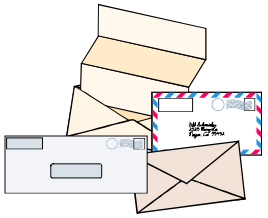
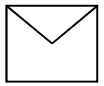
Fájlkezelés

- ***A fájlkezelés alapjai***
- ***Bináris fájlok kezelése***
  - *Logikai fájlnév definiálása*
  - *Bináris fájl megnyitása*
  - *Bináris fájl írása, olvasása*
  - *Pozicionálás a fájlban*
  - *A fájl lezárása*
- ***Szöveges fájl használata***
  - *Szövegfájl írása, olvasása*
  - *Mintaprogram szöveges fájl kezelésére*

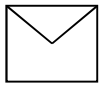


A fájlkezelés alapjai



A C nyelvben megkülönböztetik az alacsonyszintű és a magasszintű fájlkezelést. Mi a továbbiakban csak a kevésbé gépközeli és kevésbé gyors, de jobban hordozható magasszintű fájlkezelést vizsgáljuk.

*A C fájl egyetlen karaktervektor alakjában képzelhető el (stream = adatfolyam), amelynek bejárásához a fájlmutatót használhatjuk. A **fájlmutató** aktuális pozíciójától kiírhatunk a fájlba, vagy beolvashatunk a fájlból egy adott méretű adatot, és ezzel a méretnek megfelelő karakterszámmal (byte-számmal) előre mozgatjuk a fájlmutatót, vagy adatmozgás nélkül önmagában is pozicionálhatjuk azt. Írásra megnyitott fájl esetén a pozicionálás és írás a fájl korábbi végén túl is történhet, definiálatlanul hagyva a közbenső meg nem írt területek tartalmát.*

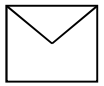


A létrehozható fájlok eltérhetnek a numerikus értékek tárolási módjában:

*a **text típusú fájlok**ban a számértékek a számjegykarakterek sorozataként, bármely szövegszerkesztővel olvasható és módosítható módon tárolódnak, szemben*

*a **bináris fájlok**kal, amelyekben a memóriabeli, kettes számrendszeren alapuló, kettő hatványait jelentő bitekből összeálló byte-ok adják a tárolandó karaktereket.*

"AC" 01000001 01000011

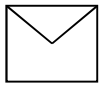


Bináris fájlok kezelése

A kezelés lépései:

- *Logikai fájlnév (fájlpointer) definiálása*
- *A fájl megnyitása*
- *A fájltartalom kezelése: írás a fájlba, olvasás a fájlból, pozícionálás a fájlban*
- *A fájl lezárása.*

A fájlok kezelése során számtalan hibalehetőség adódhat, amelyekből eredő programmegszakítást a hibakezelés beépítésével többnyire elkerülhetjük.



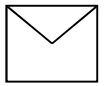
Logikai fájlnév definiálása

A logikai fájlnév szolgál a programírás során a fájl azonosítására. Egyidejűleg több fájlt is definiálhatunk. A definiálás formája:

FILE * <log_fájlnév>;

A <log_fájlnév> *által mutatott* FILE *struktúra-típusú struktúra adattagjai szolgálnak majd a fájlhasználat adminisztrálására.*

Pl.: FILE * *allomany;*
FILE * *fajl;*

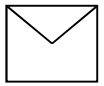


Bináris fájl megnyitása

A megnyitás szolgál a konkrét adatfájl esetleg csak futásidőben kiderülő fizikai nevének és a program megírása közben a fizikai név helyettesítésére szolgáló logikai fájlnévnek az összekapcsolására, valamint a megnyitás módjának a megadására.

A módban jelezzük azt is, hogy bináris fájlt nyitunk meg.

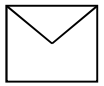
Formája: <log_fájlnév> = fopen(<fiznév>, <módstr>);



A módsztring lehetséges értékei:

"rb" létező fájl megnyitása olvasásra
"wb" új fájl létrehozása írásra, vagy létező fájl felülírása
"ab" létező fájl megnyitása hozzáírásra, vagy új fájl létrehozása, ha nem létezett
"r+b" létező fájl megnyitásra írásra és olvasásra
"w+b" új fájl létrehozása írásra, vagy olvasásra. Már létező fájl tartalma előbb törlődik.
"a+b" a fájl megnyitása írásra, vagy olvasásra, a fájl végére állított fájlmutatóval

A b karakter utal a fájl bináris jellegére.



