مدرسان: **رامتین خسروی، محمدامین صادقی**
طراحان: **آبتین باطنی، یاسمن جعفری، سعید زنگنه**

مهلت تحویل: دوشنبه ۱۳ فروردین ۱۳۹۷، ساعت ۲۳:۵۵

بازسازی[1]

تعاریف زیادی از «کد تمیز»[2] وجود دارد؛ اما احتمالاً یکی از بهترین تعریف‌ها متعلق به «بیارنه استراستروپ»[3] خالق و توسعه‌دهنده‌ی زبان C++ است. وی در تعریف خود از یک کد تمیز، دو مورد زیر را به‌عنوان معیارهای اساسی یک کد تمیز بر می‌شمارد:

- منطق و الگوریتم کد باید آن‌قدر واضح و قابل‌فهم باشد که اشکالات و نقص‌های جزئی نتوانند از چشم برنامه‌نویس و آزمونگر کد دور بمانند؛ ضمن این‌که وضوح کد باید به‌حدی بالا باشد که برنامه‌نویس را از نوشتن کامنت[4] بی‌نیاز کند.
- کارایی[5] برنامه‌ی نوشته‌شده باید در بهینه‌ترین[6] شکل ممکن باشد تا بعدها برنامه‌نویس دیگری به بهانه‌ی بهینه‌سازی[7] برنامه‌ی سابق با ایجاد تغییرات نادرست سبب نامنظم‌شدن و کثیف‌شدن کد نشود.

در عمل، در اکثر مواقع شما بعد از یک طراحی نسبتاً خوب و پیاده‌سازی آن، برای مدتی طولانی از آن کد برای هدف خود استفاده می‌کنید و در طول این مدت تغییرات و قابلیت‌های زیادی را به آن می‌افزایید.

پس از مدتی نه‌چندان طولانی، این تغییرات باعث می‌شوند که شما دیگر عملکرد کد را به‌وضوح متوجه نشوید و به تبع آن، توانایی تغییر و ارتقای کد را نیز از دست می‌دهید. همین زنجیره‌ی اتفاقات به‌ظاهر ساده در تاریخچه‌ی نسبتاً کوتاه توسعه‌ی نرم‌افزاری باعث نابودشدن شرکت‌های بسیاری در این عرصه شده است.

حال با توجه به خطرات و مشکلاتی که یک کد کثیف به همراه دارد، باید راه حلی برای رفع کثیف‌بودن کد و جلوگیری از ایجاد آن ارائه دهیم. شما در این تمرین کامپیوتری با روند بازسازی کد آشنا می‌شوید.

بازسازی عملیاتی است که در طی آن ساختار یک نرم‌افزار به‌صورتی تغییر و بهبود می‌یابد که بدون از دست‌رفتن کارآیی‌ها و تغییر رابط کاربری برنامه، ساختار درونی کد به‌طرز قابل‌توجهی تمیزتر می‌شود.

بنیادی‌ترین مفهوم یاری‌کننده‌ی یک برنامه‌نویس در طی عملیات بازسازی شناخت عناصری است که باعث کثیف‌شدن کدها می‌شوند و به‌اصطلاح به آن‌ها Code Smells گفته می‌شود.

وظیفه‌ی شما در این پروژه بازسازی کد خودتان در پروژه‌ی اول درس و اضافه‌کردن یک قابلیت جدید به آن است؛ بنابراین خوانایی و تمیزبودن کد در این تمرین بیشترین اهمیت را دارد. در ابتدا قابلیت جدید توضیح داده می‌شود و در ادامه توضیحاتی درباره‌ی عملیات بازسازی کد ارائه می‌شود.

