Úkol 1 - Steganografie

Vytvořte v programovacím jazyce C/C++, Java nebo Python program, který bude provádět steganografii ve formě ukrytí libovolného souboru (např. obrázku, textového souboru či zvukové MP3 nahrávky) do BMP obrázku. Výsledný soubor bude korektní soubor BMP, který **nebude nikterak poškozen** a bude jej možné otevřít libovolným prohlížečem obrázků.

Součástí zadání je kostra programu v Pythonu (**`main.py`**), kterou můžete buď doplnit nebo z ní vyjít pro váš vlastní program, pokud chcete. Archiv se zadáním obsahuje následující:

- **`main.py`**
- **`weber.bmp`**
- **`test/`**
- **`out/`**
- **`validation/`**
- **`decoded/`**

V zadání je k dispozici obrázek **`weber.bmp`**, který slouží jako médium pro ukrytí. Do něj postupně pomocí metody LSB (Least Significant Bit) budete ukrývat soubory ze složky **`validation/`**.

Ve složce **`validation/`** jsou validační soubory, na kterých je možné/nutné vaši práci odladit. Výsledné korektně zakódované obrázky uloží program do složky **`out/`**; každý soubor v této složce bude korektní BMP, který půjde otevřít libovolným prohlížečem obrázků. Program ve druhé fázi pak projde složku **`out/`**, dekóduje obrázky a výsledné extrahované soubory uloží do složky **`decoded/`**. Výsledkem správného běhu programu bude to, že soubory ve složce **`validation/`** a ve složce **`decoded/`** budou stejné a nepoškozené. Pokud program vyhodnotí, že se metodou LSB celý soubor nevejde do obrázku **`weber.bmp`**, **steganografii neprovádějte a soubor přeskočte**. Nedělejte tedy kódování s využitím 2 nebo 4 posledních bitů, došlo by pak k viditelným změnám v obrázku.

Vyhodnocení se bude provádět automaticky. Náhodná sada testovacích souborů se vloží do složky **`validation/`** a vyhodnocení se provede na nich. Cílem je ověřit, že program není “šitý na míru” validačním souborům, které máte k dispozici. Automatický validátor poté spustí program a porovná obsah složek **`out/`** a **`decoded/`**.

Z podstaty algoritmu steganografie musí mít zakódované obrázky ve složce **`out/`** stejnou velikost jako **`weber.bmp`**; toto je jeden z testů, který se rovněž bude provádět.

Pro zpětné dekódování budete potřebovat znát jméno souboru, příponu a také počet bajtů, které jste ukrývali. Tato metadata zakódujte současně s daty (např. do prvních 4 bytů délku souboru a do dalších 4 koncovku nebo jakkoliv jinak uznáte za vhodné).

Příklad výstupu

Uvažujme jediný soubor např. **`validation/validation1.mp3`**

- zakóduje se soubor **`validation1.mp3`** do obrázku **`weber.bmp`**
  → vytvoří se **`out/validation1___weber.bmp`**
- při zpětném dekódování nejprve přečtěte metadata (počet bajtů, které bude načítat a příponu souboru). Poté vytvořte soubor **`decoded/validation1.mp3`**

Porovnáte-li tyto dva soubory, musí být naprosto stejné.
např. pomocí “**`diff validation/validation1.mp3 decoded/validation1.mp3`**“
Toto je taky jeden z testů, který se provádí.

Jednotkové testy

Součástí archivu se zadáním je i složka **`test/`** a v ní skript s jednotkovými testy v Pythonu. Práci odlaďte tak, aby ideálně všechny jednotkové testy prošly. Pokud implementujete v jiném jazyku než v Pythonu, doporučujeme si sadu podobných jednotkových testů také napsat. Testy kontrolují následující výstupy:

1. `test_steganography_img_sizes`
   - porovná se, zda všechny obrázky po provedení steganografie ve složce **`out/`** mají stejnou velikost jako **`weber.bmp`**
2. `test_all_imgs_after_steganography`
   - porovná se, zda všechny obrázky ve složce **`out/`** jsou odlišné od **`weber.bmp`** a zároveň jdou korektně načíst pomocí **opencv** knihovny
3. `test_decoded_results`
   - porovná se, zda po provedení steganografie všechny soubory ve složce **`decoded/`** a odpovídající ve složce **`validation/`** jsou stejné

Pro další informace si prosím přečtěte soubor **`README.md`**, který je součástí archivu. Váš program společně se všemi adresáři z archivu se zadáním zabalte do ZIP souboru a pojmenujte ho podle svého osobního čísla.