# OBJEKTNO ORIJENTISANO PROGRAMIRANJE

- domaći zadatak broj 1 -

## Funkcionalna specifikacija

Na programskom jeziku C++ implementirati statičku biblioteku (.lib) klasa za obradu slike. Potom napisati glavni program (kao konzolnu .exe aplikaciju) koji testira mogućnosti biblioteke.

### *Specifikacija klase ImageEditor*

Klasa `ImageEditor` predstavlja klasu za obradu slika koje se sastoje iz matrice *piksela*. Maksimalna veličina slike je $2^{32}$ x $2^{32}$ *piksela*. Detalji o formatu slike se nalaze dalje u tekstu.

#### Propisno stvaranje i uništavanje editora slike

Implementirati propisno stvaranje i uništavanje objekata klase `ImageEditor`.

#### Učitavanje slike

Implementirati operaciju `bool ImageEditor::loadImage(unsigned char* image);` koja učitava informacije o slici date niskom `image` po formatu datom u nastavku. Povratna vrednost je indikator uspešnosti učitavanja slike (povratna vrednost je `false` ukoliko niska nije korektno formatirana ili je došlo do neke druge greške). Ukoliko je prethodno već učitana slika, sve informacije o staroj slici se brišu i učitava se nova slika.

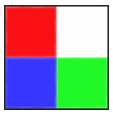
- Niska započinje dvoslovnom oznakom formata slike „`BM`“, nakon čega sledi ime slike koje je opciono. Ime slike se nalazi između dva znaka jednakosti (pr: „`=untitled=`“). Naredni deo informacija započinje na prvoj narednoj lokaciji niske koja je deljiva sa četiri, pa je niska potencijalno dopunjena proizvoljnim podacima do te lokacije.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| „B“ | „M“ | „=“ | „s“ | „l“ | „i“ | „k“ | „a“ | „=“ | fill | fill | fill | ... |

- Narednih osam bajtova niske sadrže informaciju o dimenzijama slike. Prva četiri bajta sadrže informaciju o širini slike u broju piksela, a druga četiri bajta sadrže informaciju o visini slike u broju piksela. Bajtovi koji sadrže informaciju o dimenziji slike su u niski raspoređeni tako da prvi bajt u niski odgovara najnižem bajtu dimenzije slike, a poslednji bajt u niski odgovara najvišem bajtu dimenzije slike (*little-endian*). Primer dela niske sa informacijama o dimenzijama slike čije su dimenzije 256x384 (0x00000100 = 256 piksela; 0x00000180 = 384 piksela):

| ... | 0x00 | 0x01 | 0x00 | 0x00 | 0x80 | 0x01 | 0x00 | 0x00 | ... |
|---|---|---|---|---|---|---|---|---|---|

- Ostatak niske čine informacije o svakom redu *piksela* slike. Prvi red piksela u niski odgovara donjem redu piksela sa slike, dok su pikseli poređani sleva udesno. Svaki red piksela u niski je potencijalno dopunjen proizvoljnim podacima tako da svaki novi red započinje na lokaciji u niski koja je deljiva sa četiri. *Piksel* sadrži intenzitete crvene, zelene i plave boje. Intenziteti boja su nenegativne veličine maksimalne vrednosti 255. Svaki bajt niske sadrži vrednost intenziteta jedne boje jednog piksela u redosledu plava (B) – zelena (G) – crvena (R). Primer dela niske sa informacijama o pikselima slike i dopunama čiji je rezultat dat sa strane (pikseli se popunjavaju od donjeg levog ugla po redovima):



| | B | G | R | B | G | R | | | B | G | R | B | G | R | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | 255 | 0 | 0 | 0 | 255 | 0 | un. | un. | 0 | 0 | 255 | 255 | 255 | 255 | un. | un. | ... |

#### Snimanje slike

Implementirati operaciju `unsigned char* ImageEditor::saveImage();` koja kao povratnu vrednost treba da vrati sliku koja je prethodno učitana u gorenavedenom formatu.

**Slojevi slike**

Editor slike ima mogućnost dodavanja novih slojeva (*Layer*) na sliku. Pri učitavanju slike kreira se nov sloj čija matrica piksela sadrži piksele učitane slike. Novokreirani sloj postaje aktivan sloj. Svi naknadno kreirani slojevi sadrže praznu matricu piksela u veličini učitane slike. Po praznim slojevima je moguće crtati. Svaki sloj ima procenat neprovidnosti čija je vrednost nenegativan ceo broj sa maksimalnom vrednošću 100. Prilikom čuvanja slike, pikseli u gornjim slojevima se iscrtavaju preko piksela u donjim slojevima. Ukoliko je neprovidnost sloja 100%, pikseli u tom sloju u potpunosti sakrivaju piksele u svim slojevima ispod. Ukoliko je neprovidnost sloja x%, pikseli u tom sloju učestvuju u formiranju boje piksela sa x%, dok pikseli u prvom sloju ispod učestvuju u formiranju boje piksela sa (100-x)% * y%, gde je y neprovidnost prvog sloja ispod.

Implementirati operaciju `void ImageEditor::addLayer();` koja dodaje novi sloj slike iznad aktivnog sloja. Novokreirani sloj postaje aktivan sloj.

Implementirati operaciju `void ImageEditor::deleteLayer();` koja briše aktivni sloj slike. Aktivan sloj postaje sloj koji se nalazio ispod izbrisanog sloja. Nije moguće izbrisati sloj sa učitanom slikom.

