LAP Projeto 2 - C

/\*

max width = 80 colunas

tab = 4 spaces

01234567890123456789012345678901234567890123456789012345678901234567890123456789

LAP - AMD 2019

secrets.c

------

Student 1: Tiago Gomes Cotovio - 52349

Student 2: Lourenco Ramos Duarte - 53035

Comment:

Os objetivos do projeto foram atingidos, uma vez que conseguimos realizar todas

as funcoes pedidas e, apos testarmos o programa com testes diversificados

(tentando englobar todos os casos possiveis), verificamos que os resultados

obtidos foram os esperados.

\*/

#include "Secrets.h"

// shift left and insert a new least significant bit

#define shiftl(byte, newbit) ((byte)\*2 + (newbit))

// most significant bit

#define msb(byte) (((signed char)(byte)) < 0 ? 1 : 0)

#define READ\_B "rb"

#define WRITE\_B "wb"

/\* FUNCTIONS String \*/

/\* FUNCTIONS Byte \*/

/\* FUNCTIONS Int2 \*/

/\* FUNCTIONS Pixel \*/

/\* FUNCTIONS Image \*/

/\* ENCRYPTION \*/

/\* Auxiliar functions \*/

static bool open\_file(String src, String dst, FILE \*\*f, FILE \*\*g) {

if ((\*f = fopen(src, READ\_B)) == NULL) {

error("File not found or invalid file '%s'", src);

return false;

}

if ((\*g = fopen(dst, WRITE\_B)) == NULL) {

fclose(\*f);

error("File not found or invalid file '%s'", dst);

return false;

}

return true;

}

static int getKey(FILE \*p) {

char key;

if ((key = fgetc(p)) == EOF) {

rewind(p);

fgetc(p); // dont care about '3'

fgetc(p); // dont care about '.'

key = fgetc(p);

}

key = atoi(&key);

return key;

}

static int fpeek(FILE \*f) {

int c;

if ((c = fgetc(f)) != EOF) ungetc(c, f); // pushes c back to stream buffer

return c;

}

static void makeColorMask(Pixel \*p, int bit, int color) {

switch (color % 3) {

case 0:

if (bit)

p->red |= 0b01;

else

p->red &= 0b11111110;

break;

case 1:

if (bit)

p->green |= 0b01;

else

p->green &= 0b11111110;

break;

case 2:

if (bit)

p->blue |= 0b01;

else

p->blue &= 0b11111110;

break;

}

}

static int getMaskedBit(Pixel \*p, int color) {

int bit = 0;

switch (color % 3) {

case 0:

bit = p->red & 0b01;

break;

case 1:

bit = p->green & 0b01;

break;

case 2:

bit = p->blue & 0b01;

break;

}

return bit;

}

/\* End of auxiliar functions \*/

void copy\_file(String input\_filename, String output\_filename) {

FILE \*f, \*g;

if (open\_file(input\_filename, output\_filename, &f, &g)) {

int c;

while ((c = fgetc(f)) != EOF) fputc(c, g);

fclose(f);

fclose(g);

}

}

void cesar\_encrypt(String input\_filename, int key, String encrypted\_filename) {

FILE \*f, \*g;

if (open\_file(input\_filename, encrypted\_filename, &f, &g)) {

int c;

while ((c = fgetc(f)) != EOF) {

if (c >= 'A' && c <= 'Z') {

c = c + key;

if (c > 'Z') c = c - 26;

}

fputc(c, g);

}

fclose(f);

fclose(g);

}

}

void cesar\_decrypt(String encrypted\_filename, int key,

String decrypted\_filename) {

FILE \*f, \*g;

if (open\_file(encrypted\_filename, decrypted\_filename, &f, &g)) {

int c;

while ((c = fgetc(f)) != EOF) {

if (c >= 'A' && c <= 'Z') {

c = c - key;

if (c < 'A') c = c + 26;

}

fputc(c, g);

}

fclose(f);

fclose(g);

}

}

void pi\_encrypt(String input\_filename, String pi\_filename,

String encrypted\_filename) {

FILE \*f, \*g, \*p;

if (open\_file(input\_filename, encrypted\_filename, &f, &g) &&

((p = fopen(pi\_filename, READ\_B)) != NULL)) {

int c;

char key;

fgetc(p); // dont care about '3'

fgetc(p); // dont care about '.'

while ((c = fgetc(f)) != EOF) {

if (c >= 'A' && c <= 'Z') {

key = getKey(p);

c = c + key;

if (c > 'Z') c = c - 26;

}

fputc(c, g);

}

fclose(f);

fclose(g);

fclose(p);

