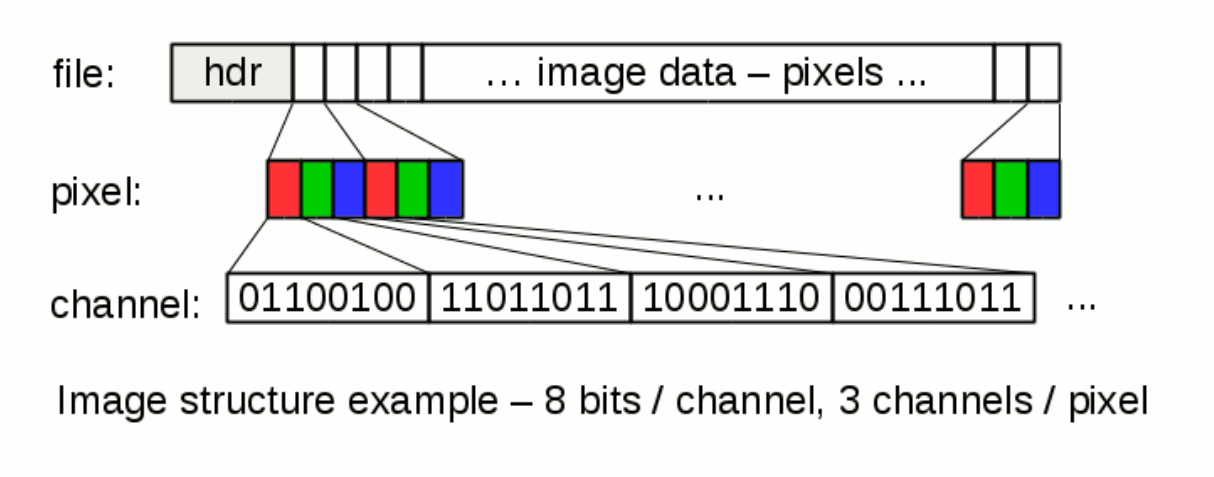
BI-PA2 (18/19 LS) ► Domácí úloha 01 ► Zrcadlení obrázků

Úkolem je vytvořit C/C++ funkci, která dokáže zpracovat obrázek uložený v souboru: načíst jej, volitelně jej zrcadlit a uložit zpět.

Pro tuto úlohu předpokládáme zjednodušené nekomprimované ukládání obrázku. Obrázek je vlastně 2D pole pixelů, tyto pixely lze nejsnáze ukládat po řádcích shora dolů, každou řádku pak zleva doprava. Velikost obrázku (šířka a výška) udává velikost tohoto 2D pole. Každý jeden pixel je tvořen několika kanály (např. jedním kanálem - odstínem šedé, třemi kanály - složkami RGB nebo čtyřmi kanály - složkami RGB a průhledností). Každý kanál je pak kódován jako celé číslo, kde počet bitů tohoto čísla určuje množství odstínů a barev pixelů, které jsme schopni do obrázku uložit. V našem příkladu budeme předpokládat varianty 1, 3 nebo 4 kanály na pixel a hodnoty kódující kanál mají 1, 8 nebo 16 bitů.



Jednou z nejjednodušších operací při zpracování obrázků je jejich zrcadlení - buď horizontální (překlopení podel svislé osy), nebo vertikální (podle vodorovné osy). Požadovaná funkce bude podle parametrů provádět zvolená zrcadlení: žádné, jedno z nich, nebo obě zároveň:

```
bool flipImage ( const char * srcFileName,
                 const char * dstFileName,
                 bool         flipHorizontal,
                 bool         flipVertical );
```

`srcFileName`

je ASCIIZ řetězec se jménem zdrojového souboru (zdrojový obrázek). Funkce tento soubor může číst, nesmí jej ale modifikovat.

`dstFileName`

je ASCIIZ řetězec se jménem cílového souboru (vytvořený obrázek). Funkce tento soubor vytváří, uloží do něj zdrojový obrázek překódovaný podle zbývajících parametrů.

`flipHorizontal`

udává, zda se má obrázek zrcadlit vodorovně (dle svislé osy),

`flipVertical`

udává, zda se má obrázek zrcadlit svisle (dle vodorovné osy),

návratová hodnota

`true` pro úspěch, `false` pro neúspěch. Za neúspěch považujte:

- chybu při práci se soubory (nelze číst, zapsat, neexistuje, ...),
- chybný formát vstupního souboru (nesprávný obsah hlavičky, nedovolený formát pixelů, nedostatek dat, nadbytek dat, ...).

Funkce podle parametrů připraví požadovaný cílový soubor. Cílový soubor bude mít následující vlastnosti:

- rozměr cílového obrázku bude odpovídat velikosti obrázku ze zdrojového souboru,
- formát pixelů cílového obrázku a kódování little/big endian bude odpovídat formátu pixelů ve zdrojovém obrázku,
- obsah obrázku je upraven podle parametrů flipHorizontal/flipVertical.

