**Kódování a komprese dat**

# Projekt č. 3

## Konverze obrazového formátu GIF na BMP

**30. dubna 2018**

Jakub Pastuszek, xpastu00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

# 1 Úvod

Implementace knihovny a aplikace v jazyce C pro převod souboru grafického formátu GIF (Graphics Interchange Format) na soubor grafického formátu BMP (Microsoft Windows Bitmap). Soubory ve formátu GIF jsou zakódovány pomocí metody LZW. Výstupní soubor ve formátu BMP pak musí být možné zobrazit běžným prohlížečem obrázků (např. Irfan View).

# 2 Popis implementace

## 2.1 Implementace knihovny

Rozhraní knihovny definuje prototyp funkce pro konverzi GIF [1] (87a i 89a) na BMP. Dále je definován záznam udávající velikost souboru GIF a BMP v bytech. Knihovna umí převádět statický i animovaný 1 – 8 bitový formát GIF. Výstupní grafický soubor BMP neobsahuje komprimaci RLE. Z důvodu přenositelnosti jsou místo standardních datových typů použity předdefinované *int8_t*, *u_int32_t* apod., definované v knihovně *sys/types.h*.

## 2.2 Implementace aplikace

Aplikace se jmenuje *gif2bmp* a využívá implementované knihovny pro převod grafického formátu GIF na BMP s využitím kompresní metody LZW. Podporované parametry příkazového řádku jsou vypsány v kapitole 3. Zpracování parametrů příkazové řádky je prováděno pomocí funkce getopt definované v knihovně *unistd.h*. Výstupní zpráva (dále logovací soubor) obsahuje login a původní velikost souboru GIF a novou velikost souboru BMP, obě hodnoty jsou v bytech.

## 2.3 Prostředí

Knihovna a aplikace jsou přeložitelné na školním serveru Merlin, a to v aktuální verzi kompilátoru GCC na daném serveru. Překlad je zajištěn pomocí souboru *Makefile*. Provedení překladu je iniciováno zadáním příkazu make do příkazové řádky.

## 2.4 Popis algoritmu

Hlavní převodní funkce dostane na vstupu již připravené souborové deskriptory, jak vstupního, tak i výstupního souboru. Nejprve se čte vstupní GIF soubor a postupně se provádí syntaktická analýza. Soubor na začátku musí obsahovat hlavičku (dekadicky „*GIF87a*“ nebo „*GIF89a*“). Ihned za hlavičkou následují rozměry plátna a logický obrazový deskriptor, který mj. obsahuje metadata o globální tabulce barev, barevném rozlišení a barvu pozadí. V případě že metadata udávají, že soubor obsahuje globální tabulku barev, tak dále následuje právě zmíněná tabulka. V logickém obrazovém deskriptoru je zapsaná také velikost této tabulky. Každá položka tabulky obsahuje tři hodnoty (barvy) a to: červenou, zelenou a modrou – každou uloženou na samostatném bytu.

Za povinnou hlavičkou následují volitelná rozšíření [2]. Jedno z rozšíření je „*Graphics control*“ (v překl. řízení grafiky), které obsahuje data o zpoždění přehrávání u animovaných

GIF souborů a příznak průhlednosti s danou barvu, která má být v obrázku průhledná. Dalším z možných rozšíření jsou „*Plain text*“ a „*Comment*“, které obsahují obyčejnou textovou informaci, která ale není v obrázku vypsána/vykreslena, proto pro převod do BMP je redundantní. Obdobně to platí také pro rozšíření „*Application*“, které slouží k řízení animace GIF souboru.

Další povinnou části vstupního GIF souboru je „*Image descriptor*“ (v překl. obrazový deskriptor), který obsahuje informace o obrázku, který následuje bezprostředně za ním. Mezi tyto data patří šířka a výška obrázku, relativní umístění vzhledem k začátku plátna a v neposlední řadě informaci o lokální tabulce barev a prokládání. Informace o tom, jestliže za daným obrazovým deskriptorem následuje lokální tabulka je uložena v metadatech společně s její velikostí. Lokální tabulka má stejnou strukturu jako globální, ale v případě daného obrázku ji překrývá a používají se hodnoty z ní. Následující data jsou již kódové hodnoty, kde se ještě před nimi nachází hodnota minimální délky LZW kódu. Data jsou rozdělená do menších pod-bloků. Každý pod-blok před svým začátkem obsahuje byte udávající délku daného bloku. Z těchto informací se následně spočítá maximální počet možných kódů. Následně algoritmus pouze čte byty ze vstupu a ukládá je v opačném pořadí do pole (viz kapitola 2.5). Na konci souboru se musí nacházet hexadecimální hodnota „*0x3B*“.

