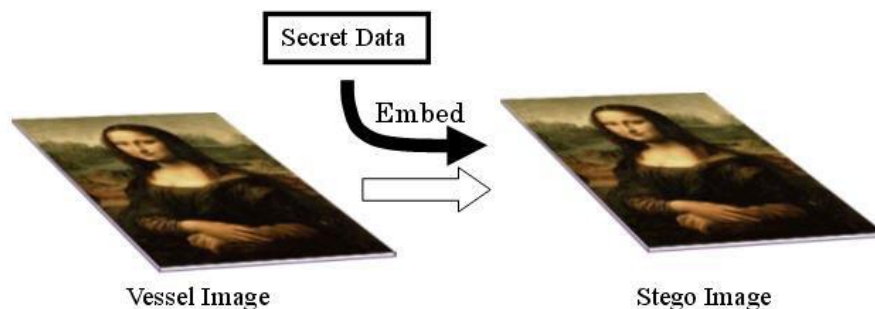


موعد تحویل: شنبه ۲۳ بهمن ۱۳۹۵

پنهان‌نگاری^۱ در عکس

در این تمرین قصد داریم با خواندن تصاویر از فایل و انجام عملیات روی آن با داده ساختارهای string و vector آشنا شویم و در عین حال از تکنیک‌های برنامه‌سازی بالا به پایین برای پیاده‌سازی استفاده کنیم.

با توجه به اینکه این تمرین، نخستین تمرین کامپیوتری شماست، فرآیند طراحی و پیاده‌سازی را به صورت مرحله به مرحله توضیح خواهیم داد و قویاً توصیه می‌کنیم که شما نیز مطابق با همین مراحل، برنامه خود را توسعه دهید.



پنهان‌نگاری علم برقراری ارتباط پنهانی است و هدف آن پنهان کردن ارتباط به وسیله‌ی قراردادن پیام در یک رسانه‌ی پوششی است. به گونه‌ای که کمترین تغییر قابل کشف را در آن ایجاد کند و هرگونه نشانه‌ای از وجود پیام، مخفی شود. تفاوت اصلی رمزنگاری با پنهان‌نگاری نیز در همین است. در رمزنگاری هدف مخفی کردن محتویات پیام است و نه وجود پیام.

به عنوان مثال اگر شخصی به متن رمزنگاری شده‌ای دسترسی پیدا کند، به هر حال متوجه می‌شود که این متن حاوی پیام رمزی می‌باشد. اما در پنهان‌نگاری شخص سوم ابداً از وجود پیام مخفی در متن اطلاعی حاصل نمی‌کند. در موارد حساس ابتدا متن را رمزنگاری کرده، آنگاه آن را در متن دیگری پنهان‌نگاری می‌کنند.

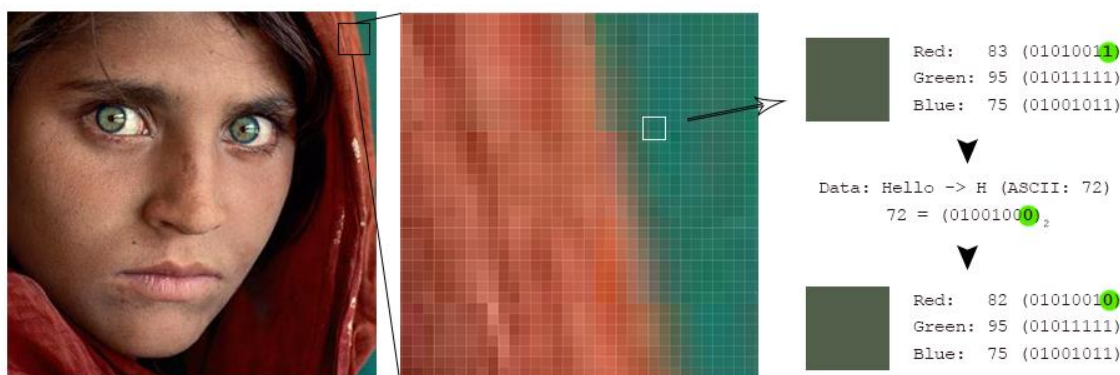
در این پروژه شما پنهان‌نگاری در عکس را انجام خواهید داد. یعنی متنی پیامی را که در اختیار دارید به نحوی داخل عکس پنهان کنید که کمترین تغییر قابل کشف در عکس ایجاد شود. هدف شما در این تمرین پنهان کردن یک متن در تصویر و استخراج متن پنهان شده در همان تصویر است. تصاویر با فرمت BMP که در ادامه با آن آشنا خواهید شد در اختیار برنامه شما قرار داده می‌شود.