} else

error("File not found or invalid file '%s'", pi\_filename);

}

void pi\_decrypt(String encrypted\_filename, String pi\_filename,

String decrypted\_filename) {

FILE \*f, \*g, \*p;

if (open\_file(encrypted\_filename, decrypted\_filename, &f, &g) &&

((p = fopen(pi\_filename, READ\_B)) != NULL)) {

int c;

char key;

fgetc(p); // dont care about '3'

fgetc(p); // dont care about '.'

while ((c = fgetc(f)) != EOF) {

if (c >= 'A' && c <= 'Z') {

key = getKey(p);

c = c - key;

if (c < 'A') c = c + 26;

}

fputc(c, g);

}

fclose(f);

fclose(g);

fclose(p);

} else

error("File not found or invalid file '%s'", pi\_filename);

}

void pack\_encrypt(String input\_filename, String encrypted\_filename) {

FILE \*f, \*g;

if (open\_file(input\_filename, encrypted\_filename, &f, &g)) {

Byte final = 0, b;

int c, finished = 0, countBits = 0;

while (!finished) {

if ((countBits % 7) == 0) {

// get the next char to encrypt

b = (c = fgetc(f));

// trash the most significant bit out (because we dont care

// about initial bit = 0)

b = shiftl(b, 0);

}

// dont write on the output file in the first loop

if ((countBits % 8) == 0 && countBits != 0) fputc(final, g);

if (c == EOF)

finished = 1;

else {

final = shiftl(final, msb(b));

// trash the msb out, once it was already used

b = shiftl(b, 0);

countBits++;

}

}

if ((countBits % 8) != 0) {

// put as many zeros as the remaining to be a multiple of 8

while ((countBits % 8) > 0) {

final = shiftl(final, 0);

b = shiftl(b, 0);

countBits++;

}

// write the last char on the output file

fputc(final, g);

}

fclose(f);

fclose(g);

}

}

void pack\_decrypt(String encrypted\_filename, String decrypted\_filename) {

FILE \*f, \*g;

if (open\_file(encrypted\_filename, decrypted\_filename, &f, &g)) {

char c = fgetc(f);

int finish = 0, bitsFromInput = 0;

Byte b = c, word = 0;

while (!finish) {

if ((bitsFromInput % 7) == 0) {

// write the decrypted char

if (bitsFromInput != 0) fputc(word, g);

word = 0;

word = shiftl(word, 0);

}

if ((bitsFromInput % 8) == 0 && bitsFromInput != 0) {

// get the new char to decrypt

b = (c = fgetc(f));

// In a rare case 0b11111111 can be read as a byte (which represents

// EOF), but the file is not over yet. If we check that there are two

// EOFs following each other, we are sure that the file is over.

if (c == EOF && fpeek(f) == EOF) finish = 1;

}

word = shiftl(word, msb(b));

b = shiftl(b, 0);

bitsFromInput++;

}

fclose(f);

fclose(g);

}

}

/\* STEGANOGRAPHY \*/

void dots\_hide(String input\_filename, String message\_filename,

String disguised\_filename) {

FILE \*f, \*g, \*p;

if (open\_file(input\_filename, disguised\_filename, &f, &g) &&

((p = fopen(message\_filename, READ\_B)) != NULL)) {

int haveMessage = 1, nBits = 8, c, i;

Byte b;

// Message's loop

while (haveMessage) {

// encrypt the end of the message

if ((c = fgetc(p)) == EOF) {

haveMessage = 0;

c = 0;

}

b = c;

nBits = 8;

// Container's loop

while (nBits > 0 && (i = fgetc(f)) != EOF) {

fputc(i, g);

// checks if there is a dot followed by a space

if (i == '.' && fpeek(f) == ' ') {

if (msb(b)) fputc(' ', g);

fputc(' ', g);

nBits--;

b = shiftl(b, 0);

// ignores all the spaces that follows the space that already have

// been read

while (fpeek(f) == ' ') fgetc(f);

}

}

// we check the case that when we need to put the zero-byte, we can have

// not space for the 8 zero bits (case boolean = 0)

if ((i == EOF && haveMessage) || (!haveMessage && nBits > 0))

error("Message does not fit in the container file", input\_filename);

}

// copy the remainder of the container after the message have been encrypted

while ((i = fgetc(f)) != EOF) fputc(i, g);

fclose(f);

fclose(g);

fclose(p);

} else

error("File not found or invalid file '%s'", message\_filename);

}

void dots\_reveal(String disguised\_filename, String decoded\_filename) {

FILE \*f, \*g;

if (open\_file(disguised\_filename, decoded\_filename, &f, &g)) {

int c, counter = 0, messageFound = 0;

