



مقدمه

هدف از این تمرین آشنایی با سربارگذاری عملگرها و استفاده از این قابلیت در محاسبات ماتریسی در پردازش تصویر است. همانطور که اطلاع دارید، ماتریس، آرایشی منظم از عناصر حقیقی است که در تعدادی سطر کنار هم قرار گرفته‌اند. هریک از این عناصر واحد، درایه نام دارد. در این تمرین برای یک ماتریس و سطرها، مجموعه‌ای از فرایندهای ریاضی را تعریف کرده و اپراتورهای متناظر با هر یک را سربارگذاری می‌کنیم. سپس به کمک این اپراتورها یک عملیات پردازش تصویر ساده را انجام خواهیم داد. توجه کنید که در این تمرین نیازی به نوشتن تابع main نیست و فایل‌های شما با استفاده از تابع آماده‌شده توسط دستیاران آموزشی تست خواهد شد.

کلاس سطر

ماتریس از تعدادی سطر هم اندازه که زیر هم قرار گرفته‌اند تشکیل شده است. جدول زیر فرایندهای مرتبط با هر سطر و عملگر متناظر آنها را مشخص می‌کنند. توجه کنید که متغیر با نام Row نمایانگر یک شیء از نوع سطر و متغیر با نام Matrix نمایانگر یک شیء از نوع ماتریس است؛ همچنین به جای نام متغیرها در جدول زیر تایپ آن‌ها قرار داده شده است.

توضیحات	Method / Operator
سطری به طول مشخص شده می‌سازد و تمام درایه‌های آن را برابر با صفر قرار می‌دهد.	Row(int)
سطری خالی می‌سازد و با بردار مشخص شده مقداردهی می‌کند.	Row(vector<double>)
سطر طرف چپ را برابر با سطر طرف راست قرار می‌دهد. ^۱	Row1 = Row2
سطر را برابر با یک بردار از اعداد قرار می‌دهد. ^۱	Row = vector<double>
سطری جدید می‌سازد که درایه‌های آن حاصل جمع درایه‌های متناظر در این دو سطر است. ^۱	Row1 + Row2
درایه‌های سطر سمت راست را به درایه‌های سطر سمت چپ اضافه می‌کند. ^۱	Row1 += Row2
ضرب ماتریس $1 \times n$ در ماتریس $n \times m$ سطر را در ماتریس ضرب کرده و مقدار حاصل را در سطر جدیدی بازمی‌گرداند. ^۲	Row * Matrix
یک عدد حقیقی را در درایه‌های سطر ضرب کرده و مقدار حاصل را در سطر جدیدی بازمی‌گرداند.	double * Row
رفرنس به درایه n ام سطر را بازمی‌گرداند. ^۳	Row[int]
مشابه عملگر قبل با این تفاوت که روی اشیای ثابت فراخوانی می‌شود. ^۳	Row[int] (Row is a constant object)
تساوی دو سطر را بررسی می‌کند.	Row1 == Row2
تعداد درایه‌های سطر را بازمی‌گرداند.	int size(void)

^۱ طول دو سطر باید یکسان باشد. در غیر این صورت یک استثنا با نام RowLengthException، throw کنید.

^۲ طول سطر باید با تعداد سطرها، ماتریس برابر باشد. در غیر این صورت یک استثنا با نام MatrixMultException، throw کنید.

^۳ در صورتیکه آرگومان ورودی بزرگتر از اندازه‌ی سطر یا عددی منفی باشد یک استثنا با نام IndexOutOfBoundsException، throw کنید.

کلاس ماتریس

مشابه بخش قبل، جدول زیر فرایندهای مرتبط با هر ماتریس و عملگرهای متناظر آنها را مشخص می‌کنند.

Method / Operator	توضیحات
Matrix()	ماتریسی خالی می‌سازد.
Matrix(int i, int c)	ماتریسی شامل ۲ سطر می‌سازد که هر سطر از c درایه صفر تشکیل شده‌است.
Matrix(vector<Row>)	ماتریسی خالی می‌سازد و با برداری از سطرها مقداردهی می‌کند.
Matrix(double)	ماتریسی شامل یک سطر می‌سازد و تنها درایه آن سطر را با عدد مشخص شده مقداردهی می‌کند.
Matrix1 = Matrix2	ماتریس طرف چپ را برابر با ماتریس طرف راست قرار می‌دهد.
Matrix = Row	ماتریس را برابر با یک سطر قرار می‌دهد.
Matrix1 + Matrix2	درایه‌های متناظر دو ماتریس را جمع کرده و مقدار حاصل را در ماتریسی جدیدی بازمی‌گرداند. ^۴
Matrix1 * Matrix2	ماتریس سمت چپ را در ماتریس سمت راست ضرب کرده و مقدار حاصل را در ماتریسی جدیدی بازمی‌گرداند. توجه کنید که تعداد سطرها در ماتریس سمت راست باید با تعداد ستون‌های ماتریس سمت چپ برابر باشد. ^۲
Matrix1 *= Matrix2	ماتریس سمت راست را در ماتریس سمت چپ ضرب کرده و مقدار حاصل را در ماتریس سمت چپ قرار می‌دهد. ^۲
int height(void)	تعداد سطرها در ماتریس را بازمی‌گرداند.
int width(void)	تعداد درایه‌های یک سطر را باز می‌گرداند.
void insert(Row, int)	یک سطر را در اندیس مشخص شده در ماتریس اضافه می‌کند.