¹ Steganography

شمای کلی

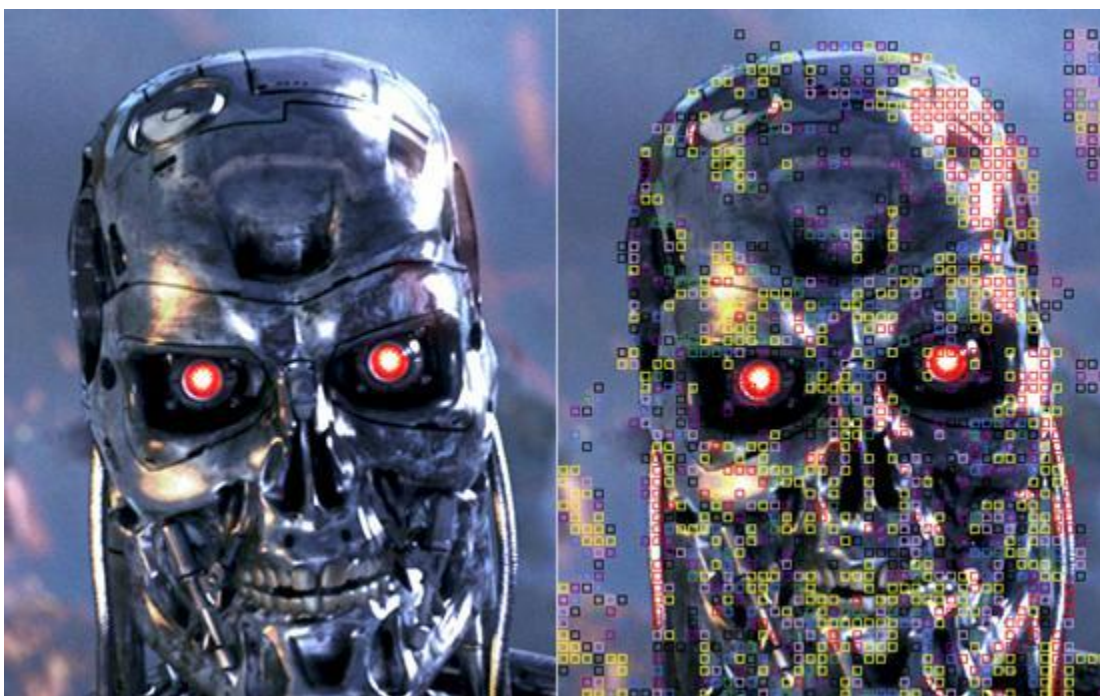
به صورت کلی در سیستم‌های کامپیوتری هر تصویر از تعدادی پیکسل تشکیل شده است. هر پیکسل نشان دهنده‌ی یک نقطه‌ی رنگی در تصویر است که به وسیله‌ی سه عدد در بازه‌ی ۰ تا ۲۵۵ که متناظر با رنگ‌های قرمز و سبز و آبی است مدل‌سازی می‌شود (این سیستم نمایش رنگ RGB نام دارد). از آنجایی که قدرت بینایی چشم ما محدود است و فرق بین رنگ‌های نزدیک به هم را نمی‌تواند تشخیص دهد، تغییر جزئی این اعداد در یک پیکسل از تصویر محسوس نخواهد بود. بنابراین می‌توان با تغییر جزئی رنگ پیکسل‌ها، داده‌هایی را در یک تصویر ذخیره سازی نمود. در این تمرین شما داده‌های خود را در کم‌ارزش‌ترین بیت رنگ‌های یک پیکسل ذخیره خواهید کرد. برای اینکار باید باینری هر کدام از این سه عدد را برای هر پیکسل در نظر گرفت و بیت کم‌ارزش (LSB) آن را تغییر داد.

