Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 2

Steganografia w pliku BMP

autor	Wojciech Janota
prowadzący	dr inż. Mirosław Skrzewski
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	2
termin laboratorium	środa, 15:30 – 17:00
sekcja	51
termin oddania sprawozdania	2020-07-27

1 Treść zadania

Proszę napisać program umożliwiający ukrycie / odczyt w/z 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. Program uruchamiany z linii komend z przełącznikami: -i plik do ukrycia, -p plik obrazu, -o plik wyjściowy, -r odczyt ukrytej informacji, -b liczba bitów składowej RGB.

Uwagi: Jako strukturę danych proszę zastosować tablicę dynamiczną.

2 Analiza zadania

Zagadnienie przedstawia problem zapisania dowolnego pliku na ostatnich n bitach składowych tablicy pixeli obrazu w formacie BMP.

2.1 Struktury danych

Program wykorzystuje tablice dynamiczne do przechowywania poszczególnych składowych pixeli oraz bajtów plików do zapisania. Wykorzystuje także struktury przechowujące nagłówki plików BMP oraz specjalny dodatkowy nagłówek steganograficzny, który przechowuje informacje o tym, czy dany plik zawiera ukryte informacje oraz rozmiar ukrytego pliku. Struktury szczegółowo omówiłem w dokumentacji, która jest w załączniku.

2.2 Algorytmy

Program ukrywa pliki dowolnego typu w podanym pliku BMP. Nie wykorzystałem żadnych skomplikowanych algorytmów, jedynie proste operacje na maskach bitowych, które umożliwiły mi zapis oraz odczyt danych do oraz z pliku BMP.

Maski bitowe których używałem dla plików 24- i 32-bitowych pozwalają mi zapisywać i odczytywać dane na ostatnich n bitach pełnego bajta danych. Obliczam je w następujący sposób: `mask = (1 << n) - 1`. Wtedy mogę za pomocą operatora bitowego `AND` oraz `NOT` "wyczyścićostatnie n bitów, czyli je wyzerować, a następnie za pomocą operatora `OR` zapisać tam podane dane. Użycie operatora `OR` jest poprawne, ponieważ te ostatnie miejsca są wyzerowane.

2.3 Analiza pliku BMP

Struktura pliku BMP jest następująca[2]:

Nazwa struktury	Rozmiar	Opcjonalny	Znaczenie
Nagłówek główny	14 bajtów	Nie	Przechowuje najważniejsze informacje o pliku.
Nagłówek DIB	7 różnych wersji	Nie	Przechowuje najważniejsze informacje o obrazie i definiuje format piksela.
Dodatkowe maski	12 lub 16 bajtów	Tak	Definiuje format piksela.
Tablica kolorów	zmienny	Tak/Nie	Definiuje kolory użyte przez obraz (tablicę pikseli)
Przerwa	zmienny	Tak	Struktura dostosowująca
Tablica pixeli	zmienny	Nie	Definiuje wartości poszczególnych pikseli
Przerwa	zmienny	Tak	Struktura dostosowująca
Profil kolorów ICC	zmienny	Tak	Definiuje profil kolorów dla systemu zarządzania kolorem

Pliki BMP zawierają na początku nagłówek główny, nazywany przez mnie dalej nagłówkiem `BMPHEADER`. Jego struktura jest następująca[1]:

Offset	Rozmiar	Przeznaczenie
0x0	2 bajty	nagłówek determinujący, czy plik jest plikiem BMP, dla pliku BMP wynosi BM
0x2	4 bajty	Rozmiar całego pliku w bajtach
0x6	2 bajty	zarezerwowany
0x8	2 bajty	zarezerwowany
0xA	4 bajty	Offset, początkowy adres tablicy pixeli

Następny jest nagłówek `DIBINFOHEADER`, nazywany przeze mnie `DIBHEADER`. Jego struktura jest następująca[1]:

