**Füzesiné Dr Hudák Mariann**

**I97122**

**A programozás alapjai 1 – Nagy HF programozói dokumentáció**

**Képfeldolgozás**

A program a felhasználó által kiválasztott *.ppm formátumú képfájlokat olvas be, dolgoz fel, és ment el szintén *.ppm formátumban, a felhasználó által megadott néven.

A program képes a képek kontrasztját, világosságát, telítettségét a felhasználó által megadott mértékben és irányban módosítani; a képet a felhasználó által megadott mértékben elhomályosítani, valamint a képen található éleket megjeleníteni.

**Adatszerkezetek, modulok**

A program fő lépéseit a **main.c** modul, és az abban található main() függvény vezérli, a részlépéseket pedig a **control modul (control.c, control.h).** A dokumentáció a mainből történő hívások, azaz a program működésének sorrendjében mutatja be az egyes modulokat, függvényeket, adatszerkezeteket.

Induláskor a program bekér egy fájlnevet a felhasználótól. A main függvény a **control.c**-ben definiált **void filenev_beker(char* filename)** függvényt hívja, amely egy paraméterként kapott stringbe bekéri a felhasználótól a megnyitandó fájl nevét, és ellenőrzi, hogy *.ppm kiterjesztésű fájl-e. Ha nem, újra bekéri a fájlnevet, amíg a felhasználó *.ppm fájlt ad meg.

A fájlok mentésekor is ugyanez a függvény kéri be a felhasználótól a fájlnevet az elmentendő kép számára.

Ezután a fájl beolvasása a **readimage (readimage.c, readimage.h)** modulban található **PixelData** readfile(char *filename, short* errorcode, ImageParams *imgdata)** függvény hívásával történik.

A **PixelData** egy ebben a modulban definiált típus, az adott pixel R, G és B értékeit tárolja short-ként (0-255 lehet az rgb érték). A kép összes pixelének az adatai egy dinamikusan foglalt 2 dimenziós PixelData tömbben tárolódnak. A függvény ennek a tömbnek a pointerét adja vissza. Az **ImageParams** szintén ebben a modulban definiált típus; a képfájl fejlécéből beolvasott, a képfeldolgozás szempontjából releváns információt tárolja el. A **readfile**

függvény a bináris fájlt karakterenként olvassa be. Ellenőrzi a *.ppm fájlok kódolására szolgáló P6 karakterek meglétét (enélkül nem érvényes *.ppm fájlt nyitottunk meg, amit a program vissza is jelez). Ezután ellenőrzi, hogy van-e komment a fejlécben, és ha van, azt kiszűri. A kép felbontásának és bitmélységének adatait az ImageParams típusú imgparams nevű cím szerinti paraméterben tárolja. A függvény egy dinamikusan foglalt 2 dimenziós PixelData tömbbe olvassa be a pixelek r, g és b értékeit a bináris fájlból, majd ezen mátrix pointerével tér vissza. Ha nem sikerül a beolvasandó mátrixnak memóriát foglalni, a paraméterébe nem 0 hibakód kerül, kiírja a hibát a képernyőre, és visszatér a **main**-be. Ha minden rendben ment, a hibakód paraméterbe 0 kerül, és a mainbe visszaadja az rgb mátrixot.

A main ezután a **control** modulban található **processimage(&imgdata, matrix, filename)** modult hívja (while ciklusban, mivel lehetséges, hogy a felhasználó több módosítást is el akar végezni a képen). Átadja a felbontás információ (imgdata) beolvasott mátrix pointerét, valamint a fájlnevet. A függvény 0-át ad vissza, ha rendben lefutott. Ha nem, a main hibaüzenetet ír ki, és 2-es hibakóddal tér vissza.

A **processimage** függvény először megjeleníti a főmenüt **a menu.c és menu.h** modulban lévű **displaymain()** függvény hívásával, majd a szintén a menu modulban található MainMenu enum típusú option változóba bekéri a menu modul selectmain függvényével, hogy a felhasználó melyik képmódosítási opciót választotta a lehetőségek közül. Ennek függvényében hívja a megfelelő függvényeket.

**EXIT:** Ha a kilépést (EXIT) választotta a felhasználó, felszabadítja az rgb adatokat tároló kétdimenziós tömböt, és kilép a programból.

**Contrast** választása esetén először a HSV színtérre konvertálja át a program az RGB mátrixot. Ez a formátum külön tárolja a színárnyalat (hue, h), a telítettség (saturation, s) és a világosság (value, v) értékeket.

A konverzió a **HSV_RGB** modulban található **HSV** RGBtoHSV(const ImageParams* imgdata, const PixelData** matrix)** fügvénnyel történik.