^۴ طول دو سطر باید یکسان باشد. در غیر این صورت یک استثنا با نام MatrixLengthException، throw کنید.

^۲ طول سطر باید با تعداد سطرها در ماتریس برابر باشد. در غیر این صورت یک استثنا با نام MatrixMultException، throw کنید.

عملگرهای ورودی و خروجی

Method / Operator	توضیحات
1) cin >> Row 2) cin >> Matrix 3) cout << Row 4) cout << Matrix	برای خواندن از ورودی استاندارد و درج در خروجی استاندارد استفاده می‌شوند.

(۱) اطلاعات هر سطر در یک خط در ورودی استاندارد وارد می‌شود. درایه‌های یک سطر توسط کاراکتر t\ از هم جدا می‌شوند.

(۲) هر ماتریس با یک براکت شروع شده، در خط بعدی سطر اول، پس از آن سطر دوم و ... به برنامه داده می‌شود؛ پس از آخرین سطر نیز یک براکت در خط بعدی وارد می‌شود. برای مثال:

```
[
1      2      3
1.3    7.8    9
12     100    1
]
```

(۳) سطر را با همان فرمت قسمت ۱ چاپ کنید. پس از چاپ کردن درایه‌ها، نباید کاراکتر n\ را در انتهای خط چاپ کنید!

(۴) ماتریس را با همان فرمت قسمت ۲ چاپ کنید. پس از چاپ کردن براکت انتهای ماتریس، نباید کاراکتر n\ را چاپ کنید!

کلاس تصویر

این کلاس نشان‌دهنده یک تصویر است. جدول زیر فرآیندهای مرتبط با هر تصویر و عملگرهای مرتبط را نشان می‌دهد:

Method / Operator	توضیحات
<code>Image()</code>	یک تصویر خالی ایجاد می‌کند.
<code>Image(string filePath)</code>	یک تصویر را از روی فایل مشخص شده ایجاد می‌کند.
<code>Image1 = Image2</code>	تصویر سمت چپ را برابر تصویر سمت راست قرار می‌دهد.
<code>Image1 = !Image2</code>	تصویر جدیدی که هر رنگ پیکسل آن برابر ۲۵۵ منهای مقدار نظیرش در این تصویر است باز می‌گرداند.
<code>Image * Matrix</code>	یک ماتریس را در یک تصویر Convolve می‌کند و در تصویری جدید بازمی‌گرداند. ^۵ (توضیحات بیشتر در ادامه)
<code>int height(void)</code>	تعداد سطرهای تصویر را برمی‌گرداند.
<code>int width(void)</code>	تعداد ستون‌های تصویر را برمی‌گرداند.
<code>void save(string filePath)</code>	تصویر را در آدرس مشخص شده ذخیره می‌کند.

^۵ ماتریس باید مربعی با اندازه‌ی فرد باشد. در غیر این صورت یک استثنا با نام `ConvolveException`، `throw` کنید.

به طور کلی در سیستم‌های کامپیوتری هر تصویر از تعدادی پیکسل تشکیل شده است. هر پیکسل نشان‌دهنده‌ی یک نقطه رنگی در تصویر است که به وسیله‌ی سه عدد در بازه‌ی ۰ تا ۲۵۵ که متناظر با رنگ‌های قرمز، سبز و آبی است مدل سازی می‌شود. این سیستم نمایش رنگ RGB نام دارد.

شما در این تمرین فقط با تصاویر bitmap کار خواهید کرد. برای سادگی کار با این تصاویر می‌توانید از کتابخانه‌ی زیر استفاده کنید:

<https://github.com/ArashPartow/bitmap>

عملیات Convolve کردن ماتریس در تصویر

یک ماتریس مربعی با اندازه‌ی فرد به این ترتیب در یک عکس `convolve` می‌شود که برای هر پیکسل `i, j` از تصویر عملیات زیر را تکرار می‌کنیم:





مرکز ماتریس (حتما درایه‌ی مرکز وجود دارد چرا که ماتریس مربعی با اندازه‌ی فرد است) را بر روی پیکسل `i, j` قرار می‌دهیم، سپس هر درایه از ماتریس را در پیکسل متناظرش ضرب می‌کنیم. حاصل جمع تمام این مقادیر برابر مقدار جدید پیکسل `i, j` خواهد بود.