Offset	Rozmiar	Nazwa	Opis zawartości
0x14	4 bajty	biSize	Wielkość nagłówka informacyjnego
0x18	4 bajty	biWidth	Szerokość obrazu w pikselach
0x22	4 bajty	biHeight	Wysokość obrazu w pikselach
0x26	2 bajty	biPlanes	Liczba warstw kolorów, zwykle 1
0x28	2 bajty	biBitCount	Liczba bitów na piksel
0x30	4 bajty	biCompression	Algorytm kompresji
0x34	4 bajty	biSizeImage	Rozmiar samego obrazu
0x38	4 bajty	biXPelsPerMeter	Rozdzielczość pozioma
0x42	4 bajty	biYPelsPerMeter	Rozdzielczość pionowa
0x46	4 bajty	biClrUsed	Liczba kolorów w palecie
0x50	4 bajty	biClrImportant	Liczba ważnych kolorów w palecie (gdy 0 to wszystkie są ważne)

Po tych nagłówkach na offsecie podanym w nagłówku BMPHEADER znajduje się tablica pixeli. Dla obrazów 16 bitowych składowe jednego pixela są zapisane na 2 bajtach, dla 24 bitowych na 3 bajtach a dla 32 bitowych na 4 bajtach. W każdym wierszu tablicy pixeli zapisanym do pliku musi być wielokrotność 4 bajtów, więc brakujące bajty sa nadpisywane zerami[2].

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Po podaniu flagi -h wyświetla się pomoc, która podpowiada prawidłową semantykę wywołania programu. Do programu należy przekazać następujące flagi:

- -w — zapisywanie danych do obrazu

- -r — odczytywanie danych z obrazu
- -p — obraz wejściowy
- -i — plik z danymi do ukrycia
- -o — plik/obraz wyjściowy
- -b — liczba bitów na których ma być zapisana informacja

Poprawne wywołanie programu powinno zawierać flagi: -h, -w -i -o -p -b, lub -r -o -b.

Poza pierwszym argumentem, kolejność flag nie ma znaczenia.

4 Specyfikacja wewnętrzna

4.1 Ogólna struktura programu

W funkcji głównej najpierw odczytywane są przekazane pierwsze flagi. Następnie, w przypadku podania flagi -h wyświetlana jest pomoc, natomiast w przypadku, gdy pierwszymi argumentami są -r lub -w rozpoczyna się wykonanie programu.

Dla flagi -w najpierw odczytywane są pozostałe flagi oraz dane, następnie po sprawdzeniu ich poprawności oraz otwarciu odpowiednich plików do pamięci za pomocą funkcji `read_headers()` zostają wczytane nagłówki pliku BMP: `BMPHEADER` oraz `DIBHEADER`. Po wypisaniu podstawowych informacji o pliku z nagłówków w zależności od głębi kolorów danego pliku wywołana zostaje funkcja `pixelArray_read_nbit()`, gdzie n to 16, 24 lub 32. Funkcja ta wczytuje do tablicy dynamicznej `pixelArray` tablicę pixeli w pliku BMP. Następnie funkcja `read_file_to_memory()` wczytuje do tablicy `dataArray` plik do ukrycia w pliku obrazu. Następnie po sprawdzeniu poprawności wczytania danych funkcja `write_data_to_image()` zapisuje dane do pliku. Po tej operacji zostają zamknięte pliki oraz zwolniona pamięć.

Dla flagi -r najpierw odcztytwane są pozostałe flagi i dane, następnie po sprawdzeniu ich poprawności oraz otwarciu odpowiednich plików do pamięci za pomocą funkcji `read_headers()` zostają wczytane nagłówki pliku BMP: `BMPHEADER` oraz `DIBHEADER`, analogicznie do wywołania z flagą -w. Po sprawdzeniu poprawności podanych danych oraz odczytu nagłówków także zostaje wywołana funkcja `pixelArray_read_nbit()`, tak samo jak to miało miejsce dla flagi -w. Następnie wywoływana jest funkcja `write_data_from_image()`, która odczytuje dane z obrazu i zapisuje je do pliku wynikowego. Po zwolnieniu pamięci i zamknięciu plików następuje zakończenie pracy programu.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany dla wszystkich obsługiwanych rodzajów plików BMP tj. 16, 24 i 32-bitowych. Ukrywanie pliku pustego spowoduje późniejsze odczytanie pliku pustego, ponieważ program zapisuje także wielkość ukrytego pliku. W wyniku limitacji powstałych przez zapisywanie wielkości pliku w wolnym miejscu w nagłówku BMP wielkość ukrywanych danych nie może przekroczyć 65kB, ponieważ do dyspozycji pozostało jedynie 16 bitów, na których można zapisać maksymalnie liczbę 65535. Przy próbie zapisania większego pliku program zwróci błąd, mówiący o zbyt dużym pliku.