Ez egy ebben a modulban definiált típusú, dinamikusan foglalt 2 dimenziós HSV mátrixot ad vissza. Ez minden pixel Hue, Saturation és Value értékét tartalmazza double típusú változóban. A konverzió algoritmusa ismert algoritmus, forrásként a

https://www.geeksforgeeks.org/program-change-rgb-color-model-hsv-color-model/?tab=article címen leírtakat és a https://lodev.org/cgtutor/color.html oldalon található kódrészletet használtam fel az **RGBtoHSV** és a **HSVtoRGB** függvények megírásához.

Miután az RGBtoHSV elvégzi a konvertálást, és visszaadja a HSV mátrixot a ProcessImage függvénynek, a ProcessImage a **setcontrast(ImageParams const *imgdata, HSV **HSVmatrix)** függvényt hívja meg, amely a **brightness** modulban található. Ez 2 segédfüggvénnyel megkeresi a kép Value értékeinek a minimumát és a maximumát (**minval** és **maxval** függvények hívásával, átadva a kép felbontását és HSV mátrixát). A setcontrast függvény kép minden pixelének Value értékét az új minimum és maximum tartományba normalizálja. Ezek értékei a felhasználó által megadott százalékos érték alapján kerülnek kiszámításra a függvényben, melyet a getpercent függvény kér be a felhasználótól, és szükség esetén korrigálja a -100..+100 tartományba, figyelmeztetve erre a felhasználót is. A felhasználó -100 és +100 között adhatja meg a kontraszt módosításának kívánt mértékét. Miután a függvény elvégezte a kontraszt módosítását a paraméterként kapott HSV mátrixon, a processimage függvény a **HSVtoRGB** függvényt hívja meg, amely visszakonvertálja RGB formátumba az adatokat, a paraméterében lévő eredeti PixelData mátrixot felülírva. Ehhez a konvertáló függvényhez adaptáltam kódrészletet az itt linkelt forrásból: https://lodev.org/cgtutor/color.html. A HSV mátrixot itt felszabadítja a processimage; az rgb formátumú mátrix még megmarad további feldolgozásra vagy fájlba írásra (ezt a program futásának legvégén, amikor a felhasználó nem akar további módosítást vagy mentést végezni az adott képen, a main fogja felszabadítani).

**Brightness**

Ha a processimage-ben a brightness opció került kiválasztásra, szintén meghívásra kerül az RGBtoHSV függvény, amely visszaadja a HSV mátrixot a processimage-nek, amely ezután a brightness modulban található **void setbrightness(ImageParams const *imgdata, HSV **HSVmatrix)** függvénynek adja át paraméterként a kép felbontási adataira mutató pointerrel együtt. Ez a pixelek value értékét a felhasználó általt kívánt -100 és +100 közötti mértékben módosítja. A processimage ezután meghívja a HSVtoRGB függvényt, hogy visszakonvertálja rgb értékekké a módosított mátrixot, majd a HSV mátrixot felszabadítja.

**Saturation**

A processimage ennek az opciónak a választásakor meghívja az RGBtoHSV-t a konverzió elvégzésére, majd a brightness modulban található **void setsaturation(ImageParams const *imgdata, HSV **HSVmatrix)** függvénynek adja át a képparaméterek és a HSV mátrix pointerét. A függvény a felhasználó által kívánt mértékben (-100..+100) mértékben módosítja a HSV mátrix s értékeit. A processimage ezután meghívja a HSVto RGB függvényt, hogy visszakonvertálja rgb értékekké a módosított mátrixot, majd a HSV mátrixot felszabadítja.

Mindhárom függvény hivja a getpercent függvényt, amely automatikusan korrigálja a felhasználó által megadott mértéket -100 és 100 közé, amennyiben nem a tartományba eső értéket adott meg, és erről figyelmeztetést ír ki a szabványos kimenetre. Mindhárom a processimage függvénybe tér vissza.

**Blur image**

Ha a felhasználó ezt az opciót választja, a processimage a **filters** modulban található **void blurimage(ImageParams const *imgdata, PixelData **matrix)** függvényt hívja meg. Ez közvetlenül a paraméterként kapott RGB mátrixba dolgozik, nem kell hozzá előzetes HSV konverzió. A felhasználó 3 és 30 közötti filter méretet adhat meg a blur-öléshez. Először a függvény ezen érték szerint előállítja a szűrőt, amely a felhasználó által megadott méret pixelben. Ha páros számú méretet adott meg, a program egyet hozzáad a mérethez. Ha a tartományon kívüli méretet adott meg a felhasználó, a program automatikusan korrigálja, és erről figyelmeztetést ír ki a szabványos kimenetre. A szűrő közepének a súlya a szűrő mérete+1 fele lesz. A szűrő szélei felé pixelenként 1-gyel csökkennek a súlyok. A szűrő értékeit egy dinamikusan foglalt 2 dimenziós double mátrixban tárolja. A kép adott pixelének a szűrőével megegyező méretű környezetét, amelynél a konvolúció épp tart, szintén egy dinamikusan foglalt 2 dimenziós mátrixba másolja (mozgó ablak, **window**). Ezt minden pixel szűrésének elkészültével felszabadítja a ciklusban, és a következő pixel számára újra foglal.

