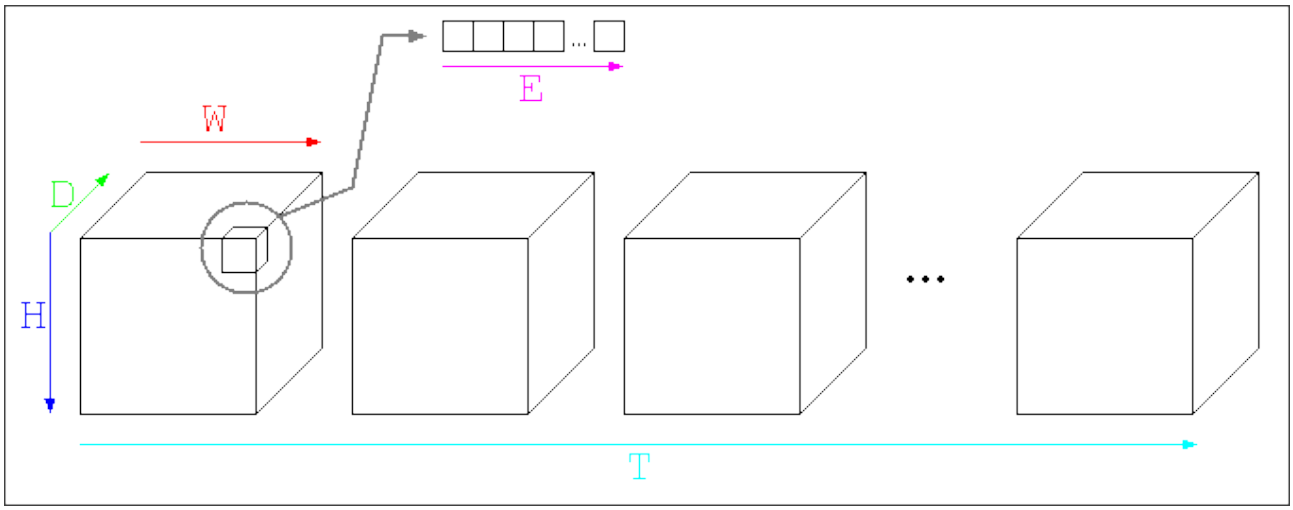
10. Gyakorlat
A `kobject` struktúra

A VisiQuest-ben magunk is létrehozhatunk képfeldolgozó műveleteket. A műveleteknek általában van egy vagy több inputja és van egy vagy több outputja. A dobozok inputja és outputja egy kobject struktúrában tárolódik el. A struktúra ismeri az attribútumokat és tárolja képi adatot is. Az adat tulajdonképpen egy többdimenziós tömb. (A valóságban lehet, hogy egydimenziós vektorban van emulálva, de kezelését tekintve többdimenziós.)



Az adatstruktúra rendelkezik szélességi (W), magassági (H), mélységi (D), időbeli (T) paraméterekkel, valamint egy extra dimenzióval (E), amit mondjuk színes képek vagy pedig többdimenziós adatok tárolására használhatunk fel. Vegyünk egy példát! Ha van egy kétdimenziós képünk, akkor annak a szélessége és magassága valamilyen pozitív érték. Mivel kétdimenziós ezért a mélysége 1, mivel állókép, ezért az időbeli dimenziója 1, és attól függően, hogy szürkeárnyalatos vagy színes képről van szó,az extra dimenzió lehet 1 vagy 3.
A kobject struktúra nem csak képeket tud tárolni, hanem bármilyen adatot, mondjuk függvényeket[1], vagy görbét is. Ekkor meg kell határozni, hogy mely dimenziókat használjuk. Pl. függvény esetében használhatjuk az időt és az extra dimenziót. Nincs szükség a másik három dimenzió használatára, tehát azok értéke 1.