Program został także sprawdzony pod kątem wycieków pamięci przy pomocy narzędzia `valgrind`. Wyniki tego testu, napierw dla komendy -w, a potem -r pozostawiam poniżej:

```
    ==33417== Memcheck, a memory error detector
==33417== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward
==33417== et al.
==33417== Using Valgrind-3.16.0.GIT and LibVEX; rerun with -h
==33417== for copyright info
==33417== Command: ./main -w -i data.bin -p
==33417== ../images/image16.bmp -o image_new.bmp -b 1
==33417==
==33417== HEAP SUMMARY:
==33417==     in use at exit: 0 bytes in 0 blocks
==33417==   total heap usage: 13 allocs, 13 frees, 5,323,078 bytes
==33417==   allocated
==33417== All heap blocks were freed -- no leaks are possible
==33417==
==33417== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

==33769== Memcheck, a memory error detector
==33769== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward
==33769== et al.
==33769== Using Valgrind-3.16.0.GIT and LibVEX; rerun with -h
==33796== for copyright info
==33769== Command: ./main -r -p image_new.bmp -o newdata.bin -b 1
==33769==
```

```
Odczytywanie ukrytego pliku z pliku BMP
Dane zapisane poprawnie!
==33769==
==33769== HEAP SUMMARY:
==33769==     in use at exit: 0 bytes in 0 blocks
==33769==   total heap usage: 10 allocs, 10 frees, 5,314,670 bytes
==33796==   allocated
==33769==
==33769== All heap blocks were freed -- no leaks are possible
==33769==
==33769== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

6 Wnioski

Program korzystający ze steganografii do ukrywania danych w obrazach nie jest bardzo skomplikowanym programem, jednak częstość korzystania z masek bitowych oraz pewne cechy plików BMP takie jak potrzeba wyrównania pliku do wielokrotności 4 bajtów sprawiły mi pewne problemy. Mój pomysł zakładający używanie wolnych miejsc w nagłówku `BMPHEADER` także spowodował, że możliwe jest zapisanie tylko plików o maksymalnym rozmiarze 65kB. Przy testowaniu na innym środowisku graficznym dla systemu Linux zauważyłem, że domyślne narzędzia dla środowiska KDE Plasma 5 nie potrafią odczytać plików BMP 16 i 32-bitowych, a program GIMP w dziwny sposób interpretuje maski bitowe dla obrazów 32-bitowych. Używając narzędzia Eye Of Gnome byłem w stanie bezproblemowo wyświetlić dowolne obrazy 16, 24 i 32-bitowe przed i po ukryciu informacji. Podejrzewam, że te narzędzia w inny sposób interpretują nagłówki plików BMP

Literatura

[1] Microsoft (2018) *Bitmap Reference - Win32 apps — Microsoft Docs*, `https://docs.microsoft.com/pl-pl/windows/win32/gdi/bitmap-reference`

[2] S. Leonard, Penango Inc (September 2016) (ISSN: 2070-1721) *RFC 7903 - Windows Image Media Types*, `https://tools.ietf.org/html/rfc7903`

Dodatek
Szczegółowy opis typów i funkcji

Steganografia w pliku .BMP

Wygenerowano przez Doxygen 1.8.18

Rozdział 1

Projekt - steganografia w pliku BMP

1.1 Założenia

Projekt steganografia w pliku BMP polegał na napisaniu programu, który umieszczał w pliku BMP dowolny plik podany przez użytkownika. Program potrafi "ukrywać" jedynie pliki wielkości do 65kB w plikach BMP o głębi kolorów wyższej niż 8 (16, 24, 32).

1.2 Limitacje

Program korzysta wyłącznie z obrazów o głębi kolorów wyższej niż 8, ponieważ dla głębi kolorów 8 oraz niżej, obrazy BMP korzystają z tablicy kolorów, gdzie każdy kolor jest jawnie wpisany i każda zmiana nawet jednego bitu bardzo znacząco wpływa na zmianę koloru danego pixela.

Program ukrywa pliki nie większe niż 65kB, ponieważ wykorzystuje puste miejsca w nagłówkach plików BMP do przechowywania wielkości ukrytego pliku, niestety do wykorzystania jest tam jedynie 2 bajty, co pozwala na zapisanie maksymalnej wielkości pliku właśnie 65kB.

1.3 Użycie

Po sklonowaniu repozytorium należy skompilować program korzystając z narzędzia `make`. Po wywołaniu `make` w głównym katalogu projektu powinien utworzyć się podkatalog `bin`. Tam znajduje się plik wywoływalny programu `main`, w przypadku korzystania z systemu MS Windows plik wykonywalny `main.exe`.

Do kompilacji projektu wymagane jest posiadanie kompilatora GCC w standardzie przynajmniej C89 oraz narzędzie `make`

Program można wywołać z flagą `-h`, wtedy wyświetli się poniższa pomoc:

```
Pierwszy argument wywołania programu powninien zawierać następujące flagi: -w lub -r
Flaga -w służy do zapisywania danych w obrazie
Flagi, które są po niej wymagane są nastepujące:
-i -- po niej należy podać plik z danymi, które program ma ukryć
-p -- po niej należy podać plik obrazu, w którym ma zostać ukryty plik
-o -- po niej należy podać plik wynikowy, w którym będzie ukryty plik
-b -- po niej należy podać na ilu bitach ma zostać zapisana informacja (obsługiwane liczby bitów: 1, 2, 4, 8)
Flaga -r służy do odczytywania danych z obrazu
Flagi, które są po niej wymagane są następujące:
-p -- po niej nalezy podać plik obrazu, z którego ma zostać odczytany ukryty plik
-o -- po niej należy podać plik wynikowy, do którego na zostać odczytana informacja
-b -- po niej nalezy podać na ilu bitach została zapisana ukryta informacja
Obsługiwane pliki BMP: 16bit[RGB(1,5,5,5)], 24bit[RGB(8,8,8)], 32bit[ARGB(8,8,8,8), XRBG(8,8,8,8)]
```

Rozdział 2

Indeks struktur danych

2.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja struktur danych

4.1 Dokumentacja struktury _BITFILEDS

`#include <structures.h>`

Pola danych

- uint32_t redMask
- uint32_t greenMask
- uint32_t blueMask
- uint32_t xMask
- uint32_t alphaMask

4.1.1 Opis szczegółowy

struktura zawierająca maski BITFIELDS

4.1.2 Dokumentacja pól

4.1.2.1 alphaMask

`uint32_t _BITFILEDS::alphaMask`

maska przeźroczystości

4.1.2.2 blueMask

`uint32_t _BITFILEDS::blueMask`

maska koloru niebieskiego

4.1.2.3 greenMask

`uint32_t _BITFILEDS::greenMask`

maska koloru zielonego

4.1.2.4 redMask

`uint32_t _BITFILEDS::redMask`

maska koloru czerwonego

4.1.2.5 xMask

`uint32_t _BITFILEDS::xMask`

maska zastępcza

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/structures.h

4.2 Dokumentacja struktury _BMPHEADER

`#include <structures.h>`

Pola danych

- char * HEADER
- uint32_t SIZE
- char * RESERVED1
- uint16_t RESERVED2
- uint32_t OFFSET

4.2.1 Opis szczegółowy

struktura zawierająca dane ze standardowego nagłówka pliku BMP

4.2.2 Dokumentacja pól

4.2.2.1 HEADER

```
char* _BMPHEADER::HEADER
```

nagłówek pliku, rozpoznaje typ pliku, jeżeli nie jest "BM", nie jest plikiem BMP standardu Windows NT

4.2.2.2 OFFSET

```
uint32_t _BMPHEADER::OFFSET
```

offset pliku pod którym można znaleźć tablicę pixeli

4.2.2.3 RESERVED1