به عنوان نمونه در عکس زیر یک بیت از حرف H پیغام Hello داخل یک بیت از رنگ قرمز یکی از پیکسل‌های تصویر ذخیره سازی شده است. همانطور که مشاهده می‌کنید، تفاوت رنگ قبل و بعد از پنهان‌نگاری برای چشم انسان نامحسوس است.



برای جلوگیری از کشف شدن پیام توسط افراد دیگر می‌توانیم به طور تصادفی تعداد مناسبی پیکسل را انتخاب کنیم و یکی از سه رنگ اصلی آن را که خود می‌تواند به صورت تصادفی تعیین شود را تغییر دهیم. این انتخاب‌های تصادفی توسط تابع `rand()` می‌توانند انجام شوند که با دریافت seed یکسان، دنباله‌ای از اعداد یکسان تولید می‌کند.

در صورتی که پیکسلی که به صورت تصادفی انتخاب می‌شود در ناحیه‌ای باشد که تنوع رنگی کم است و با پیکسل‌های اطرافش هم‌رنگ باشد، بعد از تغییر راحت‌تر قابل تشخیص است. بنابراین بهتر است که پیکسل‌های تصادفی را در محلهایی از عکس انتخاب کنیم که بیشترین پراکندگی رنگی را دارند. برای این کار تصویر را به مربع‌های ۸ پیکسل در ۸ پیکسل تقسیم می‌کنیم و پیکسل تصادفی را از مربع‌هایی انتخاب می‌کنیم که پراکندگی (تنوع) رنگی بیشتری دارند. به عنوان مثال در عکس زیر مربع‌هایی که پراکندگی رنگی بیشتری را نسبت به بقیه دارند مشخص شده‌اند.



توجه داشته باشید که از آنجایی که عکس‌های سیاه و سفید از سیستم RGB پشتیبانی نمی‌کنند و ساختار آنها متفاوت است، روش گفته شده تنها برای عکس‌های رنگی قابل استفاده است. در این تمرین نیز تنها عکس‌های رنگی به برنامه‌ی شما داده خواهد شد.

الگوریتم و طراحی

طراحی این برنامه به شرح زیر است:

پنهان نگاری:

- 1 - خواندن تصویر
- 2 - تقسیم تصویر به مربع‌های ۸ در ۸ و محاسبه‌ی پراکندگی هر یک از رنگ‌های RGB در مربع‌ها
- 3 - دریافت پیامی از کاربر که باید در عکس مخفی شود
- 4 - دریافت کلید از کاربر و تولید رشته‌ی تصادفی با توجه به آن
- 5 - برای هر مربع ۸ در ۸ (از پراکندگی رنگی بیشتر به کمتر):
 - a. انتخاب یک پیکسل به وسیله‌ی رشته‌ی تصادفی تولید شده
 - b. تغییر LSB پیکسل انتخاب شده با توجه به رشته‌ی باینری
 - c. تکرار مرحله ۵ تا زمانی که تمام پیغام در عکس ذخیره شود (متن شامل کاراکتر newline نیز می‌باشد)
- 6 - ذخیره‌سازی تصویر نهایی در دیسک

استخراج پیغام مخفی شده:

- 1 - خواندن تصویر
 - 2 - تقسیم تصویر به مربع‌های ۸ در ۸ و محاسبه‌ی پراکندگی هر یک از رنگ‌های RGB در مربع‌ها
 - 3 - دریافت کلید از کاربر و تولید دنباله‌ی تصادفی با توجه به آن
 - 4 - برای هر مربع ۸ در ۸ (از پراکندگی رنگی بیشتر به کمتر):
 - a. انتخاب یک پیکسل به وسیله‌ی رشته‌ی تصادفی تولید شده
 - b. خواندن LSB پیکسل انتخاب شده و اضافه کردن آن به یک رشته
 - c. تکرار مرحله ۴ تا زمانی که کاراکتر newline خوانده شود
 - 5 - ذخیره‌سازی پیغام در دیسک
- با توجه به اینکه فرمت عکس ورودی BMP خواهد بود، قبل از توضیح بیشتر درباره‌ی هر کدام از مراحل الگوریتم، با این فرمت ذخیره‌سازی تصاویر بیشتر آشنا می‌شویم.


