

برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۶



مدرس: رامتین خسروی

طراحان: میثاق محقق، آوا میرمحمد مهدی، نسا عباسی،

شهنام فیضیان، مبینا مهرآذر، حسام رضانیان

مهلت تحویل: جمعه ۲۴ آذر ۱۴۰۲، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین مهارت بیشتر شما در برنامه‌نویسی شیء‌گرا با استفاده از مفاهیم وراثت و چندریختی است. انتظار می‌رود از تکنیک‌های برنامه‌نویسی که تاکنون در کلاس درس فرا گرفته‌اید یا در هنگام تحویل حضوری تمرین‌ها به شما تذکر داده شده‌است به طور کامل در این تمرین استفاده کنید. طراحی کلاس‌ها، نحوه ارث‌بری آن‌ها از یکدیگر و تعریف صحیح توابع مربوط به هر کدام از کلاس‌ها اهمیت بالایی دارد؛ به همین منظور پیشنهاد می‌شود قبل از پیاده‌سازی پروژه، ابتدا طراحی‌های مختلف را بررسی و سپس مناسب‌ترین طراحی را پیاده‌سازی کنید.

شرح تمرین

در این تمرین، به طراحی یک برنامه فیلترگذاری عکس می‌پردازیم. به این صورت که آدرس یک عکس را ورودی گرفته، فیلترهای خواسته‌شده را روی عکس مدنظر به ترتیب اعمال کرده و در نهایت خروجی را ذخیره می‌کنیم. در توضیح فیلترها از عکس زیر برای نشان دادن اثر فیلترها استفاده می‌کنیم:



انواع فیلترها

فیلترهای حاوی کرنل

کرنل فیلترها نوعی از فیلترها هستند که یک پیکسل و 8 خانه‌ی اطراف آن را در یک ماتریس به نام کرنل به صورت زیر ضرب می‌کنند و مقدار پیکسل را برورسانی می‌کند. مثلا در شکل زیر می‌خواهیم مقدار جدید پیکسل $[1, 1]$ را پس از اعمال فیلتر بدست آوریم:

3	4	7	
8	6	3	
4	1	2	

1	-1	-1
1	4	1
-1	-1	-1

3×1	4×-1	7×-1
8×1	6×4	3×1
4×-1	1×-1	2×-1

همانطور که مشاهده می‌شود، ابتدا مرکز کرنل را روی خانه $[1, 1]$ قرار داده و هر یک از هشت خانه مجاور آن را در خانه متناظرش در کرنل ضرب می‌کنیم؛ در انتها تمام مقادیر را با هم جمع کرده و به جای پیکسل اصلی در عکس نتیجه قرار می‌دهیم. نتیجه به صورت زیر می‌شود:

3	4	7	
8	20	3	
4	1	2	

برای پیکسل‌هایی که توسط هشت خانه احاطه نمی‌شوند، مقدار پیکسل مرکزی را برای خانه‌های ناموجود در نظر بگیرید. برای فهم بهتر به شکل زیر توجه کنید:

0	5	
1	4	

→


0	0	0
0	0	5
0	1	4

ابتدا پیکسل گوشه را به صورت عکس سمت راست تصور کرده و پیکسل‌های گوشه‌ای که وجود ندارند را برابر پیکسل وسطی فرض می‌کنیم؛ پس از آن همانند مرحله قبل ماتریس کرنل را در آن ضرب کرده و مقدار جدید را در عکس خروجی قرار می‌دهیم.

دقت کنید که برای هر پیکسل، هنگام ضرب کردن کرنل باید از مقادیر قدیمی پیکسل‌های اطراف استفاده کنید. مثلاً اگر پیکسل $P_1 = [3, 4]$ را بروزرسانی کردید و به صورت P'_1 در عکس خروجی قرار دادید، هنگام محاسبه پیکسل $P'_2 = [4, 4]$ که مجاور P'_1 است، باید از مقادیر P_1 و P_2 استفاده کنید و نه P'_1 یا P'_2 . همچنین توجه کنید که در مثال‌های فوق از یک عدد برای نشان دادن هر پیکسل استفاده شد در حالی که هر پیکسل در اصل یک سه‌تایی از رنگ‌های آبی، سبز و قرمز است. علاوه بر آن، در هنگام ضرب کرنل‌ها ممکن است مقدار برخی از رنگ‌ها بیش‌تر از ۲۵۵ یا کم‌تر از ۰ شود؛ در این صورت مقدار پیکسل را در حالتی که از ۲۵۵ بیش‌تر شده برابر ۲۵۵ و در حالتی که کم‌تر از ۰ شده، برابر ۰ قرار دهید. توصیه می‌شود برای فهم بهتر این نوع از فیلترها، مثال‌ها و توضیحات [این لینک](#)^۱ را مشاهده کنید.