```
char* _BMPHEADER::RESERVED1
```

pamięć zarezerwowana, używane do zapisania informacji o ukrytym pliku (SG)

4.2.2.4 RESERVED2

```
uint16_t _BMPHEADER::RESERVED2
```

pamięć zarezerwowana, używana do zapisywania wielkości ukrytego pliku

4.2.2.5 SIZE

```
uint32_t _BMPHEADER::SIZE
```

wielkość pliku BMP w bajtach

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/structures.h

4.3 Dokumentacja struktury _DIBHEADER

```
#include <structures.h>
```

Pola danych

- uint32_t DIBSIZE
- uint32_t WIDTH
- uint32_t HEIGHT
- uint16_t COLORPLANES
- uint16_t BITSPERPIXEL
- uint32_t COMPRESSION
- uint32_t SIZE
- uint32_t HORIZONTALRES
- uint32_t VERTICALRES
- uint32_t COLORS
- uint32_t IMPORTANTCOLORS

4.3.1 Opis szczegółowy

struktura zawierająca podstawowy nagłówek w standardzie DIBHEADER BITMAPINFOHEADER dla Windows

4.3.2 Dokumentacja pól

4.3.2.1 BITSPERPIXEL

`uint16_t _DIBHEADER::BITSPERPIXEL`

głębia kolorów, ile bitów przypada na pixel

4.3.2.2 COLORPLANES

`uint16_t _DIBHEADER::COLORPLANES`

tablica koloru

4.3.2.3 COLORS

`uint32_t _DIBHEADER::COLORS`

liczba kolorów w palecie, 0 dla wartości domyślnej 2^n

4.3.2.4 COMPRESSION

`uint32_t _DIBHEADER::COMPRESSION`

kompresja, rodzaj algorytmu

4.3.2.5 DIBSIZE

`uint32_t _DIBHEADER::DIBSIZE`

wielkość nagłówka w bajtach

4.3.2.6 HEIGHT

`uint32_t _DIBHEADER::HEIGHT`

wysokość obrazu w pixelach (ze znakiem)

4.3.2.7 HORIZONTALRES

`uint32_t _DIBHEADER::HORIZONTALRES`

rozdzielczość na wysokość

4.3.2.8 IMPORTANTCOLORS

`uint32_t _DIBHEADER::IMPORTANTCOLORS`

"ważne kolory", wartość zwykle pomijana, 0 dla wszystkich kolorów jako ważnych

4.3.2.9 SIZE

`uint32_t _DIBHEADER::SIZE`

wielkość pliku, dopuszczajna wartość "dummy" wynosząca 0

4.3.2.10 VERTICALRES

`uint32_t _DIBHEADER::VERTICALRES`

rozdzielczość na szerokość

4.3.2.11 WIDTH

`uint32_t _DIBHEADER::WIDTH`

szerokość obrazu w pixelach (ze znakiem)

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/structures.h

4.4 Dokumentacja struktury _SGHEADER

`#include <structures.h>`

Pola danych

- char * HEADER
- uint16_t FILESIZE

4.4.1 Opis szczegółowy

struktura opisująca nagłówek specyficzny dla pliku z ukrytym plikiem, zapisywana zawsze na pierwszych 72 pojedynczych bitach tablicy pixeli

4.4.2 Dokumentacja pól

4.4.2.1 FILESIZE

```
uint16_t _SGHEADER::FILESIZE
```

wielkość pliku (1 bajt)

4.4.2.2 HEADER

```
char* _SGHEADER::HEADER
```

nagłówek steganograficzny, jeżeli jest równy "SG\0", plik zawiera ukryte informacje (wielkość: 2 bajty)

Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/structures.h

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku src/functions.c

implemenctacje funkcji

```
#include "../include/functions.h"
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Funkcje

