Тема на проектна работа

**Криптиране и декриптиране на**

**информация във визуални файлове**

Изготвена от

**Филип Андонов**

Факултетен №: **121222054**

Специалност: **„Компютърно и софтуерно инженерство“**

Факултет: **„Компютърни системи и технологии“**

Група: **42 „б“**

**Въведение**

Програмата има за цел да кодира и декодира информация от снимка, без да променя видимо илюстрирането ѝ. Такъв клас решения могат да намерят приложение в областта на борбата с корпоративен или друг вид шпионаж. Това е постигнато чрез стеганографския метод на най-младшия бит. Идеята при него е, че информацията се записва в най-младшия бит на цветовата стойност на всеки един пиксел. Така цветът остава непроменен за човешкото око, но пренася данни.

За снимков формат – носител е избран PNG с дълбочина 8 бита на канал, RGBA цветова гама, без interlace функционалност. Той е широко използван в пренасянето на данни през интернет, документиран е добре и е широко поддържан. Причините за този избор пред други формати като jpeg, bmp и tiff са, че при компресия данни не се губят, компресията става бързо и е добре оптимизирана от гледна точка на памет и стандартът на формата е наподобяващ виртуална памет на сектори, което това го прави удобен за работа.

Diagram

Description automatically generated Логическата последователност на софтуера е описана в блок схемите на Фигура 1, 2 и 3. Управлението става чрез подаване на начални аргументи при извикването ѝ. Те са описани в следващ абзац.

Фигура

Diagram

Description automatically generated

Фигура

Diagram

Description automatically generated

Фигура

Фигура

**Конвенции за наименуване на променливи**

Глобалните променливи започват с „g\_“.

Променливите, част от структури започват с „m\_“.

Булевите променливи започват с „is\_“.

Функции, структури и променливи са записани с малки букви. Ако името съдържа повече от една дума се отделя с „\_“.

Константи се пишат изцяло с главни букви.

Макросите, използвани за хедър гардове носят името на файла и са записани с главни букви. Завършват с „\_H“.

Сорс файловете и съответстващите им хедър файлове носят едно и също едно име.

**Файлова структура**

Файловете са подредени в две папки: „inc“ съдържа всички файлове, предназначени за include, а „src“ тези за компилация.

**Вход и изход на програмата**

Програмата може да работи в два режима. Първият е в режим на кодиране на данни в снимка, а вторият в режим на извличането им. За целта са дефинирани възможности за вход от потребител.

-i <input\_image> – Задължителен аргумент, указващ снимката, която ще се чете в режим на декодиране и снимката, в която ще се запише информацията в режим на кодиране.

-e <string> – Когато този флаг бива използван, програмата влиза в режим на кодиране и записва съобщението от <string> в копие на <input\_image> със същото име.

-o <output\_image> – Чрез този флаг се избира директория, в която ще бъде изведен изходния файл от програмата.

-d <text\_file> – Когато този флаг бива използван, програмата влиза в режим на декодиране и записва съобщението от <input\_image> във файла <text\_file>. Ако не бъде уточнено име на файла, тогава той бива кръстен „<input\_image>.txt“.

**Техническо описание на модулите на програмата**

Програмата се състои от 6 модула: *main, codec , png\_data\_encoder, png\_parser, png\_filtration, program\_input\_parser*

*main.c*– Съдържа една функция – **main**. Входните аргументи в нея са добавените команди при извикването на програмата, а изходът от функцията е число, показващо състоянието на програмата след изпълнение. Възможностите са 2: PROGRAM\_OK(0) и PROGRAM\_ERROR(1), дефинирани в *globals\_config.h*. Основното ѝ предназначение е да задейства процеса по обработка на входните данни, след което да задейства кодиране или декодиране на информация и да върне резултат обратно към операционната система.

*program\_input\_parser.h* – Съдържа декларации на функция **parse\_program\_input**, структура **program\_inp** и предефинирани константи, отговорни за обработката на входните данни. Структурата **program\_inp**, използвана за капсулирането и прехвърлянето на данните в останалата част на програмата, съдържа:

1. Указател **m\_input\_name** към текст, съдържащ името на снимката, която ще бъде обработена. Тя може да съдържа код, в зависимост от условието в което програмата е избрана да работи.
2. Указател **m\_output\_name** към текст, съдържащ името на снимката, която ще бъде създадена като изход от програмата, ако се избере опция за кодиране.
3. Булева променлива **m\_encode**, съдържаща информация за условието на работа, избрано от потребителя. True означава, че програмата ще кодира съобщение, а False, че ще прочете такова.
4. Указател **m\_operation\_argument** към текст, съдържащ съобщението, което ще се кодира при избран режим на кодиране или името на изходния файл, при избран режим на декодиране.
5. Цяло число **m\_error\_code**, индикатор за състоянието на входните данни. Той може да приема три стойности: PROGRAM\_INPUT\_PARSER\_OK(0), PROGRAM\_INPUT\_PARSER\_ERR\_CODE\_WRONG\_INPUT(1), PROGRAM\_INPUT\_PARSER\_ERR\_CODE\_NO\_INPUT(2), дефинирани в същия хедър файл.

