Prokládané obrázky

Termín odevzdání:	**25.03.2018 23:59:59**
Pozdní odevzdání s penalizací:	**01.07.2018 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**4.4000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	13 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit C/C++ funkci, která dokáže zpracovat obrázek uložený v souboru: načíst jej, změnit faktor prokládání a uložit jej.

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.

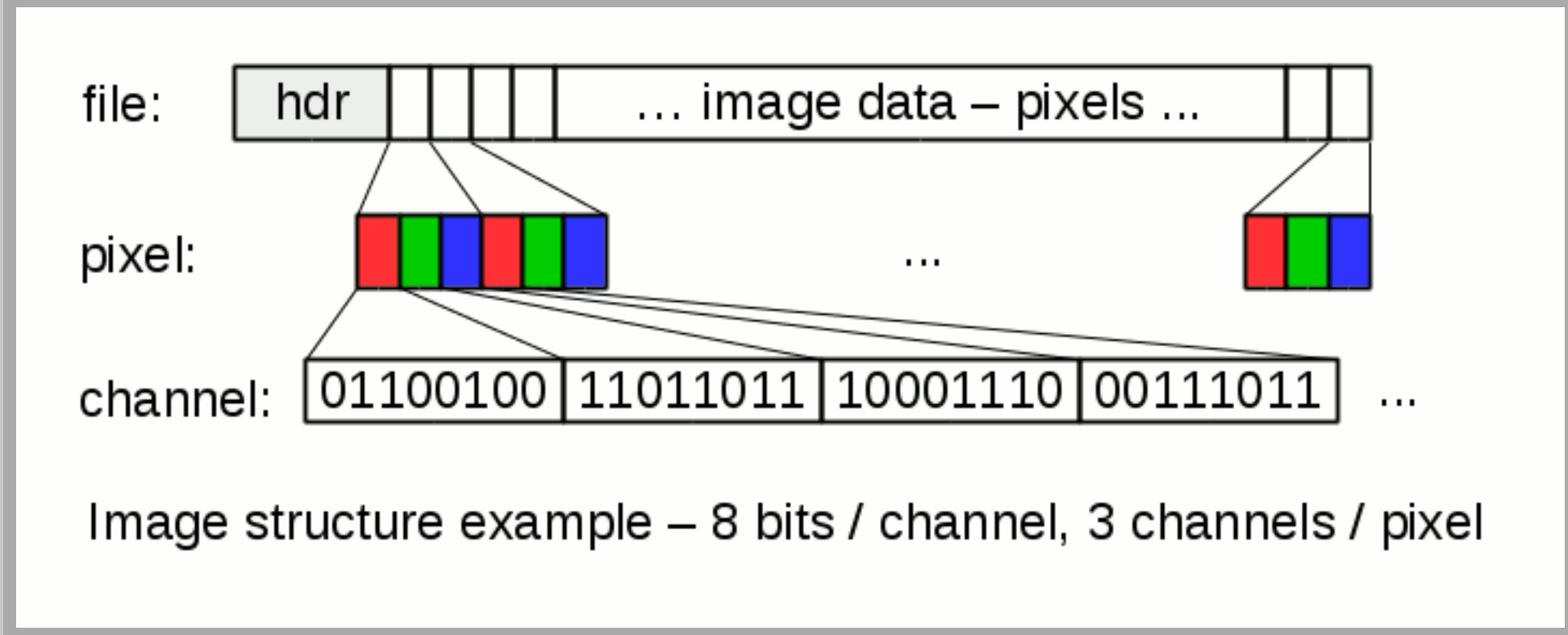

Image structure example – 8 bits / channel, 3 channels / pixel

Nevýhoda popsaného ukládání obrázku spočívá v tom, že pro zobrazení obrázku musíme čekat, dokud se nenačte prakticky celý datový obsah. Pokud je zajímavá část obrázku někde dole, nelze nic zobrazit, dokud nemáme přečtené všechny řádky nahoře. To může být na závadu, zejména pokud je obrázek přenášen po sítí a přenos dat není rychlý. Variantou je prokládané (interlaced) uložení obrázku. Pokud zvolíme faktor prokládání např. 8, uložíme nejprve obrazové pixely, které mají x i y souřadnici dělitelnou číslem 8. Takto projdeme celý obrázek, tedy uložíme pixely na souřadnicích `(x,y) = (0,0), (8,0), (16,0), (24,0), ..., (0,8), (8,8), (16,8), (24,8), ...`. Tedy po přenesení 1/64 objemu obrazových dat lze z hodnot těchto pixelů již zobrazit "hrubou" podobu obrázku. V následujícím kroku přeneseme pixely uložené na souřadnicích dělitelných 4 (ale již nepřenášíme pixely přenesené v minulém kroku). Tedy přeneseme pixely na souřadnicích `(x,y) = (4,0), (12,0), (20,0), (28,0), ..., (0,4), (4,4), (8,4), (12,4), (16,4), (20,4), (24,4), (28,4), ..., (4,8), (12,8), (20,8), (28,8), ...`. Tyto pixely přenesou dalších 3/64 objemu obrazových dat a společně s daty přenesenými v minulém kroku jsme schopni zobrazit obrázek s dvakrát vyšším rozlišením. Opakováním postupu pro souřadnice pixelů dělitelné 2 a 1 se postupně dostaneme až k finálnímu obrazu.