- int GCD (int a, int b)
- int LCM (int a, int b)
- int read_headers (FILE *filePointer, BMPHEADER *bmpHeader, DIBHEADER *dibHeader)
- uint8_t * pixelArray_read_16bit (FILE *filePointer, int pixelCount)
- uint8_t * pixelArray_read_24bit (FILE *filePointer, int pixelCount)
- uint8_t * pixelArray_read_32bit (FILE *filePointer, int pixelCount)
- uint8_t * read_file_to_memory (FILE *filePointer, int *filesize, int *arrsize)
- SGHEADER **header_generate** (uint32_t filesize)
- uint8_t * prepare_data_to_write (uint8_t *dataTab, int bitsPerPixel, int oldArrSize, int *newArrSize)
- BMPHEADER **prepare_new_header** (BMPHEADER bmpHeader, SGHEADER sgHeader, DIBHEADER dibHeader)
- int write_headers (BMPHEADER bmpHeader, DIBHEADER dibHeader, FILE *filePointer)
- int write_data_32bit (uint8_t *dataTab, uint8_t *pixelArray, BMPHEADER header, DIBHEADER dibHeader, FILE *filePointer, int pixels, int arrSize, int bitsPerPixel)
- int write_data_24bit (uint8_t *dataTab, uint8_t *pixelArray, BMPHEADER header, DIBHEADER dibHeader, FILE *filePointer, int pixels, int arrSize, int bitsPerPixel)
- int write_data_16bit (uint8_t *dataTab, uint8_t *pixelArray, BMPHEADER header, DIBHEADER dibHeader, FILE *filePointer, int pixels, int arrSize)
- void write_data_to_image (uint8_t *pixelArray, uint8_t *dataArray, BMPHEADER bmpHeader, DIBHEADER dibHeader, FILE *filePointer, uint16_t dataFileSize, int bitsPerPixelStegano, int bitsPerPixel, int pixels, int arrSize)
- int check_for_stegano (BMPHEADER bmpHeader)
- uint8_t * read_data_from_steganofile_24_32bit (uint8_t *pixelArray, BMPHEADER bmpheader, int bitsPer↩ Pixel)
- uint8_t * read_data_from_steganofile_16bit (uint8_t *pixelArray, BMPHEADER bmpheader, int bitsPerPixel)
- int write_data_array_to_file (uint8_t *dataArray, int dataSize, FILE *filePointer)
- int write_data_from_image (BMPHEADER bmpHeader, uint8_t *pixelArray, FILE *filePointer, DIBHEADER dibHeader, int bitsPerPixel)

5.1.1 Opis szczegółowy

implemenctacje funkcji

Autor

Wojciech Janota

Wersja

0.1

Data

2020-05-01

Copyright

5.1.2 Dokumentacja funkcji

5.1.2.1 check_for_stegano()

```
int check_for_stegano (
            BMPHEADER bmpHeader )
```

funkcja sprawdzająca, czy w nagłówku podanego pliku występuje nagłówek steganograficzny

Parametry

bmpHeader	nagłówek BMPHEADER

Zwraca

1 jeżeli prawda, 0 jeżeli fałsz

5.1.2.2 GCD()

```
int GCD (
            int a,
            int b )
```

pomocnicza funkcja NWD

Parametry

a	pierwsza liczba
b	druga liczba

Zwraca

NWD

5.1.2.3 LCM()

```
int LCM (
            int a,
            int b )
```

pomocnicza funkcja NWW

Parametry

a	pierwsza liczba
b	druga liczba

Zwraca

NWW

5.1.2.4 pixelArray_read_16bit()

```
uint8_t* pixelArray_read_16bit (
            FILE * filePointer,
            int pixelCount )
```

funkcja zwracająca tablicę pixeli dla pliku z 16-bitową głębią kolorów

Parametry

filePointer	wskaźnik na plik obrazu
pixelCount	licznik pixeli w obrazie

Zwraca

tablica pixeli rozbita na poszczególne składowe

5.1.2.5 pixelArray_read_24bit()

```
uint8_t* pixelArray_read_24bit (
            FILE * filePointer,
            int pixelCount )
```

funkcja zwracająca tablicę pixeli dla pliku z 24-bitową głębią kolorów

Parametry

filePointer	wskaźnik na plik obrazu
pixelCount	licznik pixeli w obrazie

Zwraca

tablica pixeli rozbita na poszczególne składowe

5.1.2.6 pixelArray_read_32bit()

```
uint8_t* pixelArray_read_32bit (
            FILE * filePointer,
            int pixelCount )
```

funkcja zwracająca tablicę pixeli dla pliku z 32-bitową głębią kolorów

Parametry

filePointer	wskaźnik na plik obrazu
pixelCount	licznik pixeli w obrazie

Zwraca

tablica pixeli rozbita na poszczególne składowe

5.1.2.7 prepare_data_to_write()

```
uint8_t* prepare_data_to_write (
            uint8_t * dataTab,
            int bitsPerPixel,
            int oldArrSize,
            int * newArrSize )
```

funkcja dzieląca tablicę bajtów odczytanych z pliku wejściowego na paczki bitów na osobnych bajtach