برای انجام این عملیات در حالات مرزی روش‌های مختلفی وجود دارد. در این پروژه برای سادگی می‌توانید از انجام عملیات روی این نقاط صرف نظر کنید. یعنی اگر برای انجام عملیات روی پیکسل `n`ام به پیکسلی خارج از مرز تصویر نیاز داشتید، همان پیکسل `n`ام را در تصویر خروجی قرار دهید.

در مثال زیر ماتریس $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ در درایه‌ی ۲,۲ تصویر `convolve` شده است.

$$\begin{array}{|c|c|c|c|c|} \hline 35 & 40 & 41 & 45 & 50 \\ \hline 40 & 40 & 42 & 46 & 52 \\ \hline 42 & 46 & 50 & 55 & 55 \\ \hline 48 & 52 & 56 & 58 & 60 \\ \hline 56 & 60 & 65 & 70 & 75 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & 0 & 1 & 0 & \\ \hline & 0 & 0 & 0 & \\ \hline & 0 & 0 & 0 & \\ \hline & & & & \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & 42 & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array}$$

با استفاده از عملیات convolve می‌توان فیلترهای بسیاری بر روی عکس‌ها داشت. جدول زیر چند نمونه از این کار را نشان می‌دهد.

همانی (Identity)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
پیدا کردن مرزها (Laplacian edge detector)	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
محو کردن (Blur)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
تیز کردن (Sharpen)	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

نکات مهم

- در کنار فایل صورت پروژه یک فایل main.cpp و یک فایل lena.bmp¹ در اختیار شما قرار داده شده است که می‌توانید از آن‌ها استفاده کنید.
- طبیعتاً فایل تست دستیاران آموزشی بزرگتر از فایلی است که در اختیار شماست، بنابراین از صحت برنامه‌ی خود اطمینان داشته باشید.
- به نامگذاری فایل‌های خود دقت کنید. (با توجه به فایل main.cpp که در اختیار شما قرار داده شده است.)
- به نامگذاری Exception‌ها دقت کنید و در مواردی که استثناهای دیگری به نظرتان می‌رسد از نام‌های مناسب استفاده کنید.
- هشدارهای کامپایلر (warning) را جدی گرفته و سعی کنید آن‌ها را رفع کنید.
- شما می‌توانید علاوه بر توابع معرفی شده در جداول بالا، توابعی دلخواه جهت افزایش خوانایی و بهبود کدهای خود در نظر بگیرید. اما هنگام تحویل حضوری باید قادر به دفاع از تصمیمات خود باشید!
- به فرمت‌های مطرح شده در صورت تمرین دقت کنید و توابع و عملگرهای خود را دقیقاً مانند جداول بالا تعریف و پیاده‌سازی کنید. در صورت یکسان نبودن فرمت‌ها، تابع main نوشته شده برای تست کردن کد شما، با خطای زمان کامپایل روبرو می‌شود و بخشی از نمره را از دست خواهید داد.

¹<https://en.wikipedia.org/wiki/Lenna>

نحوه‌ی تحویل

فایل‌های مربوط به برنامه‌ی خود را در پوشه‌ای به نام A5-SID.zip را در سایت درس آپلود کنید. (SID پنج رقم آخر شماره‌ی دانشجویی شماست. به عنوان مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۶۱۲۳ است، نام فایل شما باید A5-96123.zip باشد.) فایل آپلودی شما باید شامل پوشه‌ی کامل پروژه باشد. تحویل این تمرین به صورت حضوری است و در هنگام تحویل باید به تمام قسمت‌های کد خود مسلط باشید.

دقت کنید

- پروژه‌ی شما باید حتماً **Makefile** داشته باشد. در غیر این صورت نمره این بخش را از دست خواهید داد.
- برنامه‌ی شما باید در سیستم عامل لینوکس نوشته و با کامپایلر g++ کامپایل شود.
- به فرمت و نام فایل‌های خود دقت کنید. در صورتی که هر یک از موارد گفته شده رعایت نشود، نمره‌ی **صفر** برای بخش اجرای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، با هر دو طرف طبق قوانین درس برخورد خواهد شد.
- مجدداً تاکید می‌کنیم که به نامگذاری‌ها و فرمت‌ها دقت کنید تا در موقع تحویل دچار مشکل نشوید.