image – 11x9 pixels

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98

pixel order in the image file, interleaving factor = 8

0	8	88	96	4	44	48	52	92	2	6	10	22	24	26	28	30
32	46	50	54	66	68	70	72	74	76	90	94	98	1	3	5	7
9	11	12	13	14	15	16	17	18	19	20	21	23	25	27	29	31
33	34	35	36	37	38	39	40	41	42	43	45	47	49	51	53	55
56	57	58	59	60	61	62	63	64	65	67	69	71	73	75	77	78
79	80	81	82	83	84	85	86	87	89	91	93	95	97			

V uvedeném příkladu jsme zvolili počáteční faktor prokládání roven 8. Pokud bychom zvolili počáteční faktor prokládání 1, ukládal by se obrázek základním způsobem, tj. rovnou celý po řádcích. Naopak, vyšší volba faktoru prokládání by umožnila průběžné zobrazení obrázku s více kroky zpřesňování obsahu. To by mohlo být zajímavé pro obrázky s velkým rozlišením. Požadovaná funkce bude mít za úkol převádět obrázky mezi různými hodnotami faktoru prokládání:

```
bool recodeImage ( const char * srcFileName,
                   const char * dstFileName,
                   int          interlace,
                   uint16_t     byteOrder );
```

`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.

`interlace`
: je hodnota faktoru prokládání pro cílový obrázek. Tento parametr může nabývat hodnot 1, 2, 4, 8, 16, 32 nebo 64.

`byteOrder`
: bude mít hodnotu ENDIAN_LITTLE nebo ENDIAN_BIG (deklarace konstant je v přiloženém archivu). Podle tohoto příznaku funkce vytvoří cílový soubor s odpovídajícím kódováním vícebajtových hodnot. V povinných testech bude při volání vždy použita hodnota parametru ENDIAN_LITTLE (odpovídá HW, na kterém Váš program běží). Pro několik testů nesprávnými vstupy bude zadaná nesmyslná hodnota parametru (ani ENDIAN_BIG, ani ENDIAN_LITTLE), taková volání musí skončit chybou.

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, ...),
- zadaný faktor prokládání není mocnina 2 z intervalu 1 až 64,
- kódování vícebajtových hodnot není ani ENDIAN_LITTLE, ani ENDIAN_BIG.

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 bude odpovídat formátu pixelů ve zdrojovém obrázku,
- kódování little/big endian bude podle předaného parametru. V nepovinných testech může být zdrojový a cílový soubor v různých kódováních, navíc toto kódování se může lišit od kódování používaného HW. Pro zvládnutí nepovinného testu je potřeba dodat příslušné konverze,
- faktor prokládání cílového obrázku bude mít zadanou hodnotu. Funkce tedy musí přeuspořádat obrazová data, aby zdrojový a cílový obrázek "vypadal stejně".

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, počet bitů na kanál a faktor prokládání)
+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              8 7 6 5 4 3 2 1 0
     0 0    ...      0 I I I 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 /
```

Faktor prokládání je kódován bity 7 až 5:

```
000 -  1 (neprokládaný)
001 -  2
010 -  4
011 -  8
100 - 16
101 - 32
110 - 64
111 - nedovolená kombinace
```

Bity 8-15 jsou nevyužité, musí být nastavené na 0. Příklady kombinací:

```
0x006c = 0b01101100 - prokládání 8, 1 kanál na pixel, kanál má 8 bitů,
0x000e = 0b00001110 - prokládání 1, 3 kanály na pixel, každý kanál má 8 bitů,
0x00b3 = 0b10110011 - prokládání 32, 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). Pokud celkový počet bitů v obrázku není násobek 8, pak je potřeba poslední bajt 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.
- Prokládání se používá u standardizovaných grafických formátů. Příkladem je grafický formát PNG, který používá prokládání podle schematu Adam7, toto schéma se ale trochu liší od postupu popsaného v této úloze.

Vzorová data: Download

☐ **Referenční řešení**

13 **18.03.2018 03:43:10** Download

Stav odevzdání: Ohodnoceno
Hodnocení: 4.4000

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test se soubory dle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.002 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Mezni hodnoty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.870 s (limit: 1.998 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nespravne vstupy': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.001 s (limit: 1.128 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nahodny test': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.287 s (limit: 1.127 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test osetreni I/O chyb': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.324 s (limit: 0.840 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nahodny test, 16bit, little/big endian': Program provedl neplatnou operaci a byl ukončen (Segmentation fault/Bus error/Memory limit exceeded/Stack limit exceeded)
    - Celková doba běhu: 0.302 s (limit: 4.000 s)
    - Neúspěch v nepovinném testu, hodnocení: 80.00 %
  - Test 'Nahodny test, 1bit': Nebylo testováno
    - Neúspěch v bonusovém testu, hodnocení: Bonus nebude udělen
  - Celkové hodnocení: 80.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 0.80)
- Celkové procentní hodnocení: 80.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 0.80 * (5.00 + 0.50) = 4.40

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**10**	--	--	--