Obrázek je v souboru uložen velmi jednoduše. Soubor začíná hlavičkou popisující formát a velikost obrázku, za touto hlavičkou jsou uložená obrazová data. Hlavička obrázku má následující strukturu:

```
offset   velikost     význam
+0       2B           endian (0x4949 little endian, 0x4d4d big endian)
+2       2B           šířka uloženého obrázku
+4       2B           výška uloženého obrázku
+6       2B           formát pixelů (počet barevných kanálů a počet bitů na kanál)
+8       ??           vlastní obrazová data (pixely)
```

- Identifikátor little/big endian je první hodnota hlavičky. Udává pořadí ukládání bajtů pro dvou-bajtové hodnoty v souboru (tedy vztahuje se i na další údaje v hlavičce). Platné hodnoty jsou zvolené symetricky, tedy čtení tohoto identifikátoru správně zvládne libovolná platforma. V závazných testech bude pro platné soubory tato hodnota vždy odpovídat little endianu (tedy HW, na kterém poběží Váš program). V nepovinném testu se pak zkouší i vstupy v big-endianu.
- Šířka a výška udává rozměr obrázků (vždy v pixelech), výška ani šířka nesmí být nulová.
- Formát udává způsob kódování jednotlivých pixelů. Hodnota je složena z jednotlivých bitů, tyto bity mají následující význam:
- `bit  15            5 4 3 2 1 0`
- `     0 0    ...    0 B B B C C`

Pixely mohou být kódovány několika kanály. Bity 1 a 0 udávají počet kanálů na jeden pixel:

```
00 - 1 kanál: černo/bílý  nebo odstíny šedé
01 - nedovolená kombinace
10 - 3 kanály = RGB
11 - 4 kanály = RGBA
```

Každý kanál je přenášen s použitím zadaného počtu bitů:

```
000 -  1 bit na kanál
001 -  nedovolená kombinace
010 -  nedovolená kombinace
011 -  8 bitů na kanál
100 - 16 bitů na kanál
101 \
110  | nedovolené kombinace
111 /
```

Bity 5-15 jsou nevyužité, musí být nastavené na 0. Příklady kombinací:

```
0x000c = 0b00001100 - 1 kanál na pixel, kanál má 8 bitů,
0x000e = 0b00001110 - 3 kanály na pixel, každý kanál má 8 bitů,
0x0013 = 0b00010011 - 4 kanály na pixel, každý kanál má 16 bitů
```

Poznámky:

- Pečlivě ošetřujte souborové operace. Testovací prostředí úmyslně testuje Vaši implementaci pro soubory neexistující, nečitelné nebo soubory s nesprávným datovým obsahem.
- Jména souborů jsou řetězce, které nemusíte nijak kontrolovat. Názvy souborů ani přípony nejsou nijak omezené, podstatné je pouze, zda lze soubory daného jména otevřít a číst/zapisovat.
- Při implementaci lze použít C i C++ rozhraní pro práci se soubory.
- V přiloženém archivu najdete sadu testovacích souborů a jim odpovídající výstupy. Dále v přiloženém zdrojovém souboru naleznete ukázkovou funkci `main`, která spouští konverze na ukázkových souborech. Do tohoto zdrojového souboru můžete vložit Vaši implementaci, testovat ji a odevzdat ji na Progtest. Pokud chcete upravený zdrojový soubor odevzdávat, zachovejte v něm bloky podmíněného překladu.
- Pro manipulaci s formátem pixelů v hlavičce se hodí bitové operace (&) a bitové posuvy (<< a >>).
- Překódování little/big endian se uplatní na složky hlavičky a případně i na datový obsah obrázku, pokud se používá formát 16 bitů na kanál.
- Rozdělte řešení do více funkcí, případně si navrhněte třídu pro reprezentaci načteného obrázku. Nezkoušejte transformovat data přímo při kopírování ze souboru do souboru. Rozumné řešení načte celý zdrojový obrázek do 2D paměťové reprezentace, provede příslušné transformace a paměťovou reprezentaci uloží do cílového souboru. Paměti na to je dostatek.
- Pro zpracování hodnot se známou velikostí (například 2 bajty rozměru obrázku v hlavičce) se hodí datové typy s garantovanou velikostí (např. datové typy `int16_t`, `uint16_t` z hlavičkového souboru cstdint).
- Povinné testy pracují s obrázky ve formátu 8 bitů na kanál a se soubory ve formátu little endian (to odpovídá architektuře použitého procesoru). Pokud Vaše implementace dokáže s takovými vstupy

správně fungovat a pro ostatní nastavení (jiný endian, jiný počet bitů na kanál) vrátí neúspěch, projde povinnými testy.

- Další test (16 bit obrázky, little/big endian) je nepovinný. Jeho zvládnutí znamená, že můžete dostat až 100% bodů. Naopak, pokud jej Vaše implementace nezvládne, dojde ke krácení bodů. Nepovinnost testu znamená, že i pokud testem neprojdete vůbec, můžete dostat nenulové celkové hodnocení (pokud by byl povinný, pak by nezvládnutí testu znamenalo 0 bodů).
- Poslední test (1 bit obrázky) je bonusový. Jeho zvládnutí znamená, že dostanete body navíc (nad nominálních 100 %). Při reprezentaci 1 bit na kanál je potřeba při čtení souborů rozebírat čtené bajty na jednotlivé bity a naopak při zápisu skládat zapisované bity do celých bajtů. Bity v bajtu jsou obsazované v pořadí od LSB k MSB (least significant bit to most significant bit). Každá řádka obrázku vždy začíná na hranici bajtu. Tedy pokud celkový počet bitů na šířku není násobek 8, pak je potřeba poslední bajt na řádce doplnit nulovými bity (zero padding).
- Je rozumné začít s řešením základní varianty (8 bit/kanál, little endian) a tu rozšiřovat. Takový postup je vhodný i z hlediska získání zkušeností - donutíte se navrhovat rozhraní (funkcí, metod) tak, aby byla rozšířitelná.
- Řešení této úlohy, které projde povinnými a nepovinnými testy na 100%, lze použít pro code review.