12. gyakorlat
Sturktúratömbök, fájlkezelés
Mátrix műveletek

Struktúra tömb, dinamikus memóriafoglalás

1. Hozzunk létre egy 10 elemű könyvtárat. Definiáljuk a megadott függvényeket. Egy struktúratömbbe gyűjtsük ki a megadott időszakban megjelent műveket, majd irassuk ki azokat.

```
typedef struct book {
        char cim[50];
        char szerzo[50];
        int ev;
        float ar;
} Book;
typedef struct {
    Book* gyujtemeny;
    int darab;
} Selection;

void beolvas(Book * konyvtar, int meret);
void kiir(Book * konyvtar, int meret);
Selection kivalogat(Book * konyvtar, int meret, int tol, int ig);

int main() {
    int tol = 1995, ig = 2000;
    const int meret = 10;
    Book konyvtar[meret];
    beolvas(konyvtar, meret);
    kiir(konyvtar, meret);
    Selection s = kivalogat(konyvtar, tol, ig);
    kiir(s.gyujtemeny, s.darab);
    free(s.gyujtemeny);
    return 0;
}

Selection kivalogat(Book * konyvtar, int meret, int tol, int ig) {
    Selection s;
    int i, j, db = 0;
    for (i=0; i<meret; i++) {
        if (konyvtar[i].ev >= tol && konyvtar[i].ev <= ig)
            db++;
    }
    Book* valogatas = (Book *)malloc(db*sizeof(Book));
    if (valogatas == NULL) {
        printf("\nNincs eleg memoria!\n");  exit(-1);
    }
    j=0;
    for (i=0; i<meret; i++) {
        if (konyvtar[i].ev >= tol && konyvtar[i].ev <= ig)
            valogatas[j++] = konyvtar[i];
    }
```

```
        s.darab = db;
        s.gyujtemeny = valogatas;
        return s;
}
```

Fájlkezelés

2. Az előző feladatot oldjuk meg úgy, hogy a beolvasott adatokat tároljuk fájlban (a szöveges és a bináris fájlkezelést is próbáljuk ki). Listázzuk a fájl tartalmát, majd innen visszaolvasva történjen meg a kiválogatás. A kiválogatás alapjául szolgáló időszakot ellenőrzötten olvassuk be.

```
#define FILENAME_T "konyvtar.txt"
#define FILENAME_B "konyvtar.bin"

int main() {
        filebaolvas(FILENAME_T, 't');
        filebolkiir(FILENAME_T, 't');

        int tol, ig;
        intervallumbeolvas(&tol, &ig);
        Selection s = kivalogat(FILENAME_T, 't', tol, ig);
        structkiir(s.gyujtemeny, s.darab);
        free(s.gyujtemeny);
        return 0;
}

void filebaolvas(char* filename, char filetype) {
        int i, db;
        FILE *fp;
        Book konyv;

        //Fájl megnyitás
        if (filetype=='t')   fp=fopen(filename,"at");
        else  fp=fopen(filename,"ab");
        if (!fp) { printf("Error: cannot open file."); return; }
        printf("Hány adatot szeretne felvinni? ");
        intbeolvas(&db);

        for (i = 0; i < db; i++) {
                printf( "\nA könyv címe: "); scanf( "%s", konyv.cim );
                printf( "A könyv szerzője: "); scanf( "%s", konyv.szerzo );
                printf( "A könyv megjelenési éve: "); scanf( "%d", &konyv.ev );
                printf( "A könyv ára: "); scanf( "%f", &konyv.ar );

                if (filetype=='t') {
                /* A struktúra adattagjait egy sorban tároljuk */
                /* A sor végi újsor karaktert is el kell tárolni */
                fprintf(fp, "%s %s %d %.2f\n", konyv.cim, konyv.szerzo, konyv.ev, konyv.ar);
                }
                else  fwrite(&konyv, sizeof(Book), 1, fp);
        }
        fflush(fp);
```

```
            fclose(fp);
            return;
    }
    void filebolkiir(char* filename, char filetype) {
        int i, db=0;
        FILE *fp;
        Book konyv;
        char ch;

        if (filetype=='t') {                    //szövegfájl
            fp=fopen(filename,"rt");
            if (!fp) { printf("Error: cannot open file."); return; }

            while((ch = fgetc(fp)) != EOF) {  if(ch == '\n') db++;  }
            if (db==0) {  printf("\nFile is empty ...\n");  return ;  }

            rewind(fp);      /* Fájl elejére pozicionálás */
            for (i=0; i<db; i++) {
                fscanf(fp, "%s %s %d %f", konyv.cim, konyv.szerzo, &konyv.ev, &konyv.ar);
                printf("\nCím: %s, Szerző: %s, Kiadási év: %d, Ár: %.2f",
                    konyv.cim, konyv.szerzo, konyv.ev, konyv.ar);
            }
        } else {                                //bináris fájl
            fp=fopen(filename,"rb");
            if (!fp) { printf("Error: cannot open file.");  return; }
            fseek(fp, 0L, SEEK_END);
            db = ftell(fp)/sizeof(Book);
            if (db == 0) {  printf("\nFile is empty.\n");  return; }

            for (i=0; i<db; i++) {
                fseek(fp,sizeof(Book)*i,SEEK_SET);
                fread(&konyv,sizeof(Book),1,fp);
                printf("\nCím: %s, Szerző: %s, Kiadási év: %d, Ár: %.2f",
                    konyv.cim, konyv.szerzo, konyv.ev, konyv.ar);
            }
        }
        fclose(fp);
        return ;
    }
```

