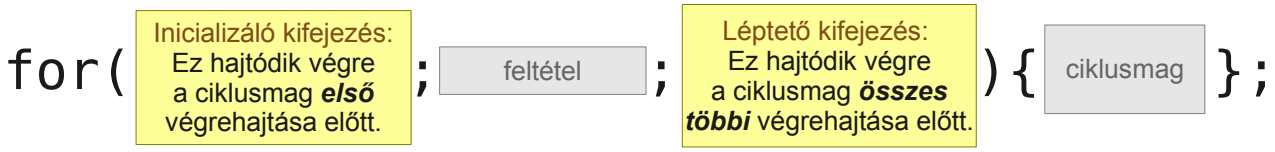



```
}
```

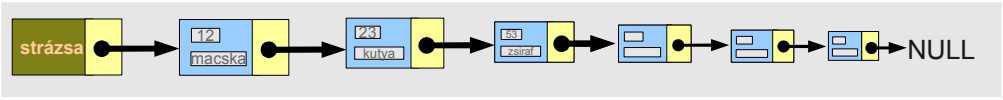
A fent megírt kódokat azonban lehet rövidebben is. Ezt szeretném megmutatni ezen az oldalon:

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

typedef struct lista_struct
{
	int szam;
	char szoveg[10];
	struct lista_struct *kovetkezo;
} lista_tipus;

int main()
{
	lista_tipus *fej;
	lista_tipus *aktualis;
	int beolvasott_szam;
	char beolvasott_szoveg[10];
	FILE *f;

	//Fajlbol valo beolvasas és lista felepitese
	f=fopen("adat.txt", "r");
	fej=NULL;
	while(fscanf(f,"%d %s",&beolvasott_szam,beolvasott_szoveg)!=EOF)
	{
		if(fej==NULL) fej=aktualis=(lista_tipus*)malloc(sizeof(lista_tipus));
		else aktualis=aktualis->kovetkezo=(lista_tipus*)malloc(sizeof(lista_tipus));
		aktualis->kovetkezo=NULL;
		//Az uj elem ertekeinek beallitasa a beolvasott ertekek alapjan
		aktualis->szam=beolvasott_szam;
		strcpy(aktualis->szoveg,beolvasott_szoveg);
	}

	//Az adatok kiiratasa a listabol a kepernyore
	for(aktualis=fej;aktualis!=NULL;aktualis=aktualis->kovetkezo)
	{
		printf("%d %s\n", aktualis->szam, aktualis->szoveg);
	}
}
```

A megértéshez érdemes ismerni a *for* működését:



Ha akarunk, alkalmazhatunk egy úgynevezett strázsát, vagyis egy olyan elemet, ami nem tartalmaz lényeges adatot, de a lista első elemeként fog szolgálni (a fej helyett).



Lehet, hogy ez a kód könnyebben érthető azok számára, akik az előzőket még nem értették meg.

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

typedef struct lista_struct
{
	int szam;
	char szoveg[10];
	struct lista_struct *kovetkezo;
} lista_tipus;

int main()
{
	lista_tipus strazsa; //A strazsa nem pointer, ezert mar itt letrejon
	lista_tipus *aktualis;
	int beolvasott_szam;
	char beolvasott_szoveg[10];
	FILE *f;

	//Fajlbol valo beolvasas és lista felepitese
	f=fopen("adat.txt", "r");
	aktualis=&strazsa; //A strazsatol kezdjuk felepiteni a listat
	while(fscanf(f,"%d %s",&beolvasott_szam,beolvasott_szoveg)!=EOF)
	{
		//Hely foglalasa uj elemnek
		//es legyen az uj elem az aktualis
		aktualis=aktualis->kovetkezo=(lista_tipus*)malloc(sizeof(lista_tipus));

		//Az uj elem legyen a lista vege
		aktualis->kovetkezo=NULL;

		//Az uj elem ertekeinek beallitasa a beolvasott ertekek alapjan
		aktualis->szam=beolvasott_szam;
		strcpy(aktualis->szoveg,beolvasott_szoveg);
	}

	//Az adatok kiiratasa a listabol a kepernyore
	//(a strazsat kihagyjuk, utana kezdjuk el a kiiratast)
	for(aktualis=strazsa.kovetkezo;aktualis!=NULL;aktualis=aktualis->kovetkezo)
	{
		printf("%d %s\n", aktualis->szam, aktualis->szoveg);
	}
}
```


Retzler Andris SCH1702
retzlerandras@gmail.com