*program\_input\_parser.c* – Съдържа дефиницията на **parse\_program\_input** и спомагателна функция **print\_help\_menu.**

**parse\_program\_input** – Основната функция, отговорна за извличането на входни данни. Създава и инициализира инстанция от вид **program\_inp** със стойности по подразбиране, дефинирани в рамките на същия файл, след което започват серия от проверки. Първата проверка е отговорна за изписването на помощна информация(дефинирана във функцията **print\_help\_menu**), ако такава е изисквана от потребител. Следва цикъл, обхождащ всички подадени от потребител аргументи, като серията от проверки в него гарантира, че изискванията за входни данни са покрити, а ако не са се извежда съобщение и се връща съответен код за грешка.

*codec.h* – Съдържа декларациите на функциите **encoding** и **decoding.**

*codec.c* – Съдържа дефинициите на функциите **encoding** и **decoding**.

**encoding** – Съдържа логиката на високо ниво, управляваща процеса на кодиране на информация в снимка. Последователно е описана логиката на функцията: извиква се функция за отваряне на снимка **png\_open** и се проверява, дали то е успешно. Заделя се автоматична памет и се записва информацията от IHDR на снимката чрез **read\_png\_IHDR**. Заделя се динамичен масив и се извлича цялата информация от IDAT секторите с **extract\_IDAT\_raw\_all** , след което се проверява дали операцията е била успешна. Снимката се затваря (**png\_close**), за да освободи този ресурс максимално бързо, тъй като той няма да бъде използван повече в рамките на тази програма. Данните се декомпресират посредством функция **uncompress\_data**, записват се в нова динамична памет и се проверяват отново, след което се освобождава паметта, съдържаща ги в компресиран вид. Извиква се функция **unfilter\_rgba\_png**, изпълняваща дефилтриращи операции върху декомпресираната информация, за да се възстанови снимката в подходящ за манипулация вид (**RGBA\_pixel** двуизмерна матрица), запаметява се в динамична памет, прави се проверка на данните и се освобождава паметта, съдържаща нефилтрирания вариант на снимката. Информацията от матрицата се манипулира, като към нея се добавя скритото съобщение с **encode\_data\_rgba** и се проверява отново. Следва заделяне на памет, филтрация **fitler\_rgba\_png**, проверка и освобождаване на матрицата. Филтрираната снимка се компресира **compress\_data**, прави се проверка на данните и се освобождава ненужната памет. Следва процес на записване на снимката, имплементиран от следните функции: **png\_open, write\_png\_IHDR, write\_png\_IDAT, write\_png\_IEND, png\_close**, освобождаване от памет и връщане на код за успешно изпълнение.

**decoding** – Съдържа логиката на високо ниво, управляваща процеса на декодиране на информация в снимка. Последователно е описана логиката на функцията: извиква се функция за отваряне на снимка **png\_open** и се проверява, дали то е успешно. Заделя се автоматична памет и се записва информацията от IHDR на снимката чрез **read\_png\_IHDR**. Заделя се динамичен масив и се извлича цялата информация от IDAT секторите с **extract\_IDAT\_raw\_all** , след което се проверява дали операцията е била успешна. Снимката се затваря (**png\_close**), за да освободи този ресурс максимално бързо, тъй като той няма да бъде използван повече в рамките на тази програма. Данните се декомпресират посредством функция **uncompress\_data**, записват се в нова динамична памет и се проверяват отново, след което се освобождава паметта, съдържаща ги в компресиран вид. Извиква се функция **unfilter\_rgba\_png**, изпълняваща дефилтриращи операции върху декомпресираната информация, за да се възстанови снимката в подходящ за манипулация вид (**RGBA\_pixel** двуизмерна матрица), запаметява се в динамична памет, прави се проверка на данните и се освобождава паметта, съдържаща нефилтрирания вариант на снимката. Скритата информация в матрицата бива прочетена и запаметена в динамична памет, посредством **decode\_data\_rgba**, и освободена. Скритото съобщение бива принтирано и изтрито. Функцията връща код за успешно изпълнение.