Gaussian Blur


این فیلتر برای تار کردن عکس استفاده می‌شود. ماتریس کرنل و نتیجه آن به صورت زیر می‌باشد:

نتیجه فیلتر	ماتریس کرنل
	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

¹ <https://setosa.io/ev/image-kernels/>


Sharpen

این فیلتر با استفاده از افزایش کنتراست پیکسل‌های مجاور، گوشه‌های عکس را واضح و برجسته نشان می‌دهد. ماتریس کرنل و نتیجه آن به صورت زیر می‌باشد:

نتیجه فیلتر	ماتریس کرنل
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Emboss

این فیلتر پیکسل‌های عکس را بر اساس روشنایی یا تیرگی پیکسل به یک سایه یا برجستگی تغییر می‌دهد. ماتریس کرنل و نتیجه آن به صورت زیر می‌باشد:

نتیجه فیلتر	ماتریس کرنل
	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$

تغییر رنگ

این نوع از فیلترها، با تغییر در مقادیر RGB مربوط به هر پیکسل، در رنگ‌های عکس تغییر ایجاد می‌کنند.

Invert

این فیلتر، رنگ‌های موجود در عکس را معکوس می‌کند. برای مثال اگر $P = (r, g, b)$ پیکسل موجود در عکس باشد، آنگاه پیکسل نتیجه P' به صورت زیر محاسبه خواهد شد:

$$P' = (r', b', g') = (255 - r, 255 - b, 255 - g)$$



نتیجه فیلتر:

Grayscale

این فیلتر مقدار هر پیکسل را برابر میانگین هر سه مقدار RGB-اش قرار می‌دهد. نتیجه این فیلتر عکس را به صورت سیاه و سفید در می‌آورد. برای مثال اگر $P = (r, g, b)$ پیکسل موجود در عکس باشد، آنگاه پیکسل نتیجه P' به صورت زیر محاسبه خواهد شد:

$$PixelAvg = \frac{1}{3}(r + g + b)$$
$$P' = (r', b', g') = (PixelAvg, PixelAvg, PixelAvg)$$



نتیجه فیلتر:

فرمت تصویر

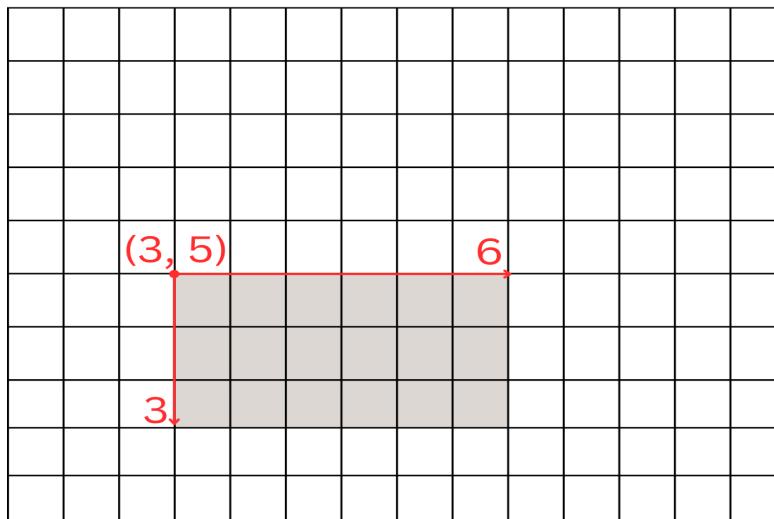
خواندن فایل تصویر

فایل عکسی که به عنوان ورودی به برنامه داده می‌شود، به فرمت bmp می‌باشد؛ برای آشنایی با این نوع فایل، می‌توانید به [این لینک](#) مراجعه کنید. کد خواندن و نوشتن این نوع از فایل به شما داده می‌شود. توجه داشته باشید که این کد صرفاً نحوه استفاده از این نوع فایل را به شما نشان می‌دهد و پیاده‌سازی طراحی شیء‌گرا و استفاده از مفاهیم ارث‌بری در آن بر عهده شما می‌باشد.

View

برخی از فیلترها بر روی یک view از تصویر اعمال می‌شوند که این view به صورت یک رشته به فرم $x:y:w:h$ می‌باشد که x و y موقعیت نقطه شروع view (از گوشه بالا چپ) و w و h ، به ترتیب طول و عرض view را نشان می‌دهند.

به عنوان مثال، view انتخاب شده در شکل زیر، برابر با $3:5:6:3$ می‌باشد:



دقت کنید که x و y ایندکس‌های پیکسل‌ها هستند و از صفر شروع می‌شوند؛ w و h اندازه را نشان داده و تضمین می‌شود مقادیری مثبت و غیر صفر داشته باشند.

