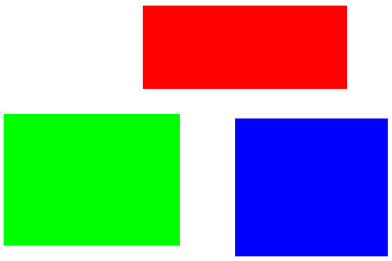
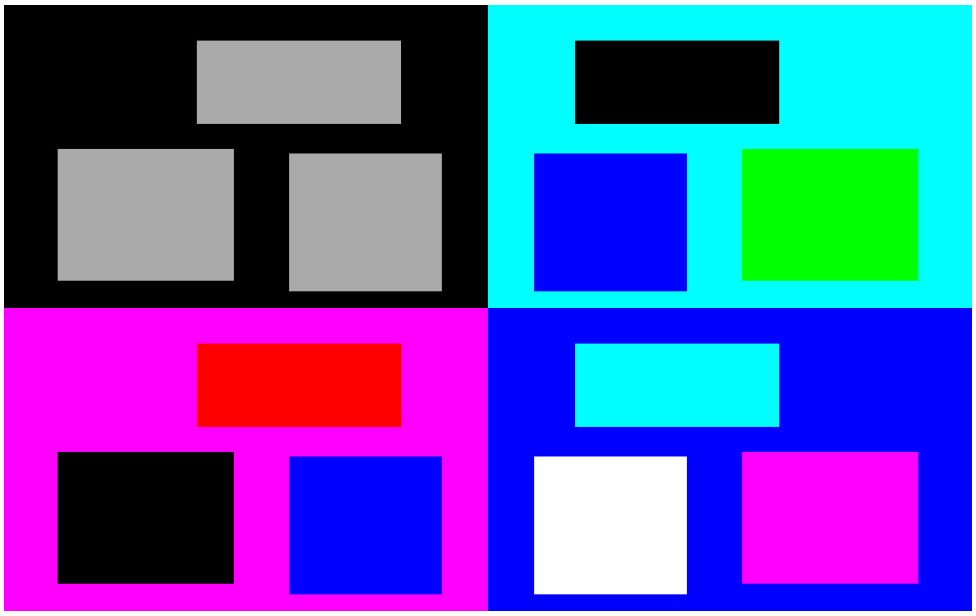
Programowanie IN & EnChJ – Projekt na zaliczenie eksternistyczne

Napisz program, który przetwarza grafikę zapisaną w pliku w formacie `.pgm` lub `ppm`[1] i tworzy nowy obraz składający się ze zmodyfikowanych kopii grafiki, ułożonych w siatkę 2×2 (w wersji rozszerzonej na ocenę 5! w siatkę $n \times n$).
Na przykład dla grafiki:



Rysunek 1: Przykładowa grafika

wynik może wyglądać tak:



Rysunek 2: Obraz 2×2 powstały w wyniku różnych transformacji grafiki z rysunku 1

Utworzony obraz $n \times n$ musi zostać zapisany w pliku na dysku. Wczytana grafika nie może być modyfikowana. Użytkownik powinien być proszony o podanie nazwy grafiki, która jest wczytywana wewnątrz funkcji. Rezerwacja miejsca na wczytywaną grafikę (podobnie jak na wszystkie inne tworzone obrazy) ma być dynamiczna, tak aby program obsługiwał grafiki o dowolnych rozmiarach. Program powinien automatycznie rozpoznawać format grafiki. Powinien takze sprawdzać poprawność wejściowych danych (w tym czy istnieje plik, który chcemy wczytać) i jeśli dane nie są poprawne, przerywać działanie ze stosownym komunikatem. Powinien przy wczytywaniu umieć pominąć komentarze zaczynające się od znaku # (w nagłówku grafiki).