Samotné dekódování [3] a ukládání do výstupního pole pixelů probíhá souběžně. Nejprve je nutné znát správnou velikost výstupního obrázku, aby bylo možné alokovat prostor pro výstupní pole. Po alokaci je celé pole naplněno barvou pozadí (pro případ výskytu transparentního pixelu). Poté následuje smyčka, která načítá kódy. V této smyčce se nejdříve ověřuje, zda není potřeba zvětšit délku kódového slova (synchronizace s kodérem). Dále je potřeba kontrolovat zaplnění tabulky barev, aby nepřetekla maximální možnou velikost 212 hodnot. V případě naplnění pokračujeme v dekódování bez přidávání nových záznamů do tabulky barev. Pokud se mezi načtenými kódy objeví „*Clear*“ kód, obnoví se původní obsah tabulka barev a původní délka kódového slova. Jestliže je tabulka barev zaplněné, ale ještě nebylo dosaženo maximální kapacity, je její velikost zdvojnásobena. Ukončení načítání a dekódování značí „*End Of Information*“ kód. Dále již následuje samotné dekódování a zápis do výstupního pole. První zakódovaná hodnota po kódu „*Clear*“ se dekóduje a jen zapíše na výstup. Každá další dekódování pracuje s aktuální dekódovanou hodnotou a hodnotou předcházející. Pokud je právě dekódovaná hodnota již v tabulce barev je ihned zapsána na výstup a dále je přidána jako nová hodnota do tabulky barev společně s prvním indexem minulé hodnoty. V případě, že se daná hodnota v tabulce barev ještě nenachází, je vyrobená pouze z předešlé hodnoty a následně zapsána na výstup. Jestliže obrázek obsahuje prokládání, je po dekódování provedeno patřičné přehození řádků výstupu [4], tak aby řádky následovaly po řadě za sebou.

Po fázi syntaktické analýzy vstupního GIF souboru následuje vytvoření výstupního BMP souboru. Tento výstupní soubor bude vždy mít 24 bitů na pixel bez komprese. Nejdříve se zapíše hlavička s určitými metadaty (výška a šířka obrázku, offset dat, ...) a následně data pixel po pixelu. Každý pixel je tvořen třemi barvami v jiném pořadí, než je tomu u souboru GIF, a to modrá, zelená a červená. Délka každého řádku dat BMP souboru musí být číslo dělitelné čtyřmi, proto pokud to nevychází, musí být řádek doplněn „vycpávkou“ a to v tomto případě v podobě samých nul.

### 2.5 Pomocné funkce

Knihovna obsahuje definici *SHOW_** hodnot pro možnost zobrazení kontrolních výpisů pomocí funkce *printDebug()*, která funguje stejně jako standardní *printf()*, ale je podmíněná právě definici dané SHOW_* hodnoty. Pro snazší výpisy bitových hodnot je definován převod z bytu na bity funkcí *BYTE_TO_BINARY()*. Další z pomocných funkcí je *toLittleEndian()*, která převede číslo z Big-Endian na Little-Endian, který je použit jak u GIF, tak BMP souboru.

Mezi funkce důležité pro správnou funkčnost celého algoritmu a zároveň spjaté s LZW dekódováním patří *reverse_byte_binary()* a *reverse_word_binary()*, které mění pořadí bitů tak, aby LSB („least significant bit“ – nejméně významový bit) byl vlevo a MSB („most significant bit“ – nejvíce významová bit) vpravo. Toto přehození se hodí v případě uložení všech vstupních kódů do pole za sebou hlavně při dekódování délky kódového slova jiném než osm bitů. Funkce *get_n_bits()* vrací n bitů od daného offsetu pole bytů. Funkce *change_row_interlaced()* je užitečná pouze v případě prokládaného GIF souboru. Zaručí správné pořadí zpracování řádků daného GIF souboru, tak aby výsledný obrázek byl správně vykreslen.

## 3 Spuštění

Parametry programu jsou čtyři. Prvním parametrem je *-i*, za kterým následuje název vstupního GIF souboru (výchozí hodnota je standardní vstup). Dalším parametrem je *-o*, následovaný názvem výstupního BMP souboru (výchozí hodnotou je standardní výstup). Třetím parametrem je *-l*, za kterým následuje název logovacího souboru. Posledním z parametrů je *-h*, který vypíše nápovědu programu. Žádný z parametrů není povinný, jsou pouze volitelné. V případě nezadání parametru *-l* se žádné logovací informace nevypíšou.

gif2bmp [ -h ] [ -i <input_file> ] [ -o <output_file> ] [ -l <log_file> ]

## 4 Závěr

Implementace knihovny pro převod souboru z grafického formátu GIF na soubor grafického formátu BMP proběhla úspěšně. Převádět lze soubory GIF s rozličnými velikostmi bitového formátu, obsahující komentáře, text či dokonce animované sekvence obrázků, dokonce také obrázky prokládané. Výstupní soubor porovnán se vstupním vizuální kontrolou v běžném prohlížeči obrázků.

# Zdroje

[1] Cover Sheet for the GIF89a Specification [online]. [cit. 2018-04-30]. Dostupné z: https://www.w3.org/Graphics/GIF/spec-gif89a.txt

[2] Project: What's In A GIF - Bit by Byte [online]. [cit. 2018-04-30]. Dostupné z: http://www.matthewflickinger.com/lab/whatsinagif/bits_and_bytes.asp

[3] Project: What's In A GIF - LZW Image Data [online]. [cit. 2018-04-30]. Dostupné z: http://www.matthewflickinger.com/lab/whatsinagif/lzw_image_data.asp

[4] GIF File Format Summary [online]. [cit. 2018-04-30]. Dostupné z: https://www.fileformat.info/format/gif/egff.htm