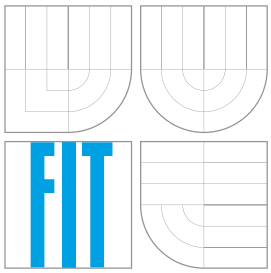
Vysoké učení technické v Brně
Brno University of Technology



Fakulta informačních technologií

Faculty of Information Technology

Konverze obrazového formátu GIF na BMP

KKO - Kódování a komprese dat

Autor práce Ladislav Šulák

Login xsulak04

Brno 3/2016

Popis knižnice

Knižnica implementovaná v jazyku C++ sa stará o konverziu grafického formátu GIF na súbor grafického formátu BMP. Vstupný súbor GIF má dáta zakódované pomocou metódy LZW, výstupný súbor nemá svoje grafické dáta zakódované. Prototyp funkcie, ktorá slúži pre konverziu a volá ostatné funkcie v tejto knižnici má tvar:

```
int gif2bmp(tGIF2BMP *gif2bmp, FILE *inputFile, FILE *outputFile);
```

S parametrami:

- *gif2bmp* obsahuje veľkosti vstupného a výstupného súboru v bytoch,
- *inputFile* ukazateľ na vstupný súbor, prípadne štandardný vstup a
- *outputFile* ukazateľ na výstupný súbor alebo štandardný výstup.

Popis aplikácie

Aplikácia využíva knižnicu popísanú v predchádzajúcej kapitole má nasledovné argumenty príkazového riadku, spracované pomocou knižnice getopt:

- *-i <inputFile>* pre zadanie vstupného súboru vo formáte GIF. V prípade absencie tohto parametra bude použitý štandardný vstup,
- *-o <outputFile>* pre zadanie výstupného súboru vo formáte BMP. V prípade, že tento parameter nebude zadaný, bude použitý štandardný výstup,
- *-l <logFile>* pre vytvorenie výstupnej správy zobrazujúcej login a veľkosti vstupného a výstupného súboru v bytoch,
- *-h* pre vypísanie nápovedy na štandardný výstup.

Grafický formát GIF

V tejto sekcii bude v stručnosti popísaná štruktúra grafického formátu GIF (s verziou 89a, avšak aplikácia podporuje aj verziu 87a). Informácie budú čerpané priamo zo špecifikácie[1]. Štruktúra grafického formátu GIF je nasledovná:

- Postupnosť *GIF89a* ako súčasť hlavičky, ktorou je identifikovaný formát súboru,
- *Logical Screen* obsahujúci informácie o logickej obrazovke obrázku, je to povinná položka zastúpená práve raz,
- *Global/Local Color Table* je zoznam RGB hodnôt, ktoré reprezentujú intenzity farieb pre každý index tejto tabuľky,
- *Image Descriptor* obsahuje informácie nutné k správnemu spracovaniu obrazových dát, na každý dátový stream môže byť maximalne 1,
- *Extensions*, teda rozšírenia, ktoré sú nepovinné, prípadne sa môžu aj opakovať: *Graphic Control*, *Comment*, *PlainText* a *Application Extension*,
- *Trailer* - hodnota *0x3B*.

[1] https://www.w3.org/Graphics/GIF/spec-gif89a.txt

Grafický formát BMP

Štruktúra grafického formátu BMP, ktorý je vytváraný v tomto projekte je nasledovná:

- Postupnosť *BM*, identifikujúca formát súboru,
- *Total size* je položka označujúca celkovú veľkosť súboru,
- *Reserved1, Reserved2* sú nevyužívané hodnoty, nastavené na 0,
- *Offset bits* je posunutie vzhľadom ku ktorému začínajú obrazové dáta,
- *Header size* je veľkosť hlavičky, minimálne 40 bajtov,
- *Header*. Typ hlavičky vytváraný v tomto projekte je typu Microsoft Windows BMP, ktorý má nasledovné položky:
  - *Width*, *Height* reprezentujú šírku a výšku celého obrázku,
  - *Planes* je hodnota nastavená na 1,
  - *Bits per Pixel* je hodnota nastavená na 24, teda každá farebná zložka má 8 bytov,
  - *Compression* je v tomto projekte hodnota vždy 0, pretože výsledný BMP súbor nie je komprimovaný,
  - *Image size* označuje veľkosť obrazových dát,
  - *X, Y pixels per meter* sú hodnoty nastavené na 0,
  - *Number of Colors*, *Colors Important* sú hodnoty tiež nastavené na 0 pretože sa budú využívať všetky farby a sú rovnako dôležité,
- *Image data* uschovávajú samotné obrazové dáta pripravené pre výstup.

Algoritmus

Implementovaná knižnica spracúva vstupný súbor a naplní potrebné dátové štruktúry informáciami z hlavičky a ostatných sekcií. Následne dekóduje obrazové dáta pomocou algoritmu LZW, znázorneného pseudokódom 1. Ak je to potrebné a sú prekladané, tak ich preusporiada a výsledok opäť zoradí, prípadne zarovná tak, ako je to dané formátom BMP.

```
1  inicializácia slovníku
2  // CODE predstavuje index a {CODE} predstavuje hodnotu v slovníku na tomto indexe
3  načítanie prvého kódového slova CODE zo vstupu
4  výstup += {CODE}
5  while (vstup nie je prázdny)
6      načítanie ďalšieho kódového slova CODE zo vstupu
7
8      // Veľkost slovníku je väčšia ako CODE
9      if (CODE je v slovníku)
10         K je prvý index z {CODE}
11         výstup += {CODE}
12         pridaj {CODE-1} + K do slovníku
13     else
14         K je prvý index z {CODE-1}
15         výstup += {CODE-1} + K
16         pridaj {CODE-1} + K do slovníku
```

Pseudokód 1: Algoritmus LZW, pomocou ktorého sa dekomprimujú obrazové dáta.