Programozás C nyelven (15. ELŐADÁS)

Sapientia EMTE

2015-16

Makrók

- `#define <szimbólum> <karaktersor>`
  - A pre-processzor átvizsgálja a kódot soronként, és minden szimbólumot lecserél a megadott karaktersorra

```
#define TRUE 1
#define FALSE 0
#define YES TRUE //két lépésben cserélődik le
```

- Függvényszerű makrók

```
#define abs(x) ( (x)<0 ? (-(x)) : (x) )
#define min(a,b) ( (a)<(b) ? (a) : (b) )
#define negyzet(x) ( (x)*(x) )
```

Makrók (példák)

- `#define abs(x) ( (x)<0 ? (-(x)) : (x) )`

  …

  `alfa = abs(y + 2);`

  - 1. lépésben: `alfa = ( (x)<0 ? (-(x)) : (x) );`
  - 2. lépésben: `alfa = ( (y+2)<0 ? (-(y+2)) : (y+2) );`

- `#define negyzet(x) ( (x)*(x) )`

  …

  `beta = negyzet(++a);`

  - lecserélés után: `beta = ( (++a)*(++a) );`

Makrók vs. függvények

- Gyorsabb kódot eredményeznek, mint a függvények használata
  - a függvényhivások a futási időből vesznek el, míg a makróhívások már az előfordítás alatt lecserélődnek
- „Típus-függetlenség”
  - mivel egyszerű szöveghelyettesítésről van szó, ezért „univerzalisak”.

További részletek végett lásd a jegyzet 10. fejezetét

```
# define csere(a,b) a=a+b; b=a-b; a=a-b;

void rendez(int n, int a[])
{
   int i,j;
   for(i=0; i<n; i++)
      for(j=i+1; j<n; j++)
         if(a[i]>a[j]) {csere(a[i], a[j])}
}
```

Állománykezelés

- INPUT: billentyűzet/állomány
  - A billentyűzet is felfogható *bemeneti* állományként, amely a begépelt karaktereket tartalmazza
- OUTPUT: monitor/nyomtató/állomány
  - A monitor/nyomtató is tekinthetők *kimeneti* állományoknak
- A háttértárakon levő állományok lehetnek úgy *bemeneti*-ek, mint *kimeneti*-ek