Mátrix műveletek

3. Töltsünk fel két mátrixot véletlenszámokkal és jelenítsük meg. Adjuk össze a két mátrixot.

```
void matrix_feltolt(int* mx, int sor, int oszlop);
void matrix_kiir(int* mx, int sor, int oszlop);
int * matrix_osszead(int* mx1, int* mx2, int sor, int oszlop);

int main(){
   int matrix1[3][2];
   int matrix2[3][2];
   int* eredmeny;
```

```
    srand(time(0));
    matrix_feltolt((int*)matrix1,3,2);
    matrix_feltolt((int*)matrix2,3,2);
    matrix_kiir((int*)matrix1,3,2);
    matrix_kiir((int*)matrix2,3,2);
    eredmeny=matrix_osszead((int*)matrix1,(int*)matrix2,3,2);
    matrix_kiir((int*)eredmeny,3,2);

    free((int*)eredmeny);
    return 0;
}

void matrix_feltolt(int* mx, int sor, int oszlop) {
    int i, j;
    for(i=0; i<sor; i++) {
        for(j=0; j<oszlop; j++)
            mx[i*oszlop+j] = (rand()%100)/10+1;   /*1...10*/
    }
}

void matrix_kiir(int* mx, int sor, int oszlop) {
     int i, j;
     for(i=0; i<sor; i++) {
          for(j=0; j<oszlop; j++)
               printf("%3d ", mx[i*oszlop+j]);
          printf("\n");
     }
     printf("\n");
}

int* matrix_osszead(int* mx1, int* mx2, int sor, int oszlop) {
     int i, j;
     int* e;
     /*lefoglaljuk az eredménymátrixnak megfelelo méretu területet*/
     e=(int*)calloc(sor*oszlop, sizeof(int));
     if(e==NULL){
          printf("Nincs elég memória!");
          return NULL;
     }
     for(i=0; i<sor; i++)
          for(j=0; j<oszlop; j++)
               e[i*oszlop+j]= mx1[i*oszlop+j] + mx2[i*oszlop+j];
     return e;
}
```

Házi feladatok

1. A beolvasott könyv adatok eltárolása fájlban, majd a fájl kilistázása. A fájlból visszaolvasva az adatokat végezzük el az alábbi feladatokat, amelyeket külön függvényként implementálunk.
1) Számoljuk meg hány könyv található a könyvtárban.
2) Összesítsük a könyvtárban található könyvek árát.

3) Keressük meg a legrégebbi / legújabb könyvet.
4) Döntsük el h. egy adott című könyv szerepel-e a könyvtárban.

2. Hozzon létre egy autó adatait tárolni képes struktúrát.
a) Hozzon létre egy 10 autó adatait tároló dinamikusan lefoglalt tömböt. Definiálja a struktúrán elvégezhető alábbi műveleteket: beolvasás, kiírás, 15%-os árleszállítás, a legdrágább/legolcsóbb autó megkeresése, autók átlagárának kiszámítása, megadott ár-értékhatárok közé eső autók megszámolása (pl.: 2 000 000 Ft és 3 000 000 Ft).
b) Tárolja el fájlban a beolvasott adatokat. Listázza a fájl tartalmát, és az előző műveleteket a fájlból visszaolvasás után valósítsa meg.
c) Módosítsa az autó struktúra típust: az árat összeg és devizanem formájában akarjuk megadni (beágyazott struktúra) (pl.: 20 000 Euro, 2 000 000 Ft). Ennek megfelelően módosítsa a programban definiált funkciókat.

3. Az Akasztófa játék programját írja át úgy, hogy a kitalálandó szavak fájlban legyenek eltárolva és innen válasszon egyet a program véletlenszerűen. Ezután tegye a játékot végtelen működésűvé, ami akkor áll le, ha a felhasználó már nem akar tovább játszani. Tárolja el fájlban a játékmenet eredményeit (hány kérdésből találta ki a felhasználó a megoldást). Az eredményekhez egy fájlt használjon, amihez minden kör végén hozzáírja az új eredményt.

4. A 12. előadás anyaga alapján készítsen mátrix műveleteket végző programot.
- Két mátrix összeadása.
- Két mátrix szorzása.
- Mátrix transzponálása (sorok és oszlopok felcserélése).
- Mátrix determinánsának kiszámítása.

5. Aknakereső játék. Egy 10x10-es játékmezőn helyezzünk el véletlenszerűen 20 darab, nem szomszédos aknát. A felhasználó 10szer tippelhet. Minden tipp után mondjuk meg, hogy talált-e. Számoljuk, hány aknát talált meg összesen.

6. Készítse el a kő-papír-olló játék programját. Használjon enum típusú változót a válaszlehetőségek tárolásához. A felhasználó játszik a géppel. A program akkor álljon le, amikor a felhasználó már nem akar tovább játszani. Minden kör végén írja ki, hogy ki nyert. A játék legvégén pedig legyen statisztika: ki hányszor nyert.

Segítség:

```
enum tipp {KO, PAPIR, OLLO};    //KO=0, PAPIR=1, OLLO=2

do {
   printf("Mit választasz?");    // felhasználó válaszát beolvasni, eltárolni 'valasz' változóban
   gep = rand()%3; //0-2 közötti véletlenszám

   // eldönteni ki nyert, tárolni
   // eredményt kiírni

   printf("Akarsz még játszani? ");
   scanf(" %c", &tovabb);
} while (tovabb=='i' || tovabb=='I');

// statisztika
```