VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Konverze obrazového formátu GIF na BMP (Projekt č. 3)

Projekt do predmetu Kódování a komprese dat (KKO)

Peter Tisovčík (xtisov00)

1. apríla 2018

1 Úvod

Zadaním projektu bolo vytvoriť knižnicu a ukážkový program, ktorý prevedie obrázok vo formáte GIF do obrázku vo formáte BMP s použitím vytvorenej knižnice a zadanými parametrami programu.

2 Použitie

Ukážkovú aplikáciu je možné skompilovať pomocou príkazu `make` a spustiť `./gif2bmp` v terminály s nasledujúcimi voľbami:

Prepínač	Parameter	Popis prepínača
-h		Zobrazenie nápovedy.
-i	**<file>**	Vstupný súbor vo formáte GIF, ak nie je zadaný použije sa `stdin`.
-o	**<file>**	Výstupný súbor vo formáte BMP, ak nie je zadaný použije sa `stdout`.
-l	**<file>**	Výstupný súbor zo štatistikou, ktorý obsahuje informácie o logine autora a veľkostiach GIF a BMP súborov.

Tabuľka 1: Tabuľka prepínačov ukážkovej aplikácie

3 Implementácia

Pre vývoj knižnice bol zvolený jazyk C++. Rozhraním knižnice je funkcia `int gif2bmp(tGIF2BMP *gif2bmp, FILE *inputFile, FILE *outputFile)`. Funkcia vracia 0 pri úspešnom prevedení obrázku do BMP formátu, pri chybe -1. `tGIF2BMP` je štruktúru s informáciami o veľkosti vstupného GIF súboru, výstupného BMP súboru a loginom riešiteľa projektu. Ďalšími parametrami sú ukazovatele na vstupný GIF a výstupný BMP súbor.

Po zavolaní tejto funkcie sa overí, či dané ukazovatele sú nastavené na súbor. Následne sa načíta celý obsah súboru, ktorý sa vloží ako parameter pri vytváraní objektu typu `GIFImage`. Trieda `GIFImage` reprezentuje všetky potrebné operácie a dáta, ktoré sú potrebné pre prevod obrázku do BMP formátu.

GIF súbor obsahuje niekoľko povinných a voliteľných blokov, ktoré sa spracuj v tomto poradí:

- Overenie, či vstupný formát je GIF súbor podľa signatúry a podľa verzie hlavičky. Knižnica podporuje GIF verzie `87a` a `89a`.
- Načítanie informácii z *logical screen descriptor-u*, kde sa nachádzajú základné informácie o obrázku ako napríklad veľkosť plátna.
- Načítanie zoznamu farieb z globálnej tabuľky farieb.
- Spracovanie rozšírení:
  - **Graphics control extension** - podporované rozšírenie a spracovanie informácii v tomto bloku
  - **Plain text extension** - nepodporované rozšírenie
  - **Comment extension** - blok s daným rozšírením sa len preskočí
  - **Application extension** - blok s daným rozšírením sa len preskočí
- Spracovanie obrazových dát komprimovaných LZW algoritmom.

Po skončení spracovania GIF súboru nastáva vytvorenie BMP formátu. BMP formát obsahuje 2 hlavičky, ktoré je potrebné nastaviť správnymi hodnotami ako napríklad veľkosť súboru, veľkosť plátna a hodnoty masiek[1]. Následne sa zapíšu jednotlivé hlavičky a za nimi nasledujú dáta obrázku. Ak je daná farba priehľadná je nutné ešte nastaviť *alpha* bajt na hodnotu `0xff`.

Knižnica si poradí s prekladanými obrázkami a umožňuje z animovaného GIF obrázku vybrať prvý snímok a ten uložiť do BMP formátu.

[1] https://www.root.cz/clanky/graficky-format-bmp-pouzivany-a-pritom-neoblibeny/

3.1 LZW algoritmus

LZW je bezstratový komprimačný algoritmus, ktorý sa používa vo formáte GIF.

Na začiatku algoritmu sa nastaví počiatočná veľkosť slovníka. Táto veľkosť sa vypočíta ako $2^{(N+1)}$, kde N je uložená veľkosť v bloku *Logical screen descriptor* v položke *Size of global color table*[2]. Slovník obsahuje významné hodnoty *End of LZW*, ktorý je uložený na indexu $2^{(N+1)}$ a značí koniec komprimovaných dát. Ďalšia významná hodnota *Clear code* je uložená na indexe $2^{(N+1)} + 1$ a značí, že sa má slovník vyčistiť od starých hodnôt.

Algoritmus pre svoju činnosť potrebuje vedieť veľkosť kódového slova. Ak táto veľkosť dosiahne hodnotu 13 zmenší sa na 12 a ak je to potrebné táto operácia sa opakuje. Pseudokód LZW algoritmu[3]:

Algorithm 1 LZW algoritmus

```
1:  procedure LZW
2:      dictionary.init ← inicializácia slovníka
3:
4:      cW = next() ← načítanie prvého kódového slova zo vstupu
5:      color.cW = cW ← farba s indexom cW v lokálnej/globálnej tabuľke farieb
6:
7:      loop:
8:          pW = cW ← uloženie predchádzajúceho kódového slova
9:          cW = next() ← načítanie ďalšieho kódového slova zo vstupu
10:
11:         if color.cW is in dictionary then
12:             zápis color.cW do výstupného zoznamu farieb
13:             P = color.pW
14:             C = color.cW.first ← výber prvej farby z color.cW
15:             dictionary.insert({P, C}) ← vytvorenie nového záznamu v slovníku s danými hodnotami
16:         else
17:             P = color.pW
18:             C = color.pW.first ← výber prvej farby z color.pW
19:             dictionary.insert({P, C}) ← vytvorenie nového záznamu v slovníku s danými hodnotami
20:             zápis {P, C} do výstupného zoznamu farieb
21:
22:         if dictionary.isEmpty() then
23:             break
```

4 Testovanie

Počas vývoja knižnice a testovacej aplikácie boli napísané vlastné automatizované testy, ktoré overovali správnosť vstupných parametrov (správnych ale aj nesprávnych). Ďalším typom testov boli testy pre overenie prevodu obrázku vo formáte GIF do obrázkov vo formáte BMP. Použité boli obrázky zo zadania ale aj vlastné, ktoré rozšírili sadu na niekoľko desiatok rôznych veľkostí a vlastností (prekladané, transparentné, s vloženým textom, animované...).

5 Záver

Cieľom projektu bolo vytvoriť funkčnú knižnicu pre prevod obrázkov vo formáte GIF do obrázkov vo formáte BMP a ukázať ju na príklade aplikácie, ktorá bude používať túto knižnicu. Výsledná knižnica a ukážková aplikácia je funkčná.

Knižnica podporuje prekladané, priehľadné, animované obrázky a obrázky s komentármi. Overovanie funkčnosti prebiehalo automatickými testami.

Aplikáciu by bolo možné ďalej rozšíriť o detailnejšie vypisovanie chýb, exportovanie každého snímku animovaného obrázku ale aj o optimalizáciou na zvýšenie efektivity používaných zdrojov.

[2] http://www.matthewflickinger.com/lab/whatsinagif/bits_and_bytes.asp

[3] http://www.matthewflickinger.com/lab/whatsinagif/lzw_image_data.asp