فایل BMP

ذخیره سازی عکس ها با فرمت bitmap ساده ترین نوع ذخیره ی عکس ها بر روی سیستم های کامپیوتری است. در فایل های bitmap داده ها بدون فشرده سازی به صورت بایت بایت و پشت سر هم در فایل با پسوند bmp نوشته می شوند. اطلاعات مهم در این فایل ها به سه قسمت تقسیم می شوند :

۱. هدر فایل: شامل اطلاعاتی مانند فرمت فایل، حجم آن و اطلاعات کلی دیگری درباره ی فایل می شود. اندازه این قسمت ۱۴ بایت است. نحوه ی تقسیم این ۱۴ بایت در جدول زیر آمده است.

سایز(بایت)	داده	آفست شروع از اول فایل
۲	نوع فایل	۰
۴	حجم فایل	۲
۲	(رزرو شده)	۶
۲	(رزرو شده)	۸
۴	آفست شروع داده های پیکسل ها	۱۰

همانطور که در جدول نشان داده شده است، بایتهای ۰ و ۱ (دو بایت اول فایل) نوع فایل را مشخص میکنند و بایتهای ۲ تا ۵ مشخص کننده ی حجم آن هستند.

۲. هدر تصویر: همانطور که از اسم این قسمت مشخص است  اطلاعات کلی عکس، نظیر طول و عرض عکس و یا نحوه ی فشرده سازی، در این قسمت قرار دارد. اندازه ی این قسمت ۴۰ بایت است. طول تصویر در بایتهای ۴ تا ۷ و عرض آن در بایتهای ۸ تا ۱۱ از ۴۰ بایت مربوط به هدر تصویر قرار دارند.

۳. داده ی پیکسل: در این قسمت به ازای هر پیکسل از عکس، اطلاعات آن نگهداری می شود. رنگ هر پیکسل با سه عدد بین ۰ تا ۲۵۵ متناظر با سه رنگ قرمز، سبز و آبی نگهداری می شود.

نحوه ی ذخیره سازی اطلاعات در این بخش به این ترتیب است که ابتدا اطلاعات پیکسل های پایین ترین سطر نگهداری می شوند و سپس اطلاعات سطرها بالاتر تا سطر صفر. در هر سطر اطلاعات پیکسل ها از سمت چپ ترین ستون به سمت راست نگهداری می شوند. برای هر پیکسل سه بایت ذخیره می شود که به ترتیب وزن رنگهای آبی، سبز و قرمز هستند. همچنین، تعداد بایتهای هر سطر باید مضرب ۴ باشد، در غیر این صورت به تعداد بایت باقیمانده (تا تعداد بایتهای سطر مضرب چهار شود) بایت صفر در انتهای سطر قرار می گیرد. (به این نکته هم در خواندن و هم در نوشتن فایل توجه کنید).

در خواندن فایل BMP به نکات زیر توجه کنید.

- برای باز کردن یک فایل از قطعه کدی شبیه کد زیر استفاده کنید. بعد از باز کردن فایل می توانید از image_file همانند cin استفاده کنید.

```

#include <fstream>
using namespace std;

void my_function() {
    ifstream img_file(file_name.c_str());
    if(!img_file) {
        cerr << "can't open file [" << file_name << "]\n";
        return false;
    }
    //...
    // your code here
}

```

- خواندن یک بایت: از تابع `image_file.get()` استفاده کنید. (چرا؟)
- اطلاعات پیکسل‌ها در فایل BMP به صورت برعکس نوشته می‌شود. یعنی ابتدا بایت B، سپس بایت G و سپس بایت R نوشته می‌شود.
- در هدر فایل بایت کم ارزش‌تر اول نوشته می‌شود و بایت‌های پر ارزش پس از آن می‌آیند. به مثال زیر که نمای HEX بایت‌های ابتدایی یک عکس BMP با ابعاد 2560x1440 را نشان می‌دهد، دقت کنید.

نوع فایل
حجم فایل 0x00a8c08a
آفست داده‌های پیکسل‌ها

عرض فایل
طول فایل
0x0000008a = 138

0x000000a00 = 2560
0x000005a0 = 1440

در زیر توضیحات مربوط به هر قسمت از الگوریتم آمده است:

خواندن تصاویر Bitmap

برنامه شما باید قادر باشد هر تصویر bitmap را خوانده و آن را به طور کامل تحلیل کند و ویژگی های تصویر را استخراج کند. شما می توانید برای نگه داری عکس در برنامه خود از ساختار struct بهره ببرید.