Byte b = 0;

char z = ' ';

while ((c = fgetc(f)) != EOF && !messageFound) {

// checks if there is a dot followed by a space

if (c == '.' && fpeek(f) == ' ') {

fgetc(f);

// check if there is two spaces instead of one

if (fpeek(f) == ' ') {

b = shiftl(b, 1);

} else {

b = shiftl(b, 0);

}

counter++;

}

if (counter == 8) {

counter = 0;

z = b;

// if z is not a zero-byte we need to write it on the output file

if (z)

fputc(z, g);

else {

messageFound = 1;

}

}

}

fclose(f);

fclose(g);

}

}

Int2 crude\_hide(Image img, Int2 n, String message\_filename, Image result) {

FILE \*f;

if ((f = fopen(message\_filename, READ\_B)) != NULL) {

int c;

Int2 i;

image\_copy(img, n, result);

for (i.y = 0; i.y < n.y; i.y++) {

for (i.x = 0; i.x < n.x; i.x++) {

if ((c = fgetc(f)) != EOF)

result[i.x][i.y].green = c;

else {

// encrypt the zero-byte that marks the end of the message

result[i.x][i.y].green = 0;

fclose(f);

return n;

}

}

}

error("Message does not fit in the container file", message\_filename);

fclose(f);

} else

error("File not found or invalid file '%s'", message\_filename);

return n;

}

void crude\_reveal(Image img, Int2 n, String decoded\_filename) {

FILE \*f;

if ((f = fopen(decoded\_filename, WRITE\_B)) != NULL) {

int finish = 0;

Int2 i;

for (i.y = 0; i.y < n.y && !finish; i.y++) {

for (i.x = 0; i.x < n.x && !finish; i.x++) {

if (img[i.x][i.y].green != 0)

putc(img[i.x][i.y].green, f);

else

finish = 1;

}

}

fclose(f);

} else

error("File not found or invalid file '%s'", decoded\_filename);

}

Int2 image\_hide(Image img, Int2 n, String message\_filename, Image result) {

FILE \*f;

if ((f = fopen(message\_filename, READ\_B)) != NULL) {

image\_copy(img, n, result);

int counter = 0, color = 0, countZeros = 0, c = 0, bit = 0;

Int2 i;

Byte b = 0;

for (i.y = 0; i.y < n.y && countZeros < 8; i.y++) {

// our loop accepts i.x = limit but just in case that the colors of the

// pixel are not already with encryptation. in this case, we guarantee

// that inside of the loop i.x will be decreased and that we will never go

// to limit position (512 in the example case)

for (i.x = 0;

(i.x < n.x || (i.x == n.x && (color % 3) != 0)) && countZeros < 8;

i.x++) {

// in order to continue on the same pixel as before, when the colors of

// the pixel are not all with encryptation

if ((color % 3) != 0) i.x--;

if ((counter % 8) == 0) {

b = (c = fgetc(f));

}

if (c == EOF) {

b = 0;

countZeros++;

}

bit = msb(b);

b = shiftl(b, 0);

Pixel \*p = &result[i.x][i.y];

makeColorMask(p, bit, color);

color++;

counter++;

}

}

if (fpeek(f) != EOF || countZeros != 8)

error("Message does not fit in the container file", message\_filename);

fclose(f);

} else

error("File not found or invalid file '%s'", message\_filename);

return n;

}

void image\_reveal(Image img, Int2 n, String decoded\_filename) {

FILE \*f;

if ((f = fopen(decoded\_filename, WRITE\_B)) != NULL) {

int counter = 0, color = 0, c = 0, finish = 0;

Int2 i;

Byte b = 0;

for (i.y = 0; i.y < n.y && !finish; i.y++) {

// our loop accepts i.x = limit but just in case that the colors of the

// pixel are not already with encryptation. in this case, we guarantee

// that inside of the loop i.x will be decreased and that we will never go

// to limit position (512 in the example case)

for (i.x = 0; (i.x < n.x || (i.x == n.x && (color % 3) != 0)) && !finish;

i.x++) {

// in order to continue on the same pixel as before, when the colors of

// the pixel are not all without encryptation

if ((color % 3) != 0) i.x--;

if (counter == 8) {

if (b == 0)

finish = 1;

else {

fputc(b, f);

b = 0;

counter = 0;

}

}

Pixel \*p = &img[i.x][i.y];

b = shiftl(b, getMaskedBit(p, color));

color++;

counter++;

}

}

fclose(f);

} else

error("File not found or invalid file '%s'", decoded\_filename);

}