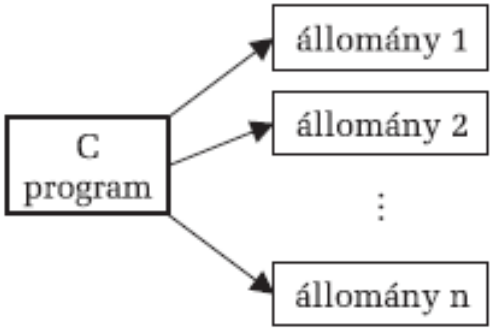
Input device

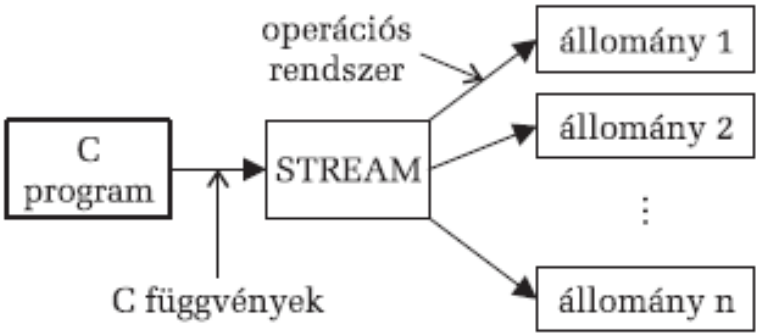
Output device

Input/output device

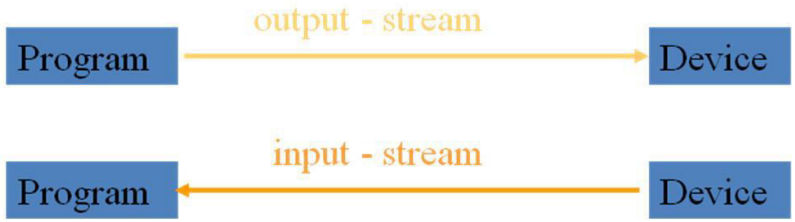
STREAM-ek

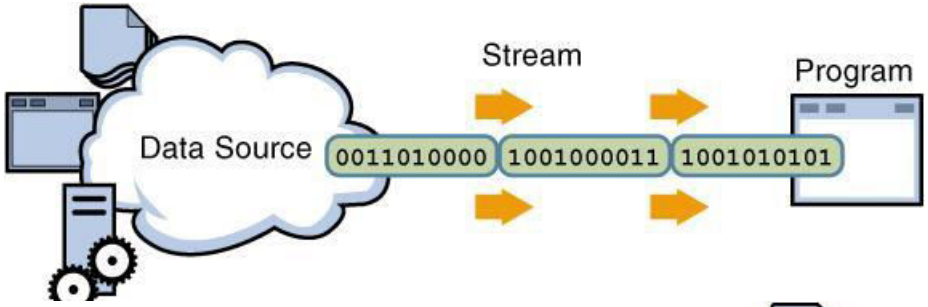
A STREAMek elrejtik az állományok terhes sokféleségét