قطعه بندی تصویر به مربع های ۸ در ۸

لازم است در این مرحله تصویر خود را به مربع های 8×8 تقسیم کنید به طوری که مربع ها حتی در یک پیکسل هم همپوشانی نداشته باشند (چرا؟). در صورتی که طول یا عرض عکس مضرب ۸ نبود می توانید حاشیه های عکس را در نظر نگیرید. به عنوان مثال نحوه ی تقسیم بندی عکسی به طول ۵۸۸ پیکسل در زیر آمده است:



در یک مربع 8×8 پراکندگی یا واریانس هر یک از سه رنگ قرمز (R)، سبز (G) و آبی (B) را به کمک واریانس محاسبه می کنیم به این صورت که ابتدا میانگین آن رنگ را در مربع مورد نظر می یابیم و سپس حاصل جمع مجذور اختلاف همان رنگ هر پیکسل مربع را از میانگین حساب می کنیم. از آنجایی که ممکن است LSB هر رنگ بعد از پنهان نگاری در پیکسل ها تغییر کند، واریانس اعداد را بدون در نظر گرفتن LSB حساب کنید.

می توانید برای نگه داری این اطلاعات از struct استفاده کنید که مختصات شروع مربع، رنگ مورد نظر و حاصل واریانس آن مربع را به ازای آن رنگ ذخیره کند (در نهایت لیستی از این داده ساختار به طول ۳ برابر تعداد پیکسل های عکس خواهیم داشت).

حال این اطلاعات را درون لیستی که بر حسب واریانس ها به صورت نزولی مرتب شده است، نگه داری کنید.

خواندن فایل پیغام

برنامه ی شما پیغامی که قرار است در عکس ذخیره شود را با چاپ پیام مناسب از کاربر دریافت می کند. دقت داشته باشید که طول این پیغام باید محدود باشد. اگر طول متن بیشتر از مقدار مجاز باشد (و نتوان آن را در عکس پنهان کرد)، برنامه شما باید بتواند تشخیص دهد و اعلام کند که پیام قابل پنهان سازی در این تصویر نیست. برای اینکار فرمول محاسبه ی حداکثر ظرفیت ذخیره سازی عکس (بر حسب بیت) را به دست بیاورید و از آن استفاده کنید.

تولید دنباله عدد تصادفی به وسیله کلید

برای پنهان‌نگاری هر عکس یک عدد ۴ تا ۶ رقمی از کاربر به عنوان کلید دریافت می‌شود. از این کلید برای تولید دنباله‌ی عدد تصادفی استفاده می‌شود. بازیابی پیغام پنهان شده در عکس تنها به وسیله‌ی این کلید امکان‌پذیر است. شما باید این عدد را در هنگام اجرای برنامه توسط یک پیغام مناسب از کاربر دریافت کنید. در صورتی که عدد وارد شده بین ۴ تا ۶ رقم نبود، اجرای برنامه با نمایش پیغام خطای مناسب متوقف خواهد شد.

اعداد تصادفی‌ای که به روش‌های معمول در کامپیوتر تولید می‌شوند کاملاً تصادفی نیستند. به این اعداد، اعداد شبه تصادفی می‌گویند و توسط مولد اعداد شبه تصادفی^۲ تولید می‌شوند. یک مولد اعداد شبه تصادفی یک عدد صحیح به نام seed دریافت کرده و یک دنباله عدد تصادفی بر حسب آن تولید می‌کند. مولد اعداد شبه تصادفی به ازای seed یکسان، دنباله‌ی یکسانی تولید می‌کند. بنابراین هنگام بازیابی پیغام دچار مشکل نخواهیم شد.

شما در این پروژه از مولد اعداد تصادفی که در کتابخانه‌ی استاندارد C++ (هدر `stdlib`) قرار دارد استفاده می‌کنید. در این کتابخانه به وسیله‌ی تابع `srand` مقدار seed (که همان کلید است) را مشخص کرده و به وسیله‌ی تابع `rand` به دنباله‌ی تولید شده دسترسی خواهید داشت. برای آشنایی با نحوه‌ی کار با این مولد، به اینترنت مراجعه کنید.