*png\_data\_encoder.h* – Съдържа декларации на функциите **encode\_data\_rgba** и **decode\_data\_rgba**.

*png\_data\_encoder.c* – Съдържа дефинициите на функциите **encode\_data\_rgba** и **decode\_data\_rgba**.

**encode\_data\_rgba** – Функция, записваща подадена информация в двуизмерна **RGBA\_pixel** матрица и връщаща True при успех и False при грешка. Записването се случва на два етапа, но преди това се проверява дали снимката има достатъчно свободни пиксели, които да съберат тази информация. Следва записване на големината, посредством побитови операции. Това се случва като се нулира последният бит на дадената цветова стойност с логическо „И“, за да се гарантира записването на числото посредством логическо „ИЛИ“ на по-късен етап. Винаги се работи с последния бит от големината, защото след всяко записване тя бива „отместена“ надясно със съответното отстояние и върху нея се извършва логическа операция „И“ с 0x01, за да се вземе само последният бит. След това той е записва посредством споменатата операция „ИЛИ“. Записването на новите данни се случва по същия начин.

**decode\_data\_rgba** – Функция, извличаща информация от подадената **RGBA\_pixel** двуизмерна матрица и връщаща True при успех и False при грешка. Четенето се случва отново на два етапа: първо се чете дължината, а после, след заделяне на динамична памет, и самата информация. Извличането на битовете става в следната последователност: Първо се взима последният бит във временна променлива чрез логическа операция „И“ между 0x01 и цветовата стойност, после този бит бива отместен наляво до съответната му позиция и добавен към масива с информация, посредством логическа операция „ИЛИ“. Важно е да се спомене, че тази функция добавя един символ „\0“, който не присъства във файла, за да окаже край на информацията при принтиране. Това може да бъде отстранено.

*png\_filtration.h* – Съдържа декларациите на **unfilter\_rgba\_png**, **filter\_rgba\_png** и структурата **RGBA\_pixel**.

**RGBA\_pixel** – Структура, съдържаща стойностите на стандартен RGBA пиксел. Той се състои от 4 канала – червен, зелен, син и алфа (прозрачност).

*png\_filtration.c* – Съдържа дефинициите на функциите, отговорни за филтриране и дефилтриране на снимка. Две от тях **unfilter\_rgba\_png** и **filter\_rgba\_png** са видими отвъд компилационния файл, другите са налични само в рамките му.

**filter\_rgba\_pnga** – Функция, извеждаща филтриран масив от RGBA матрица с пиксели. Работи по следния начин: изчислява големината на масива, който ще бъде върнат, заделя динамична памет, проверява паметта, създава временен -1ви ред, който да играе роля на предишен и го инициализира със стойности 0, защото изискване по стандарт е всяка стойност на елемент извън рамките на снимката да се разглежда като 0. Прави се повикване към **calc\_filter\_type**, за да се определи филтърът, който ще бъде използван за ред 0 и съответно после се влиза в една от следните функции, отговорни за филтриране на един ред: **apply\_rgba\_png\_filter\_none, apply\_rgba\_png\_filter\_sub, apply\_rgba\_png\_filter\_up, apply\_rgba\_png\_filter\_average, apply\_rgba\_png\_filter\_paeth**. Следва освобождаване на временно заделената памет. След като се обработи първият ред, започва обработка на всеки следващ със същите функции.

**unfilter\_rgba\_png** – Функция, извеждаща нефилтрирана RGBA матрица, съдържаща стойностите на пикселите от снимката в суров вид. Работи по следния начин: изчислява големината и заделя динамична памет за матрицата, проверява я, заделя и инициализира временна памет за -1ви ред като по стандарт, дефилтрира първи ред(**strip\_filter\_per\_rgba\_row**), запазва го и освобождава временната памет. Следва дефилтриране и запазване на следващите редове.

**strip\_filter\_per\_rgba\_row** – Дефилтрира един ред. Процесът е: заделя се памет за реда, проверява се, чете се филтърът на реда (Той се намира на елемент 0 по стандарт) и се извиква една от следните функции, която да дефилтрира реда: **reverse\_rgba\_png\_filter\_none, reverse\_rgba\_png\_filter\_sub, reverse\_rgba\_png\_filter\_up, reverse\_rgba\_png\_filter\_average, reverse\_rgba\_png\_filter\_paeth**.