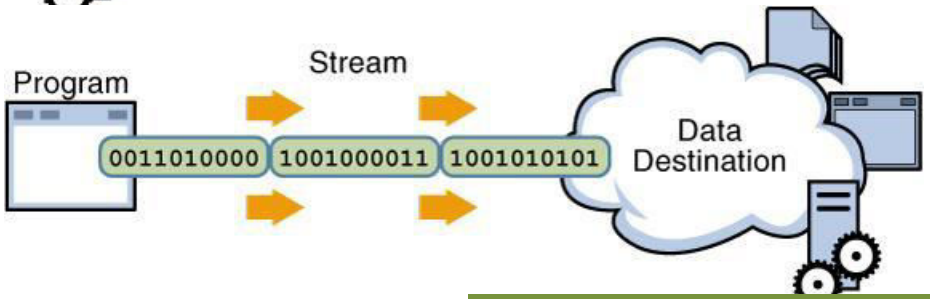




STREAM: *irányított* adat-fluxus

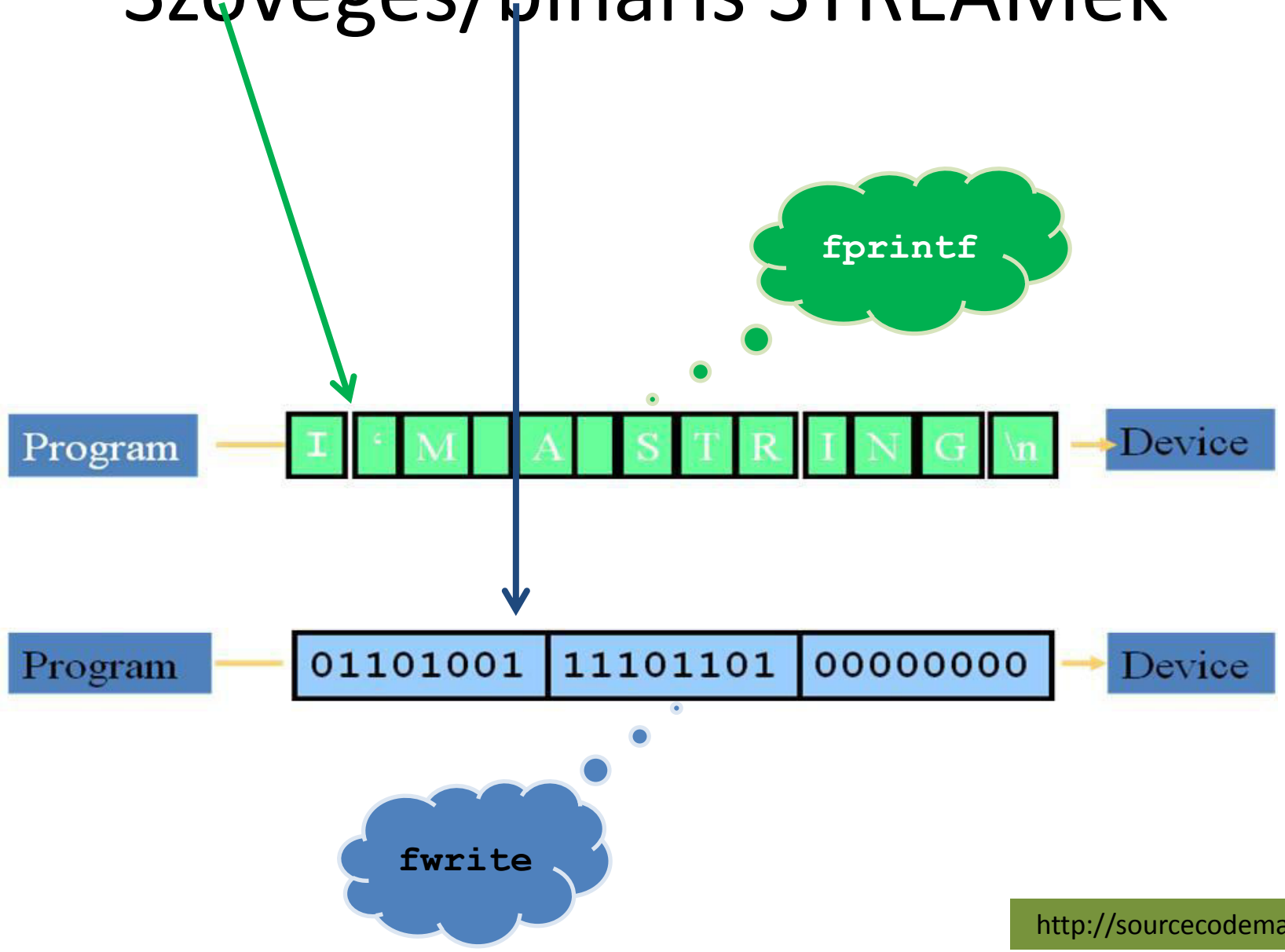






http://sourcecodemania.com

Szöveges/bináris STREAMek



Állománykezelő függvények

```
FILE * <állomány_pointer>;
```

```
fp = fopen(<állomány_név>, <mód>);
```

rt,r	szöveges mód, olvasás
wt,w	szöveges mód, létrehozás és írás
at,a	szöveges mód, írás a végére
rb	bináris mód, olvasás
wb	bináris mód, létrehozás és írás
ab	bináris mód, írás a végére
r+t,r+	szöveges mód, olvasás/írás
w+t,w+	szöveges mód, létrehozás és írás/olvasás
a+t,a+	szöveges mód, írás a végére vagy létrehozás és írás/olvasás
r+b	bináris mód, olvasás/írás
w+b	bináris mód, létrehozás és írás/olvasás
a+b	bináris mód, írás a végére vagy létrehozás és írás/olvasás

Ekkor kerül kijelölésre egy STREAM

...

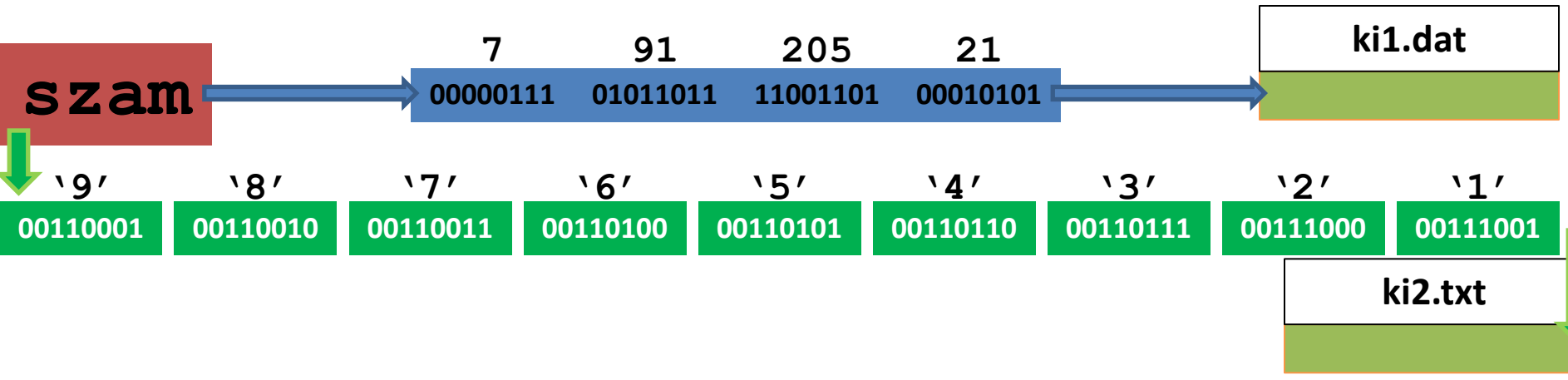
```
int fclose(FILE * fp);
```

Olvasás/Írás állományból/ba

- Olvasás/Írás *szöveges* állományból/ba
  - `fscanf / fprintf`
- Olvasás/Írás *bináris* állományból/ba
  - `size_t` **`fread`**`(void *`**`változócím`**`,`
    `size_t` **`elemméret`**`,`
    `size_t` **`elemszám`**`,`
    `FILE *`**`állományptr`**`);`
  - `size_t` **`fwrite`**`(const void *` **`változócím`**`,`
    `size_t` **`elemméret`**`,`
    `size_t` **`elemszám`**`,`
    `FILE *` **`állományptr`**`);`

