Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Steganografia-obraz

Autor:	Tomasz Szewczyk
Prowadzący:	mgr inż. Marek Kokot
Rok akademicki:	2019/2020
Kierunek:	Informatyka
Rodzaj studiów:	SSI
Semestr:	2
Termin laboratorium:	środa, 13:15-14:45
Sekcja:	5
Termin oddania sprawozdania:	2020-09-09

1. Treść zadania

Proszę napisać program umożliwiający ukrycie w pliku BMP innego pliku dowolnego typu. Program powinien przyjmować liczbę bitów każdej ze składowych RGB, które zostaną przeznaczone na ukrycie informacji. Program powinien działać dla różnych głębi kolorów.

2 Analiza zadania

Zadanie przedstawia problem ukrycia w pliku typu .bmp innego rodzaju pliku o dowolnym jego formacie. Analizując treść zadania od początku od razu można pomyśleć o argumentach, które będziemy przekazywać naszemu programowi czyli plik wejściowy, ukrywany oraz wyjściowy, który zostanie utworzony a także ilość bitów jakie poświęcimy każdej składowej R,G,B na ukrycie informacji. Program powinniśmy wykonywać z wiersza poleceń. Z racji tego, że mamy do czynienia z formatem .bmp należy zapoznać się z tym jak owy format jest zbudowany. Z racji tego, że program powinien działać dla różnych głębi kolorów, należy zapoznać się z mechaniką działania poszczególnych głębi tj. 16-bit, 24-bit.

2.1 Struktury danych

Program używa zmiennych typu char do obsługi pojedynczych bajtów oraz zmiennych int operacji binarnych.

2.2 Algorytmy

Program wykorzystuje algorytm LSB, dzięki któremu zostają wykorzystane najmniej znaczące bity w bajtach pliku wejściowego i po kolei te bajty są zamieniane z bajtami pliku, który chcemy ukryć przy użyciu operatorów logicznych. W wyniku tej zamiany zmiany w pliku wyjściowym są niewidoczne gołym okiem dla osoby, która otwiera plik z ukrytym plikiem. Osoba otwierająca plik widzi plik wejściowy.

3. Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego, ukrywanego i pliku, który utworzymy czyli plik

wyjściowy. Następnie przy użyciu przełączników -R, -G, -B przekazujemy ilość bitów, które mają zostać przeznaczone na ukrycie informacji.

<source file> <secret file> <destination file> -R [] -G [] -B []

Przykład: marbles.bmp secret.txt marbles1.bmp -R [1] -G [2] -B [3]

Dzięki takiej instrukcji program wie, ile bitów ma zostać przeznaczonych na ukrycie informacji z każdej ze składowych RGB.

Plik wejściowy musi być w formacie .bmp. Plik do zaszyfrowania może mieć dowolny format. Plik wyjściowy musi mieć format pliku wejściowego.

Przy wpisaniu zbyt dużej ilości parametrów konsola wypisuje nam komunikat z instrukcją jak wpisać argumenty dla programu.

4.Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs(komunikację z użytkownikiem) od logiki aplikacji (znajdowanie tras dla baz spedycyjnych).

4.1 Ogólna struktura programu

W funkcji głównej, wywołujemy **funkcje kontrola_argumentów**, której zadaniem jest sprawdzenie czy poprawnie podano argumenty dla programu przez użytkownika. Jeżeli wszystko jest w porządku program używa funkcji **wczytaj_pliki**, której zadaniem jest otwarcie pliku wejściowego, pliku do zaszyfrowania oraz utworzenie pliku wyjściowego a także zwracanie błędów jeżeli podczas otwierania lub tworzenia pliku wystąpił błąd. Funkcja ta ładuje także wartości zmiennych RGB. Następnie przy użyciu funkcji **czytaj_pliki** uzyskujemy informacje takie jak długość pliku, długość nagłówka, głębia kolorów przy użyciu funkcji **znajdz_dlugosc_nagłowka**, **znajdz_długość_pliku**, znajdz_glebie_kolorow. Następuje także sprawdzenie czy ukrywany plik nie przekracza wielkości pliku wejściowego- funkcja **sprawdzanie_pamięci**. Funkcja **pisz_nagłowek** przepisuje nagłówek pliku wejściowego aby informacje w nim podane zgadzały się w pliku wyjściowym. Teraz gdy wszystkie dane się zgadzają, można przejść do wykonania

szyfrowania w zależności od tego z jaką głębią kolorów została podana przez użytkownika w pliku wejściowym. Przy użyciu funkcji **zakoduj_24** oraz **zakoduj_16** szyfrujemy plik. Jezeli podana głębia kolorów jest błędna funkcja **błędna_głębia_kolorów** zwraca błąd.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku doxyfile na platformie.

5. Testowanie

Program został testowany pod kątem wprowadzania błędnej liczby parametrów, został przed tym zabezpieczony poprzez wyrzucanie komunikatu z instrukcją. Testowany był pod kątem tworzenia pliku wyjściowego a także pod kątem podania błędnych danych. Program został sprawdzony pod kątem wycieków pamięci przy użyciu narzędzia pobranego z platformy. Program został przetestowany na dwóch komputerach. Na obydwóch działał prawidłowo.

6. Wnioski

Program Steganografia obrazu dla początkującego programisty jest dosyć trudnym projektem. Najwięcej trudności sprawia obycie się z programowaniem niskopoziomowym oraz operacje bitowe i dojście do mechaniki działania głębi kolorów. Dzięki temu projektowi nauczyłem się korzystać poprawnie z wielu funkcji oraz przywiązywać dużą uwagę do najdrobniejszych szczegółów, ponieważ każdy mógł spowodować utratę jakości obrazu wejściowego. Dzięki researchowi udało mi się poznać operacje na obrazach oraz zależności między głębią kolorów. Nauczyłem się tworzyć usługę dla ludzi, którzy potrzebują dobrze ukryć swoje dane a następnie je przekazać.

Literatura:

-Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Wprowadzenie do algorytmów. Wydawnictwa Naukowo-Techniczne, Warszawa, 2001.

-https://pl.wikipedia.org/wiki/Najmniej_znaczący_bit
-https://pawelskaruz.pl/2017/03/stegocore-algorytm-ukrywania-danychs-lsb/
-https://pl.wikipedia.org/wiki/Windows_Bitmap