Parametry

dataTab	tablica z odczytanymi bajtami z pliku wejściowego
bitsPerPixel	na ilu bitach ma być zapisana informacja
oldArrSize	wielkość starej tablicy bajtów
newArrSize	nowa wielkość końcowej tablicy z danymi

Zwraca

końcowa tablica danych

5.1.2.8 read_data_from_steganofile_16bit()

```
uint8_t* read_data_from_steganofile_16bit (
            uint8_t * pixelArray,
            BMPHEADER bmpheader,
            int bitsPerPixel )
```

funkcja odczytująca ukryte dane z obrazów 16-bitowych

Parametry

pixelArray	tablica pixeli obrazu wejściowego
bmpheader	nagłówek BMPHEADER obrazu wejściowego
bitsPerPixel	liczba bitów na których została zapisana informacja

Zwraca

tablica bajtów danych odczytanych z obrazu

5.1.2.9 read_data_from_steganofile_24_32bit()

```
uint8_t* read_data_from_steganofile_24_32bit (
            uint8_t * pixelArray,
            BMPHEADER bmpheader,
            int bitsPerPixel )
```

funkcja odczytująca ukryte dane z obrazów 24 i 32-bitowych

Parametry

pixelArray	tablica pixeli obrazu wejściowego
bmpheader	nagłówek BMPHEADER obrazu wejściowego
bitsPerPixel	liczba bitów na których została zapisana informacja

Zwraca

tablica bajtów danych odczytanych z obrazu

5.1.2.10 read_file_to_memory()

```
uint8_t* read_file_to_memory (
            FILE * filePointer,
            int * filesize,
            int * arrSize )
```

funkcja wczytująca plik bajt po bajcie do pamięci

Parametry

filePointer	wskaźnik na plik
filesize	wielkość pliku
arrSize	wielkość tablicy z bajtami

Zwraca

tablica bajtów z plikiem wejściowym

5.1.2.11 read_headers()

```
int read_headers (
            FILE * filePointer,
            BMPHEADER * bmpHeader,
            DIBHEADER * dibHeader )
```

funkcja wczytująca dane z nagłówków

Parametry

filePointer	wskaźnik na plik, z którego odczytane mają zostać nagłówki
bmpHeader	wskaźnik na nagłówek główny
dibHeader	wskaźnik na nagłówek DIB

Zwraca

int jeżeli wczytywanie się powiedzie, zwracana jest wartość 1, w przeciwnym wypadku zwracana jest wartość 0

5.1.2.12 write_data_16bit()

```
int write_data_16bit (
            uint8_t * dataTab,
            uint8_t * pixelArray,
            BMPHEADER header,
            DIBHEADER dibHeader,
            FILE * filePointer,
            int pixels,
            int arrSize )
```

funkcja zapisująca tablicę pixeli z podmienionymi bitami do pliku wynikowego dla obrazów 16-bitowych

Parametry

dataTab	tablica podzielonych danych
pixelArray	tablica pixeli
header	nagłówek BMPHEADER
dibHeader	nagłówek DIBHEADER
filePointer	wskaźnik na plik wynikowy
pixels	liczba pixeli w obrazie
arrSize	wielkość tablicy danych

Zwraca

1 jeżeli funkcja wykonała się poprawnie, 0 w przeciwnym wypadku

5.1.2.13 write_data_24bit()

```
int write_data_24bit (
            uint8_t * dataTab,
            uint8_t * pixelArray,
            BMPHEADER header,
            DIBHEADER dibHeader,
            FILE * filePointer,
            int pixels,
            int arrSize,
            int bitsPerPixel )
```

funkcja zapisująca tablicę pixeli z podmienionymi bitami do pliku wynikowego dla obrazów 24-bitowych