fread / fwrite

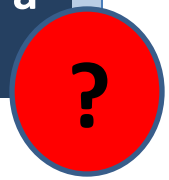
```
int szam = 123456789;
FILE *fout1 = fopen("ki1.dat", "wb");
FILE *fout2 = fopen("ki2.txt", "wt");
fwrite(&szam, sizeof(int), 1, fout1);
fprintf(fout2, "%i", szam)
fclose(fout1);
fclose(fout2);
```



fread / fwrite

```
int a[] = {1,2,3,4,5};
int n = sizeof(a) / sizeof(a[0]);
FILE *fout1 = fopen("ki1.dat", "wb");
FILE *fout2 = fopen("ki2.txt", "wt");
fwrite(a, sizeof(int), n, fout1);
for( i = 0 ; i < n ; ++i ){
   fprintf(fout2, "%i", a[i]);
}
fclose(fout1);
fclose(fout2);
```

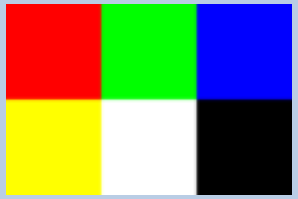
Hány bájt méretűek lesznek a kimeneti állományok?



fread / fwrite

```
FILE *fin = fopen("poszter.ppm", "rb");
char line[100];
int x, y, c; //szelesseg, magassag, maxszin

fscanf(fin, "[^\n]\n", line); //magic number (P6)
fscanf(fin, "[^\n]\n", line); //skip comment
fscanf(fin, "%i %i %i\n", &x, &y, &c);

unsigned char *kep;
kep = (unsigned char*)malloc(x*y*3*sizeof(unsigned char));

fread(kep, sizeof(unsigned char), x*y*3, fin);

fclose(fin);
```

FF0000 00FF00 0000FF FFFF00 FFFFFF 000000



Olvassuk be szöveges állományból n könyv adatai(cím, szerző, év), majd mentsük ki az adatbázist egy bináris állományba. Ezt követően olvassuk vissza az adatbázist és írassuk ki képernyőre.

```
typedef struct{
   char cim[30];
   char szerzo[20];
   int ev;
} KONYV;
```

```
FILE *text, *bin;
text = fopen("konyvek.txt", "rt");
bin = fopen("konyvek.dat", "wb");
int n,i;
fscanf(fin, "%i\n", &n);
KONYV *a = (KONYV*)malloc(n*sizeof(KONYV));
for( i = 0 ; i < n ; ++i ){
   fscanf("%s %s %i\n", a[i].cim, a[i].szerzo, &a[i].ev);
}
fwrite(a, sizeof(KONYV), n, bin);
free(a); fclose(text); fclose(bin);

bin = fopen("konyvek.dat", "rb");
KONYV *a = (KONYV*)malloc(n*sizeof(KONYV));
fread(a, sizeof(KONYV), n, bin);
for( i = 0 ; i < n ; ++i ){
   printf("C:%s Sz:%s E:%i\n", a[i].cim, a[i].szerzo, a[i].ev);
}
free(a); fclose(bin);
```

Egyéb állománykezelő függvények

- Karakterek ki/be
  - `fgetc /fputc`
- Karakterlánc ki/be
  - `fgets /fputs`

Karakter billentyűzetről:
- `getch, getche, getchar`

Karakter képernyőre:
- `putchar`

Karakterlánc billentyűzetről:
- `gets`

Karakterlánc képernyőre:
- `puts`

- `ferror, **feof**, rewind, **fseek**, ftell, **fflush**`
- **`remove, rename`**

Pufferelt fájlkezelés

ISMÉTLÉS

További részletek végett lásd a jegyzet 11. fejezetét

- Szöveges/bináris STREAMek
- Olvasás/Írás szöveges állományokból
  - `fscanf/fprintf`
- Olvasás/Írás bináris állományokból
  - **`fread/fwrite`**
- Egyéb függvények (például)
  - `feof, fflush, fseek, remove, rename`