[1] https://en.wikipedia.org/wiki/Netpbm_format, https://pl.wikipedia.org/wiki/Portable_anymap

- Wersja podstawowa - maksymalna ocena 5:
  Założyć, że $n = 2$ oraz, że ciągi transformacji prowadzące do utworzenia kolejnych zmodyfikowanych kopii obrazka są następujące[2] (licząc od obrazka w lewym górnym rogu i poruszając się wierszami od lewej do prawej):

  1. rzutowanie na szarości[3] $\longrightarrow$ negatyw
  2. odbicie w pionie $\longrightarrow$ usunięcie koloru czerwonego
  3. usunięcie zielonego
  4. odbicie w pionie $\longrightarrow$usunięcie niebieskiego $\longrightarrow$negatyw,

  przy czym program ma być napisany tak, aby można było łatwo zmodyfikować kod, wstawiając inne transformacje. Każda transformacja grafiki musi być realizowana przez wywołanie odpowiedniej funkcji. Program po uruchomieniu powinien prosić tylko o podanie nazw dwóch plików: z początkową grafiką i tego w którym zostanie zapisany obraz 2×2.

- Wersja rozszerzona - na 5!:
  Program po wczytaniu grafiki powinien pytać o wybór dalszego trybu procedowania. W trybie automatycznym powinien działać tak jak w wersji podstawowej. Tryb manualny oznacza, że program pyta użytkownika o wymiar grafiki $n \times n$ (np. $n = 3$ oznacza obraz 3×3 i tym samym 9 obrazków składowych, powstałych w wyniku różnych tranformacji początkowej grafiki) i o to, które transformacje ma wykonać, żeby powstał i-ty obrazek. Zachowanie funkcji usuwania kolorów ma być w trybie manualnym nieco inne niż w automatycznym: funkcja ma pytać o to, które spośród trzech kolorów usunąć, tzn. funkcja może usuwać więcej niż jeden kolor [4]. Możliwe ma być również nieusuwanie żadnego.

  W wersji zaawansowanej nie należy tworzyć osobnych funkcji do transformacji w trybie automatycznym i osobnych do trybu manualnego – funkcje mają mieć na swoich listach argumentów odpowiedni argument, którego wartość ma decydować w którym trybie funkcje mają pracować.

W programie należy zwalniać obszary pamięci możliwie wcześnie i nie dopuszczać do sytuacji, w której traci się informację do adresu początku obszaru przydzielonego dynamicznie.

Przed przeczytaniem dalszej części konieczne jest zapoznanie się ze specyfikacją formatów `ppm` i `pgm`: https://en.wikipedia.org/wiki/Netpbm_format,
https://pl.wikipedia.org/wiki/Portable_anymap .

Szczegóły algorytmu:

Liczba komórek zarezerwowanych na obraz w formacie `pgm` ma być 3 razy mniejsza niż dla obrazka o tych samych wymiarach, ale kolorowego. W rezultacie, jeżeli na grafikę $n \times n$ składają się tylko obrazy w szarościach, to będzie on zapisany w formacie `.pgm`. Jednakże, jeżeli część obrazków składowych jest kolorowa, a część szara, to te szare muszą być zamienione, na etapie scalania obrazków składowych, na kolorowe. Przyjąć tu regułę, że jeśli `t` oznacza nasycenie szarego piksela, a `x,y,z` nasycenia odpowiednika kolorowego, to:

$$x = y = z = t$$

Tworzenie obrazu $n \times n$ ma być realizowane w funkcji.

[2] przykład na rysunku 2

[3] szarości = grafika w odcieniach szarości, format `pgm`

[4] przyjąć, że usunięcie wszystkich kolorów ma skutkować zapamiętywaniem obrazu jako szarego

Transformacje:

- odbicie w pionie: pierwsza kolumna pikseli obrazu ma się znaleźć na miejscu ostatniej, druga na przedostatniej, itd.

- negatyw: jeśli oznaczyć przez d głębię obrazu (wczytywana z nagłówka pliku z grafiką) oraz przez x wartość nasycenia danego piksela (lub danego koloru danego piksela, w przypadku obrazka kolorowego), to w wyniku tej operacji nasycenie ma przyjąć wartość $d - x$. Oczywiście, należy zastosować tę procedurę do wszystkich pikseli obrazu

- szarości: jeśli przyjąć, że x, y, z to nasycenia kolorów danego piksela oraz t oznacza nasycenie piksela z grafiki w szarościach, to

$$t = \frac{x + y + z}{3}$$

- usunięcie czerwonego/zielonego/niebieskiego: funkcja realizująca tę transformację ma przyjąć kolor do usunięcia jako argument. Jeżeli np. x oznacza nasycenie składowej czerwonej danego piksela i wybieram do usunięcia ten właśnie kolor, to w obrazie po transformacji $x = 0$ (zgodnie z definicją formatu `.ppm`)