Parametry

dataTab	tablica podzielonych danych
pixelArray	tablica pixeli
header	nagłówek BMPHEADER
dibHeader	nagłówek DIBHEADER
filePointer	wskaźnik na plik wynikowy
pixels	liczba pixeli w obrazie
arrSize	wielkość tablicy danych

Zwraca

1 jeżeli funkcja wykonała się poprawnie, 0 w przeciwnym wypadku

5.1.2.14 write_data_32bit()

```
int write_data_32bit (
            uint8_t * dataTab,
            uint8_t * pixelArray,
            BMPHEADER header,
            DIBHEADER dibHeader,
            FILE * filePointer,
            int pixels,
            int arrSize,
            int bitsPerPixel )
```

funkcja zapisująca tablicę pixeli z podmienionymi bitami do pliku wynikowego dla obrazów 32-bitowych

Parametry

dataTab	tablica podzielonych danych
pixelArray	tablica pixeli
header	nagłówek BMPHEADER
dibHeader	nagłówek DIBHEADER
filePointer	wskaźnik na plik wynikowy
pixels	liczba pixeli w obrazie
arrSize	wielkość tablicy danych

Zwraca

1 jeżeli funkcja wykonała się poprawnie, 0 w przeciwnym wypadku

5.1.2.15 write_data_array_to_file()

```
int write_data_array_to_file (
            uint8_t * dataArray,
            int dataSize,
            FILE * filePointer )
```

funkcja zapisująca dane z tablicy bajtów do pliku wynikowego

Parametry

dataArray	tablica bajtów odczytanych z obrazu
dataSize	wielkośc tablicy bajtów
filePointer	wskaźnik na plik wynikowy

Zwraca

1 jeżeli funkcja wykonała się poprwanie, 0 w przeciwnym wypadku

5.1.2.16 write_data_from_image()

```
int write_data_from_image (
            BMPHEADER bmpHeader,
            uint8_t * pixelArray,
            FILE * filePointer,
            DIBHEADER dibHeader,
            int bitsPerPixel )
```

funkcja wrapper zapisująca ukryte dane z przekazanej tablicy pixeli do pliku

Parametry

bmpHeader	nagłówek BMPHEADER
pixelArray	tablica pixeli obrazu wejściowego
filePointer	wskaźnik na plik wynikowy
dibHeader	nagłówek DIBHEADER
bitsPerPixel	liczba bitów na których zostały zapisane dane

Zwraca

5.1.2.17 write_data_to_image()

```
void write_data_to_image (
            uint8_t * pixelArray,
            uint8_t * dataArray,
            BMPHEADER bmpHeader,
            DIBHEADER dibHeader,
            FILE * filePointer,
            uint16_t dataFileSize,
            int bitsPerPixelStegano,
            int bitsPerPixel,
            int pixels,
            int arrSize )
```

funkcja wrapper służąca do wpisywania danych z pliku do obrazu

Parametry

pixelArray	tablica pixeli z obrazu wejściowego
dataArray	tablica bajtów pliku wejściowego

Parametry

bmpHeader	nagłówek BMPHEADER obrazu wejściowego
dibHeader	nagłówek DIBHEADER obrazu wejściowego
filePointer	wskaźnik na plik obrazu wyjściowego
dataFileSize	wielkość pliku wejściowego
bitsPerPixelStegano	na ilu bitach ma zostać zapisana informacja
bitsPerPixel	głębia kolorów obrazu
pixels	liczba pixeli
arrSize	wielkośc tablicy bajtów pliku wejściowego

5.1.2.18 write_headers()

```
int write_headers (
            BMPHEADER bmpHeader,
            DIBHEADER dibHeader,
            FILE * filePointer )
```

funkcja zapisująca nagłówki do pliku

Parametry

bmpHeader	nagłówek BMPHEADER
dibHeader	nagłówek DIBHEADER
filePointer	wskaźnik na plik wynikowy

Zwraca

1 jeżeli wykonano poprawnie, 0 jeżeli wystąpił błąd

Indeks