**calc\_filter\_type** – Функция калкулираща вида на филтъра, който да бъде приложен. Това става като се избере най-малкият резултат след прилагане на петте възможни филтъра върху един цялостен ред. За прилагане на филтър са използвани функциите **apply\_heur\_png\_filter\_none, apply\_heur\_png\_filter\_sub, apply\_heur\_png\_filter\_up, apply\_heur\_png\_filter\_average, apply\_heur\_png\_filter\_paeth**, за сумирането – **sum\_row\_for\_heuristics**, а за намиране на най-малък елемент – **find\_min**.

**find\_min** – Намиране на минимален елемент с линейна итерация и връщането на индекса му в масива.

**sum\_row\_for\_heuristics** – Намиране на сума на ред, като всеки елемент по-голям от 127 бива изваден от 256 и резултатът бива добавен към общата сума.

**apply\_heur\_png\_filter\_none** – Функция, копираща информацията без да прилага филтър.

Формула: Filt(x) = Recon(x);

**apply\_heur\_png\_filter\_sub** – Функция, прилагаща филтър sub на цял ред, без да взема предвид цветовите канали.

Формула: Filt(x) = (Orig(x) - Orig(left)) % 256;

**apply\_heur\_png\_filter\_up** – Функция, прилагаща филтър up на цял ред, без да взема предвид цветовите канали.

Формула: Filt(x) = (Orig(x) - Orig(upper)) % 256;

**apply\_heur\_png\_filter\_average** – Функция, прилагаща филтър average, без да взема предвид цветовите канали.

Формула: Filt(x) = (Orig(x) - floor((Orig(left) + Orig(upper)) / 2)) % 256;

**apply\_heur\_png\_filter\_paeth** – Функция, прилагаща филтър paeth, без да взема предвид цветовите канали.

Формула: Filt(x) = (Orig(x) - PaethPredictor(Orig(left), Orig(upper), Orig(upper\_left))) % 256;

**apply\_rgba\_png\_filter\_none** – Функция, копираща информацията без да прилага филтър.

Формула: Filt(x) = Recon(x);

**apply\_rgba\_png\_filter\_sub** – Функция, прилагаща филтър sub за всеки цветови канал отделно.

Формула: Filt(x) = (Orig(x) - Orig(left)) % 256;

**apply\_rgba\_png\_filter\_up** – Функция, прилагаща филтър up за всеки цветови канал отделно.

Формула: Filt(x) = (Orig(x) - Orig(upper)) % 256;

**apply\_rgba\_png\_filter\_average** – Функция, прилагаща филтър average за всеки цветови канал отделно.

Формула: Filt(x) = (Orig(x) - floor((Orig(left) + Orig(upper)) / 2)) % 256;

**apply\_rgba\_png\_filter\_paeth** – Функция, прилагаща филтър paeth за всеки цветови канал отделно.

Формула: Filt(x) = (Orig(x) - PaethPredictor(Orig(left), Orig(upper), Orig(upper\_left))) % 256;

**reverse\_rgba\_png\_filter\_none** – Функция, копираща информацията без да премахва филтър.

Формула: Recon(x) = Filt(x);

**reverse\_rgba\_png\_filter\_sub** – Функция, премахваща филтър sub за всеки цветови канал.

Формула: Recon(x) = (Filt(x) + Recon(left)) % 256;

**reverse\_rgba\_png\_filter\_up** – Функция, премахваща филтър up за всеки цветови канал.

Формула: Recon(x) = (Filt(x) + Recon(upper)) % 256;

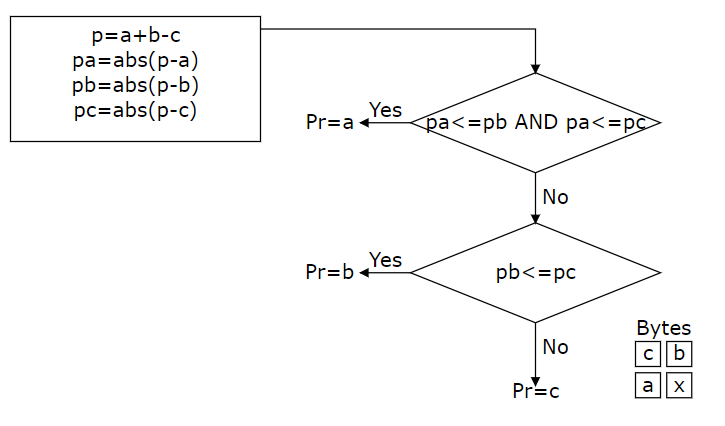
**reverse\_rgba\_png\_filter\_average** – Функция, премахваща филтър average за всеки цветови канал.