Implementirati operaciju `void ImageEditor::selectLayer(int i);` koja i-ti sloj iznad sloja sa učitanom slikom čini aktivnim slojem.

Implementirati operaciju `void ImageEditor::setLayerOpacity(int);` koja menja procenat neprovidnosti aktivnog sloja.

**Manipulacija slikom**

Implementirati operaciju `void ImageEditor::invertColors();` koja invertuje boje svih piksela aktivnog sloja. Invertne boje su boje čiji odgovarajući intenziteti u zbiru daju 255.

Implementirati operaciju `void ImageEditor::toGrayScale();` koja konvertuje boje svih piksela aktivnog sloja u nijanse sive boje po formuli: R' = G' = B' = 0,3 * R + 0,59 * G + 0,11 * B.

Implementirati operaciju `void ImageEditor::blur(int size);` koja zamućuje sve piksele aktivnog sloja tako što svaki piksel menja pikselom čiji su intenziteti boja dobijeni kao srednje vrednosti intenziteta boja piksela koji se menja i svih piksela koji se nalaze oko piksela koji se menja na udaljenosti od maksimalno `size` redova i `size` kolona.

Implementirati operaciju `void ImageEditor::flipHorizontal();` koja rotira piksele po y osi.

Implementirati operaciju `void ImageEditor::flipVertical();` koja rotira piksele po x osi.

Implementirati operaciju `void ImageEditor::crop(int x, int y, int w, int h);` koja odbacuje sve piksele svih slojeva koji se ne nalaze u delu slike širine `w` i visine `h` čiji je gornji levi ugao u koordinatama `x` i `y`. Dodatno se ažuriraju širina i visina slike.

**Crtanje po slici**

Implementirati operaciju `void ImageEditor::setActiveColor(string hex);` koja postavlja aktivnu boju za crtanje na boju zadatu kroz parametar `hex`. Parametar `hex` je formata „#RRBBGG“, gde su RR, BB i GG heksadecimalne vrednosti intenziteta crvene, plave i zelene boje. Podrazumevana aktivna boja prilikom kreiranja editora je crna (R = 0, G = 0, B = 0).

Implementirati operaciju `void ImageEditor::fillRect(int x, int y, int w, int h);` koja iscrtava pravougaonik aktivne boje širine `w` i visine `h` čiji je gornji levi ugao u koordinatama `x` i `y`.

Implementirati operaciju `void ImageEditor::eraseRect(int x, int y, int w, int h);` koja uklanja piksele koji se nalaze unutar pravougaonika širine `w` i visine `h` čiji je gornji levi ugao u koordinatama `x` i `y`.

*Test funkcija*

Javni test sadrži funkciju `void test();` koja testira obradu slika pozivajući funkciju `unsigned char* editImage(unsigned char* input);`. Studentima je javno dostupna implementacija funkcije `editImage` i mogu da je menjaju da bi dodatno testirali svoj kod kao što je opisano komentarima u kodu. Funkcija `test` vrši učitavanje slika u .bmp formatu i konvertuje ih u format koji je pogodan za objekte klase ImageEditor, zatim poziva funkciju `editImage` i na kraju kreira novu sliku u .bmp formatu nakon obrade. Funkcija test se nalazi u prevedenom test.obj fajlu tako da njena implementacija nije javno dostupna. Studentima su date slike koje predstavljaju ulazne test primere kao i slike koje predstavljaju očekivane odgovarajuće izlaze radi mogućnosti provere. Slike ulaznih test primera treba da budu smeštene u folder glavnog projekta.

## Tehnički zahtevi i smernice za izradu rešenja

Iz kolekcije standardne biblioteke dozvoljeno je koristiti SAMO tip podataka *string*. Za smeštanje informacija o slici NIJE DOZVOLJENO koristiti kolekcije iz standardne biblioteke, već isključivo ugrađene i sopstevene tipove podataka i strukture. Sve klase i operacije moraju biti imenovane prema zahtevima iz domaćeg zadatka. Voditi računa o rukovanju dinamičkom memorijom. Programski kod klasa rasporediti u odgovarajuće **.h** i **.cpp** fajlove. Nije dozvoljeno korišćenje globalnih promenljivih za razmenu podataka. Ukoliko u zadatku nešto nije dovoljno jasno definisano, treba usvojiti razumnu pretpostavku i na temeljima te pretpostavke nastaviti izgrađivanje svog rešenja.

**VAŽNE NAPOMENE**

Za uspešno odbranjen domaći zadatak potrebno je na odbrani pokazati kod podeljen na odgovarajuće projekte, .h i .cpp fajlove.

1. Klase kojima su implementirani osnovni koncepti treba da budu smeštene u poseban projekat rešenja koji se prevodi kao statička biblioteka (**`ImageEditor.lib`**).
2. Glavni program treba da se nalazi u posebnom projektu koji se prevodi kao Win32 Console Application (**`testdz1.exe`**) fajl i koji treba povezati sa statičkom bibliotekom. Glavni program je dat u fajlu **`Main.cpp`** koji je javno dostupan i kojeg treba uključiti u projekat. U glavni projekat je dodatno potrebno uključiti fajl **`Test.obj`**.
3. NIJE DOZVOLJENO SMESTITI CEO KOD U JEDAN PROJEKAT ILI CPP fajl!

21.10.2019. godine sa predmeta