فرمت ورودی برنامه

آرگومان‌های خط فرمان

برنامه شما باید ورودی‌های مد نظر را از طریق آرگومان‌های خط فرمان به صورتی که در ادامه مشاهده می‌کنید دریافت کند. تمام فیلترها به ترتیب در ورودی می‌آیند؛ در صورتی که نیاز باشد این فیلتر روی یک View از عکس اعمال شود، ابتدا مقادیر این View به صورتی که پیش‌تر ذکر شد می‌آیند و در ادامه آن باقی فیلترها نوشته می‌شوند. دقت کنید که فیلترها با یک «-» در ابتدای نامشان مشخص می‌شوند و View-ها در فرمت x:y:w:h، بدون هیچ علامت اضافه‌ای می‌آیند. همچنین توجه کنید که برخی از فیلترها روی کل عکس اعمال می‌شوند و نه View-ای از آن؛ در این صورت صرفاً نام فیلتر آمده و هیچ مقداری برای View مشخص نمی‌شود:

```
./ImageEditor <FilterName> [<View>] [<FilterName> [<View>]...]
```

هر <FilterName> به صورت زیر جایگزین می‌شود:

Filter	Blur	Sharpen	Emboss	Invert	Grayscale
Flag	-B	-S	-E	-I	-G

دقت کنید که ترتیب اعمال فیلترها، باید مطابق ترتیب آرگومان‌های اجرای برنامه باشد. برنامه را مثل یک پایپ‌لاین در نظر بگیرید که در طی آن خروجی مرحله قبل به عنوان ورودی به مرحله بعدی داده می‌شود. مثلاً در نمونه زیر ابتدا فیلتر Blur روی کل عکس اعمال می‌شود؛ سپس فیلتر Sharpen روی View مشخص شده از نتیجه مرحله قبل اجرا می‌شود؛ در فیلتر Grayscale روی کل نتیجه مرحله قبل اجرا می‌شود:

```
./ImageEditor -B -S 3:4:30:30 -G
```

روند اجرای برنامه

پس از آنکه ترتیب فیلترها و View-ها مشخص شد، برنامه شما باید در هر خط نام فایل ورودی و فایل خروجی را به ترتیب دریافت کند؛ به ازای هر خط، عکس ورودی را خوانده، فیلترها را اعمال کرده و عکس خروجی را در آدرس مشخص شده ذخیره می‌کند. دقت کنید که ابعاد هر عکس ورودی با دیگری ممکن است متفاوت باشد و تضمین می‌شود که View-های مشخص شده قطعا در این ابعاد قرار می‌گیرند.

نمونه‌های ورودی

ورودی اول

```
./ImageEditor -I -B 0:0:50:50 -G -I
```

```
pic1_in.bmp pic1_out.bmp
```

```
pic2_in.bmp pic2_out.bmp
```

در این نمونه، ابتدا فیلتر Invert روی کل عکس اعمال می‌شود؛ سپس فیلتر Blur روی View مشخص شده (0:0:50:50) اجرا می‌شود؛ در ادامه آن فیلتر Grayscale و در انتها نیز فیلتر Invert اجرا می‌شود.

نکات و نحوه تحویل

- در تمامی مراحل این پروژه سعی کنید از قوانین ارث‌بری استفاده کنید و هر جا که ممکن است رفتار کلاس‌ها را به صورت چندریخت (polymorphic) پیاده‌سازی کنید و از بررسی مجزای کلاس‌ها خودداری کنید.
- به غیر از خطاهای ذکر شده در صورت پروژه، نیاز به رسیدگی به هیچ خطای دیگر نمی‌باشد و تضمین می‌شود ورودی‌ها به درستی به برنامه داده می‌شوند.
- پرونده‌های خود را در قالب یک پرونده‌ی zip با نام `A6-<SID>.zip` در صفحه‌ی elearn درس بارگذاری کنید که SID شماره‌ی دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۱۰۰۰ است، نام پرونده‌ی شما باید `A6-810101000.zip` باشد.
- از zip کردن پوشه‌ای که داخل آن فایل‌های پروژه‌تان قرار دارد خودداری فرمایید.
- دقت کنید که پروژه‌ی شما باید Multi-file باشد و Makefile داشته باشد. همین‌طور در Makefile خود مشخص کنید که از استاندارد C++20 استفاده می‌کنید.
- دقت کنید که نام برنامه قابل اجرای شما باید ImageEditor باشد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

نمرات

- تمیزی کد

- رعایت کردن نام‌گذاری صحیح و انسجام
- عدم وجود کد تکراری
- رعایت دندان‌گذاری²
- عدم استفاده از متغیرهای گلوبال
- استفاده صحیح از متغیرهای ثابت به جای Magic Value-ها

- درستی کد

- آزمون‌های خودکار

- طراحی

- طراحی صحیح و منطقی
- رعایت Encapsulation
- عدم استفاده از if-else یا switch-case و down-cast برای پیدا کردن نوع زیرکلاس
- استفاده صحیح از ارث‌بری و چندریختی
- جداسازی منطق کد از ورودی/خروجی

دقت کنید که موارد ذکر شده لزوماً کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.

² Indentation