A függvény ellenőrzi, hogy a kép széleinél tart-e. A széleknél a kódban megadott értékkel egészíti ki a mátrixot a függvény (jelen esetben 127, középszürke) a szűrő méretének megfelelően, a túlindexelés elkerülése érdekében.

A függvény elvégzi a konvolúciót a mátrix r, g és b értékeivel, amelyhez egy segédmátrixot foglal dinamikusan, melynek mérete megegyezik az eredeti mátrixéval. A konvolúció során a függvény ebbe az új mátrixba dolgozik. Végül visszamásolja a konvolvált mátrixot az eredetibe a konvolúció végeztével, és felszabadítja a szűrő mátrixát és a segédmátrixot, amibe a konvolúció dolgozott. A függvény szintén a processimage függvénybe tér vissza.

**Find edges**

Először **RGBtoHSV** konverzió történik, majd a **filters** modulban található **void findedges(ImageParams const *imgdata, HSV **HSVmatrix**) függvénynek adja át a processimage a HSV mátrixot és a kép paramétereit. Ez egy 3x3-as mátrix-szal szűri meg a képet, melyben a középső pixel súlya 8, a környező 8 db pixelé pedig egyenként -1, így a szűrő összege 0 (Laplace operátor). Ez a kép éleit emeli ki. A paraméterben kapott HSV mátrixba dolgozik, majd a konvolúció végeztével a processimage-be tér vissza. Itt meghívásra kerül a HSVtoRGB konvertáló függvény, mely az eredeti mátrixba visszakonverálja a feldolgozott kép rgb értékeit, további feldolgozáshoz vagy mentéshez.

A **processimage** 0-val tér vissza **main**-be, ha rendben megtörtént a feldolgozás (ha valamelyik mátrixhoz nem sikerült memóriát foglalni, akkor nem 0 hibakóddal tér vissza).

A main ez után a **control** modul **bool saveimage(ImageParams *imgdata, PixelData **matrix, char *filename)** függvényét hívja, amely egy bool-t ad a finished változóba. Ez akkor lesz igaz, ha a felhasználó már nem akar több módosítást a képen. Ha még szeretne módosítást, akkor hamis lesz az ide visszatérő érték. A main belső while ciklusának ez adja a ciklus feltételét: ha hamis, akkor újra hívja a process-image függvényt, és a felhasználó választhat újra képfeldolgozási módot a főmenüből, amelyet ugyanezen a képen tud elvégezni.

A **saveimage** függvény megjeleníti az almenüt (a menu modulan lévő **displaysubmenu()** meghívásával. Ezután a **selectsubmenu**() függvényt hívja, ami bekéri a választott opciót, ami egy SubMenu enum, a menu modulban definiálva. Ennek függvényében a felhasználó további módosításokat végezhet a képen (**further modification**), mentheti a képet más néven (**save as..)** vagy felülírhatja az eredetit (**save**). A further modification választásakor a már az eddigiekben módosított mátrix lesz az inputja a következő módosításnak. A program ekkor készít egy biztonsági mentést a jelenlegi állapotról „PROBA.ppm” néven. Ezen opció választása esetén hamissal tér vissza a main finished bool változójába, ami által a while ciklus

folytatódik, és újra választhat a felhasználó a kép módosítási lehetőségeinek közül. Ha a **save as.**.-t választja, meghívásra kerül a filename_beker függvény, ami bekéri az új fájlnevet, és ellenőrzi, hogy *.ppm formátumot adott-e meg a felhasználó. Ha a **save –**et választja, a program visszakérdez, hogy valóban felül szeretné-e írni a fájlt. Ha igen, az eredeti fájlnéven menti a képet. Ha nem, meghívja a filename_beker függvényt.

A save és a save as opció választásakor igaz értékkel tér vissza a main finished változójába, ezáltal a **main** kilép a belső while ciklusából, és felszababadítja az rgb mátrixot. Ekkor a main külső while ciklusa bekéri a felhasználótól, hogy szeretne-e újabb képet feldolgozni. Igen válasz esetén indul előről az egész folyamat. Nem válasz esetén a végéhez ér a program futása.

A memóriaszivárgás ellenőrzését az infoc.eet.bme.hu portálról letöltött **debugmalloc**.h modul végzi.