Формула: Filt(x) = (Orig(x) - floor((Orig(left) + Orig(upper)) / 2)) % 256;

**reverse\_rgba\_png\_filter\_paeth** – Функция, премахваща филтър paeth за всеки цветови канал.

Формула: Filt(x) = (Orig(x) - PaethPredictor(Orig(left), Orig(upper), Orig(upper\_left))) % 256;

**paeth\_predictor** – (Фигура 2) Функция, избираща стойност за paeth филтриране и дефилтриране. Изчисляват се три коефициента, след което се връща стойността генерирала най-малкия.



Фигура 4

*png\_parser.h* – Съдържа декларации на функциите и структурите, служещи за отваряне, четене, декомпресиране, компресиране, записване и затваряне на снимка.

*png\_parser.c* – Модулът работи на принципа на четене в паметта. Две глобални променливи следят за отварянето на файла. Едната е от вид файлов указател към снимката, а другата – булева променлива, помнеща дали снимката е отворена. Дизайнът на функциите е написана така, че да връща в началната позиция файловия указател, освен ако характера ѝ не го налага. Следва описание на функциите, имплементирани в този файл:

**change\_endianness** – Сменя редът на байтовете в дадено съобщение. Дефинирана е ако системата е обозначена като SMALL ENDIAN.

**chunk\_pointer\_get** – Връща указател към файла

**chunk\_pointer\_reset** – Връща указателя в началото на файла

**chunk\_pointer\_set** – Позиционира файловия указател на дадения виртуален адрес

**png\_open** – Извиква стандартната функция **fopen**

**png\_close** – Извиква стандартната функция **fclose**

**get\_outside\_chunk** – Тази функция извлича външната информацията за всеки chunk и я връща в структура от вид **outside\_chunk**. Прави се проверка дали файлът е отворен, после се прочита дължината на данните в него, видът му, прескача се самият блок с информация, ако има такъв, прочита се CRC-32 и се връща резултат.

**chunk\_seek** – Функция, намираща определен вид блок. Ако бъде подаден флаг **PNG\_PARSER\_RESET** търсенето започва от началото на файла, а при **PNG\_PARSER\_SET** от последната дестинация на файловия указател.

**read\_png\_IHDR** – Функция, четяща IHDR в **IHDR\_chunk** структура. Търсенето става чрез **chunk\_seek**, след което се прочита и самата информация в ред, дефиниран в стандарта.

**read\_png\_IDAT** – Функция, намираща следващия IDAT chunk и запазваща виртуалния адрес на данните в него във структура **IDAT\_chunk**.

**extract\_IDAT\_raw** – Функция, генерираща масив от компресирана информация, прочетена от снимка. Функцията заделя динамична памет, намира следващия IDAT chunk посредством **chunk\_seek** и запазва информацията от него в паметта.

**extract\_IDAT\_raw\_all** – Функция, извличаща цялата информация от всички IDAT сектори. Тя заделя динамичен масив, в който запазва информацията посредством **extract\_IDAT\_raw**.

**uncompress\_data** – Wrapper функция, капсулираща извикването на **uncompress** от библиотеката zlib.

**compress\_data** – Wrapper функция, капсулираща извикването на **compress** от библиотеката zlib.

Важно: следващите три функции не запазват състоянието на указателя преди извикването им.

**write\_png\_IHDR** – Функция, записваща подаден **IHDR\_chunk** в началото на файла.

**write\_png\_IDAT** – Функция, записваща подадените компресирани данни на мястото на файловия указател. Това се случва след като се изчисли новата CRC-32 стойност, посредством **crc32** от библиотеката zlib.

**write\_png\_IEND** – Функция, записваща IEND сектор на мястото на файловия указател.

**Компилация**

Програмата е компилирана с GCC toolchain под Windows 10 (използван е MinGW). Компилационната команда е:

*gcc \*.c -Wextra -Wall -o {output\_file\_name} -I {headers\_path} -L {source\_path} -lz*

**Библиотеки**

В този проект са използвани стандартната библиотека и zlib (<https://www.zlib.net/>). Втората бива употребена за компресиране и декомпресиране на файлове. Причината за избор на точно тази библиотека е, че е много широко използвана, добре написана, документирана и оптимизирана. Налага се използването ѝ, защото предоставя лесен интерфейс към LZSS и Huffman алгоритми, използвани от PNG стандарта.

**Източници**

https://en.wikipedia.org/wiki/Steganography

https://en.wikipedia.org/wiki/PNG

https://www.w3.org/TR/2003/REC-PNG-20031110/

https://www.zlib.net/manual.html