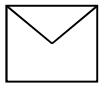
Példák fájlnyitásra:

```
allomany = fopen( "C:\\adatok\\nyilvant.dat","rb" );
fajl = fopen( fiznev,"wb");
```

Megj.: *a "wb" és "w+b" esetekben a már létező fájl tartalma jelzés nélkül elvész!*

A + jellel megnyitott fájloknál írás és olvasás váltása között a pufferelt adatokat célszerű kiíratni a fájlba az fflush(<log_név>) függvény, vagy pozicionáló függvény hívásával.

Sikertelen megnyitás esetén az fopen() *függvény* NULL *pointerértéket ad.*



Bináris fájl írása, olvasása

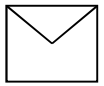
Változók tartalmának fájlbaírására és beolvasására szolgáló függvények:

fwrite(&<változó>, <változó mérete>, <darab>,<log_fájlnév>)
fread(&<változó>, <változó mérete>, <darab>,<log_fájlnév>)

A függvények visszatérési értéke a hibátlanul kiírt, ill. beolvasott értékek darabszáma.

Pl.:

```
fwrite(&valt, sizeof valt, 1, allomany);
fwrite(vektor, sizeof (vektor [0]), 10, allomany);
fread( &adag, sizeof (adag) , 1 , fajl );
```



Pozicionálás a fájlban

A pozicionálás alapvetően byte-okban történik, és a fájlmutató értékének lekérdezése is byte-okban mért pozíciót ad vissza, ***long int*** *értékkel.*

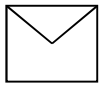
A 0. kezdőelemre való pozicionálás függvénye a

rewind(<log_fájlnév>) .

Tetszőleges byte-pozícióra az

fseek(<log_fájlnév>, <mennyit>, <honnan>)

függvénnyel állhatunk. A <mennyit> *érték byte-okban adja meg azt az értéket, amelyet a* <honnan> *által beazonosított kezdőpozícióhoz adva megkapjuk a célpozíciót.*



A <honnan> előredefiniált értékei:

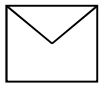
SEEK_SET *a fájl elejétől számítva*
SEEK_CUR *az aktuális pozíciótól mérve*
SEEK_END *a fájl végéhez adva.*

Az aktuális pozíciót az

ftell(<log_fájlnév>)

függvénnyel kérdezhetjük le.

Pl.:
```
rewind(allomany);
fseek( fajl, 12*sizeof(adag), SEEK_CUR );
pozicio = ftell( fajl );
```



A fájl lezárása

az

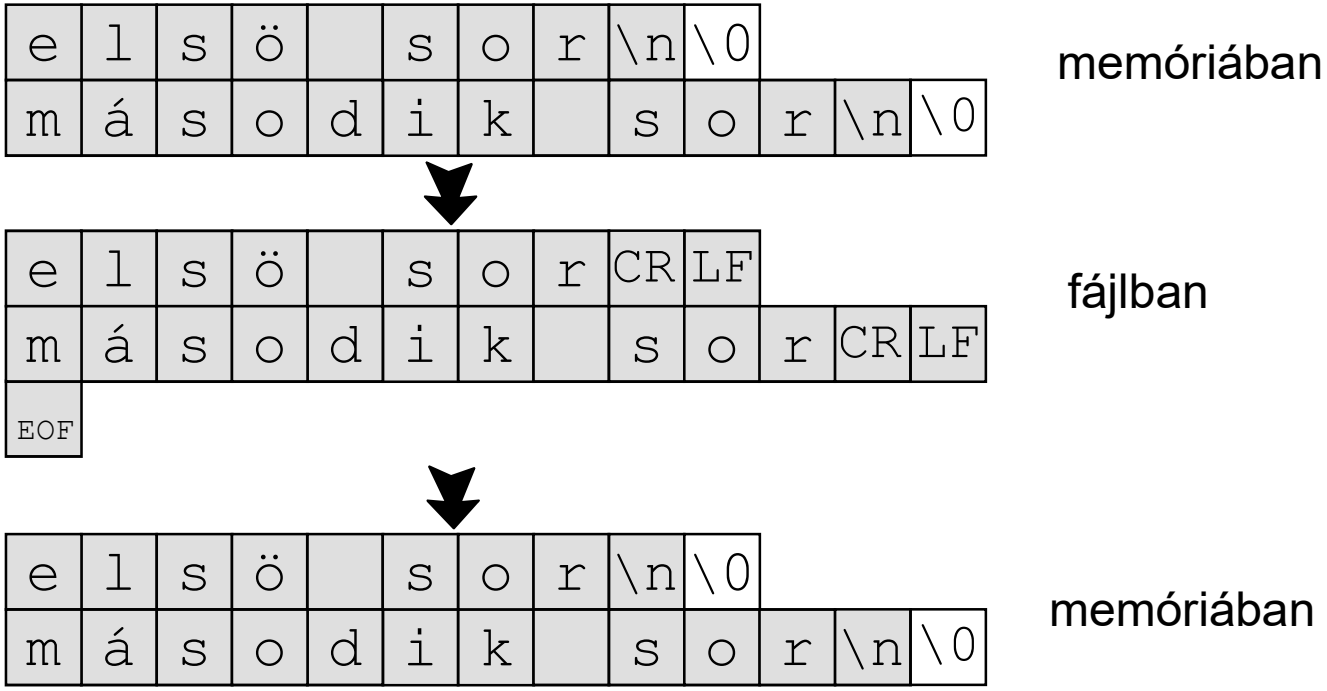
fclose(<log_fájlnév>)