پنهان کردن فایل در عکس

همان طور که در مرحله‌ی قطعه‌بندی توضیح داده شد. بعد از قطعه‌بندی تصویر به مربع‌ها و محاسبه‌ی پراکندگی آن‌ها در هر یک از سه رنگ، آن‌ها را به صورت مرتب شده نگه‌داری می‌کنیم. در هر مرحله، یک مربع انتخاب می‌کنیم، سپس یک عدد تصادفی (با توجه به قسمت قبل) تولید می‌کنیم. این عدد تصادفی مشخص می‌کند که بیت مورد نظر در کدام یک از پیکسل‌های مربع ۸ × ۸ انتخاب شده ذخیره شود. در نهایت یک بیت از پیغام را در پیکسل مشخص شده توسط عدد تصادفی ذخیره می‌کنیم. دقت کنید که این عدد باید در LSB یکی از رنگ‌های این پیکسل (رنگی که واریانس بر حسب آن محاسبه شده) ذخیره شود.

راهنمایی: تغییر یک بیت از یک عدد به مقدار دلخواه را می‌توان به وسیله‌ی عملگرهای bitwise مانند `|` و `&` انجام داد.

دقت کنید که هنگام پنهان‌نگاری متن در عکس کاراکتر `'\n'` یا همان `newline` را نیز حتماً در عکس پنهان کنید. از این کاراکتر هنگام بازیابی متن استفاده خواهیم کرد.

ذخیره کردن تصویر

برای ذخیره تصویر با فرمت `bmp` باید از همان قوانین اشاره شده در ابتدای پروژه استفاده کنید. در صورتی که اطلاعات کلی فایل (ابعاد، سایز) تغییر کرده است لازم است که هدر را خودتان بسازید ولی در غیر این صورت می‌توانید از همان هدر تصویر اولیه استفاده کرده و فقط اطلاعات پیکسل‌ها را تغییر دهید.

راهنمایی: برای تبدیل یک عدد به ۴ بایت مجزا کافی است تا مقدار آن را در مبنای ۲۵۶ محاسبه کنید. (تقسیم و محاسبه باقی‌مانده به صورت متوالی)

¹ Pseudo-Random Number Generator (PRNG)

ورودی

تصویر اصلی در کنار فایل کامپایل شده (اجرایی) شما قرار خواهد گرفت. در ابتدای برنامه به ورودی استاندارد (stdin) یکی از دو دستور encrypt یا decrypt داده می‌شود. سپس نام تصویر اصلی و کلید ۴ تا ۶ رقمی داده خواهد شد. در نهایت در صورتی که دستور وارد شده encrypt باشد، متن پیغام در غالب یک خط به برنامه داده می‌شود.

نمونه‌ی ورودی:

```
encrypt
image.bmp
8831
You are the SEMICOLON to my STATEMENT
```

```
decrypt
coded-image.bmp
62425
```

خروجی

در صورتی که دستور مورد نظر encrypt باشد، عکس پنهان‌نگاری شده در فایل به نام output.png در کنار فایل اجرایی برنامه ذخیره خواهد شد.

در صورتی که دستور مورد نظر decrypt باشد، پیغام استخراج شده در خروجی استاندارد قرار خواهد گرفت.

دقت داشته باشید که برنامه‌ی شما باید نسبت به ورودی‌های نامعتبر پیغام خطای مناسب چاپ کند.

دقت کنید

- در صورتی که نتوانستید کل پروژه را پیاده سازی کنید، سعی کنید به ترتیب قسمت‌هایی را که می‌توانید پیاده سازی کنید تا قسمتی از نمره را کسب کنید.
- استفاده از vector های موازی حرام است!
- شما مجاز به استفاده از آرایه و پوینتر نیستید. استفاده از struct و typedef بلامانع است.
- برنامه‌ی شما باید در سیستم‌عامل لینوکس نوشته شده و با مترجم g++ کامپایل شود.
- برنامه‌ی شما باید بر اساس استاندارد c++98 باشد (-std=c++98)
- به فرمت و نام فایل‌های خود دقت کنید. در صورتی که هر یک از موارد گفته شده رعایت نشود، نمره‌ی صفر برای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، مطابق سیاست‌های درس با آن برخورد خواهد شد.