A `kbject` struktúrát kezelő függvények:

```
kobject in_object = NULL;
kobject out_object = NULL;
```

- `in_object = kpds_open_input_object(clui_info->i_file)`: megnyitja az input file-t és betölti az `in_object` objektumba.
- `out_object = kpds_open_output_object(clui_info->o_file)`: megnyitja az output file-t és betölti az `out_object` objektumba.
- `kpds_close_object( kobject obj )`: lezárja az objektumot.
- `kpds_create_value( kobject obj )`: minden kobject objektumnak rendelkeznie kell egy value szegmenssel, ami az adatot tárolja. Ez a függvény üres objektumhoz hoz létre egy üres value szegmenst.
- `int kpds_copy_object( kobject from, kobject to )`: egy objektumot átmásol egy másikba, ekkor a két objektum minden attribútuma és szegmense megegyezik. A visszatérési érték azt jelzi, hogy sikerült-e a másolás.

1 Példákat találtok a http://www.cab.u-szeged.hu/local/doc/khoros/Tutorial/snippets1.html weboldalon.

- **kpds_set_attribute**(kobject obj, *ATTRIBÚTUM*, *ÉRTÉK(EK)*): ezzel a függvénnyel állítható be egy adott kobject objektum valamely attribútuma.
  Attribútumok:
  - KPDS_VALUE_DATA_TYPE: adattípus (KUBYTE, KUINT, KINT, ...)
  - KPDS_VALUE_SIZE: az adat mérete (minden dimenzóban)
  - KPDS_VALUE_POSITION: pozícionálás az adatban
  - KPDS_HISTORY: a history string beállítása (a program végén)
- **kpds_get_attribute**(kobject obj, *ATTRIBÚTUM*, *ÉRTÉK(EK)*): ezzel a függvénnyel kérdezhető le egy adott kobject objektum valamely attribútuma.
- **kpds_get_data**(kobject obj, *ADAT_FLAG*, *VÁLTOZÓ*): az ADAT_FLAG típusú adat lekérése az VÁLTOZÓ-ba. A változó típusának meg kell egyeznie a Value szegmens típusával!
- **kpds_put_data**(kobject obj, *ADAT_FLAG*, *VÁLTOZÓ*): az ADAT_FLAG típusú adat másolása az VÁLTOZÓ-ból. A változó típusának meg kell egyeznie a Value szegmens típusával!

Program készítése VisiQuest-ben:

Lássunk egy C nyelvű mintaprogramot, amely bemutatja, hogy hogyan is kell kezelni a kobject objektumokat! FONTOS, hogy a VisiQuest már előállít egy programkódot, amit nekünk kell kiegészítenünk.

```
int run_mythreshold(void)
{
   /*-- Put Your Code Here --*/

   /* -main_variable_list */
   char lib = "mythreshold_obj"; /* ez a kerror() függvényhez kell */
   char rtn = "main";            /* ez a kerror() függvényhez kell */
   kobject in_object = NULL;     /* input objektum, inicializálva */
   kobject out_object = NULL;    /* output objektum, inicializálva */
   unsigned char *plane;         /* tömb az input adat számára */
   unsigned char *res_plane;     /* tömb az eredménykép számára */
   int w, h, d, t, e;            /* ezekbe a változókba kérdezzük le az attribútumokat */
   int cw, ch, ct, pos;          /* segédváltozók */
/* -main_variable_list_end */

/* -main_before_lib_call */
   /* megnyitjuk az input fájlt és az adatot bemásoljuk az input struktúrába
      le kell ellenőrizni, hogy sikerült-e megnyitni a fájlt, és átmásolni a struktúrát,
      különben ki kell lépni a programból
   */
   if ((in_object = kpds_open_input_object(clui_info->i_file))
         == KOBJECT_INVALID) {
     kerror(lib, rtn, "Can not open input object %s.\n", clui_info->i_file);
     kexit(KEXIT_FAILURE);
   }

   /* megnyitjuk az input fájlt és az adatot bemásoljuk az input struktúrába
      le kell ellenőrizni, hogy sikerült-e megnyitni a fájlt, és átmásolni a struktúrát,
      különben ki kell lépni a programból
   */
   if ((out_object = kpds_open_output_object(clui_info->o_file))
        == KOBJECT_INVALID) {
     kerror(lib, rtn, "Can not open output object %s.\n", clui_info->o_file);
     kexit(KEXIT_FAILURE);
   }

   /* átmásoljuk az input képet az outputba, ezzel egyúttal be is állítjuk minden
      szegmensét és attribútumát
   */
```

```
   if (!kpds_copy_object(in_object, out_object)) {
     kerror(lib, rtn, "Can not copy input object to output object.\n");
     kexit(KEXIT_FAILURE);
   }

   /* beállítjuk mindkét objetum adattípusát UNSIGNED BYTE-ra */
   kpds_set_attribute(in_object,  KPDS_VALUE_DATA_TYPE, KUBYTE);
   kpds_set_attribute(out_object, KPDS_VALUE_DATA_TYPE, KUBYTE);

  /* lekérdezzük a változókba a méret attribútumokat */
     kpds_get_attribute(in_object, KPDS_VALUE_SIZE, &w, &h, &d, &t, &e);

  /* létrehozunk egy tömböt a képi adat számára. A tömb egydimenzós lesz, mérete
     megegyezik a kép méretével.
  */
  plane = (unsigned char *)kmalloc(w*h*sizeof(unsigned char));
  res_plane = (char *)kmalloc(w*h*sizeof(unsigned char));
  if (!plane || !res_plane) {
     kerror(lib, rtn, "Could not allocate memory for the image\n");
     kexit(KEXIT_FAILURE);
  }
/* -main_before_lib_call_end */

/* -main_library_call */
  /* Itt kezdődik a képfeldolgozó művelet érdemi része. Végighaladunk a képen, és
     küszöböljök a paraméterként (clui_info->threshold változó) megadott értékkel.
     Ami ez alatti intenzitás, az 0 lesz, az e fölöttiek pedig 255 értéket kapnak.
     Most csak az általunk lefoglalt memória területen dolgozunk, nem manipuláljuk
     a kobject strukúrák adatait.
  */
  for (ct = 0; ct < t; ct++) {
    /* pozícionálás: a ct-edik kép kezdőpozíciójára állunk és kimásoljuk a képi adatot a
       segédtömbbe
    */
    kpds_set_attribute(in_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_get_data(in_object, KPDS_VALUE_PLANE, (kaddr)plane);
    kmemset(res_plane, 0, w*h*sizeof(char)); /* kinullázzuk az eredmény képet */

    for (ch = 0; ch < h; ch++) {
      for (cw = 0; cw < w; cw++) {
        pos = ch*w + cw;               /* Az adott pozíciónak megfelelő tömb indexet
                                          így számoljuk ki */
        if (plane[pos] <= clui_info->threshold_int)
          res_plane[pos] = (unsigned char)0;
        else
          res_plane[pos] = (unsigned char)255;
      }
    }

    /* beállítjuk a pozíciót úgy, hogy a struktúrán belüli mutató a képi adat
       ct-edik képének kezdőpozíciójára mutasson és bemásoljuk erre a pozícióra az
       eredmény képet
    */
    kpds_set_attribute(out_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_put_data(out_object, KPDS_VALUE_PLANE, (kaddr)res_plane);
  }
/* -main_library_call_end */

/* -main_after_lib_call */
   /* beállítjuk a history stringet (ezt mindig ugyanígy kell csinálni ) */
   if (!kpds_set_attribute(out_object, KPDS_HISTORY, kpds_history_string())) {
     kerror(lib,rtn,"Unable to set history on the destination object");
     kexit(KEXIT_FAILURE);
   }

   /* felszabadítjuk a lefoglalt memóriaterületet */
   if (plane)
     kfree(plane);
   if (res_plane)
     kfree(res_plane);
```

```
   /* a program végén lezárjuk az objektumokat, ezek a függvények elvégzik a struktúrán
      belüli memóriafelszabdítást is.
   */
   kpds_close_object(in_object);
   kpds_close_object(out_object);
/* -main_after_lib_call_end */

   return TRUE;
}
```

Feladatok:

1. Készítsd el egy szürkeárnyalatos kép inverzét! (Egészítsd ki a myinverse.c forrást!)
2. Transzponáld a szürkeárnyalatos input képet! (Egészítsd ki a mytranspose.c forrást!)

A megoldások az óra után kerülnek fel a weboldalra.