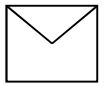
függvénnyel történik.

Pl.: fclose(*allomany*); fclose(*fajl*);

Szöveges fájl használata

Szövegek fájlbaírásakor, ill. visszaolvasásakor automatikus LF⇒CR+LF⇒LF konvertáláson megy át az újsor (\n) karakter:



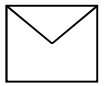


A szöveges fájlok kezelése a megnyitásban és a fájl írásában-olvasásában tér el a bináris fájlok kezelésétől. A megnyitásnál a módot megadó szövegben b helyett t írandó. Ezzel a mód lehetséges értékei: "rt", "wt", "at", "r+t", "w+t", "a+t".

Példa szövegfájl megnyitására:

```
textfajl = fopen( "C:\\texts\\level.txt", "wt" );
Ncrfajl = fopen( regifnev, "rt" );
```

Szövegfájl írása, olvasása



Karakterenkénti írás és olvasás a

putc(<karakter>,<log_fájlnév>)
getc(<log_fájlnév>)

makrókkal, szövegsorok (karaktervektorok) fájlbaírása és beolvasása az

fputs(<karaktervektor>, <log_fájlnév>)
fgets(<karaktervektor>, <pufferméret>, <log_fájlnév>)

függvényekkel, valamint elsősorban numerikus adatok szöveges alakúra és vissza konvertálásával a printf() és a scanf() mintájára használható

fprintf(<log_fájlnév>, <formátumsztring>, <kifejezéslista>)
fscanf(<log_fájlnév>, <formátumsztring>, <változócím-lista>)

függvényekkel végezhető.

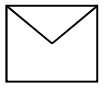
Mintaprogram szöveges fájl kezelésére

Számjegyvezérlésű marógép tabulátorokkal formázott NC-programjából kell a tabulátorokat kiszedni és szóközzel helyettesíteni.

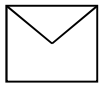
```
N    G    X    Y    Z    M    S    F
%
N001→G00→   →   →Z250.
N002→   →X120.→Y65.7
N003→   →   →   →Z22. →M03→S33
. . .
```

```
%
N001 G00 Z250.
N002 X120. Y65.7
N003 Z22. M03 S33
. . .
```

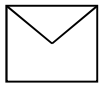
Szövegfájl írása, olvasása



```
#include <stdio.h>
main()
{
   FILE * NCrfajl, * NCufajl;
   char regifnev[79], ujfnev[79];
   char rsor[80],usor[80];
   int i, j;
   printf("Tabulatorjelek helyettesitese szokozzel\n");
   printf( "A forras NC program neve:" );
   gets( regifnev );
   NCrfajl = fopen( regifnev, "rt" );          /* megnyitás olvasásra  */
   if ( NCrfajl == NULL )
   {
            printf( "Sikertelen a régi fájl megnyitása" );
      exit(1);
   }
```



```
printf( "Az uj NC program neve:" );
gets( ujfnev );
 NCufajl = fopen( ujfnev, "wt");          /* megnyitás írásra    */
 if (NCufajl == NULL)
    {
      printf( "Az új fájl nem hozható létre." );
      exit(-1);
    }
```



```
while ( !feof( NcrfajI ) )                    // fájlvég tesztelése
{
  fgets(rsor, 80, NCrfajl);                   //  sor beolvasása
  i = 0; j = 0;
 while ( rsor[ i ] != '\0' )
  {
    if ( rsor[ i ] != '\t' )                  //  '\t' a tabulátorkarakter
        {
          usor[ j ] = rsor[ i ]; i++; j++;
        }
    else
        {
         usor[ j ] = ' '; j++;
          while (rsor[ i ] == '\t' )   i++;
        }
   }
  usor[ j ] = '\0';                                  // sor lezarasa
  printf("%s", usor );
  fputs(usor, NCufajl);                       // új sor kiírása a fájlba
 }
 fclose(NCrfajl);
 fclose(NCufajl);
}
```