کانتکتدان پیشرفته

پس از فراگیرشدن سیستم‌عامل موبایلی که با دوستان خود آن را توسعه داده بودید و استقبال جهانی از آن، کاربران یک مشکل در برنامه‌ی مدیریت مخاطبین گزارش کرده‌اند: آن‌ها نمی‌توانند آدرسی برای مخاطبین خود ذخیره کنند! برای همین، شما باید به‌سرعت این قابلیت را به برنامه‌ی خود اضافه کنید تا کاربران سیستم‌عامل را راضی نگه دارید.

برای همین دستورهای برنامه باید با توجه به توضیحاتی که در ادامه آمده است تغییر کنند:

---

[1] Refactoring

[2] Clean Code

[3] Bjarne Stroustrup

[4] Comment

[5] Performance

[6] Optimal

[7] Optimization

اضافه کردن مخاطب

این دستور باید از گزینه‌ی `-a` پشتیبانی کند که پس از آن آدرس مخاطب قرار می‌گیرد. استفاده از این گزینه در دستور `add` اختیاری است. همچنین توجه کنید که آدرس می‌تواند شامل کاراکتر فاصله باشد؛ برای همین، باید آدرس را عبارتی که بین گزینه‌ی `-a` و گزینه‌ی بعدی می‌آید یا عبارتی که بین گزینه‌ی `-a` و انتهای خط قرار می‌گیرد در نظر بگیرید.

ساختار دستور

```
add <-f first_name> <-l last_name> <-p phone_number> <-e email_address> [-a address]
```

ورودی و خروجی نمونه

ورودی	خروجی
`add -l Hajiloo -e m.taghi@gmail.com -f  Mohammad -p 09191991919 -a Tehran, AmirAbad, UniversityOfTehra`	`Command Ok`

جست‌وجوی مخاطب

اگر مخاطب پیداشده آدرس داشته باشد، باید در هر خط خروجی این دستور، پس از شماره‌ی تلفن، آدرس مخاطب نمایش داده شود. همچنین در نتایج جست‌وجو باید مخاطبینی که واژه‌ی مشخص‌شده زیررشته‌ی آدرس آن‌ها باشد هم نمایش داده شوند.

ساختار هر خط خروجی

```
<id> <first_name> <last_name> <email_address> <phone_number> [address]
```

ورودی و خروجی نمونه

ورودی	خروجی
`search  gmail`	`0 Mohammad Hajiloo m.taghi@gmail.com 09191991919 Tehran, AmirAbad, UniversityOfTehran`

ویرایش مخاطب

دستور `update` هم باید از گزینه‌ی `-a` پشتیبانی کند و اگر این گزینه وجود داشت، آدرس مخاطب را تغییر دهد.

ساختار دستور

```
update <id> [-f first_name] [-l last_name] [-p phone_number] [-e email_address] [-a address]
```

کد تمیز

عواملی در کد وجود دارند که ممکن است باعث کثیف‌شدن آن شوند؛ برای همین در ادامه برخی از آن‌ها که به کثیف‌شدن کد منجر می‌شوند توضیح داده شده‌اند. توجه کنید که در انتهای این پروژه نمره‌ی شما فقط بر اساس عوامل زیر سنجیده می‌شوند و به ازای هر یک از موارد زیر که در کد وجود داشته باشند نمره‌ی شما کاسته خواهد شد. این عوامل خلاصه‌ای از فصل ۱۷ کتاب Clean Code ــ از منابع درس ــ هستند و عبارت بعد از هر عامل شماره‌ی آن عامل را در کتاب نشان می‌دهد.

کامنت‌ها

- در این پروژه کامنت گذاری به هیچ نحوی قابل قبول نیست.

توابع

- آرگومان‌هایی که به‌عنوان خروجی تابع استفاده می‌شوند. یک تابع فقط باید بتواند از طریق مقدار بازگشتی خود بر محیط بیرون تأثیرگذار باشد و تغییر در آرگومان‌ها نباید بر محیط تأثیری داشته باشد. (F2)
- آرگومان‌هایی فقط برای تعیین نحوه‌ی عملکرد کد؛ به‌طور مثال، آرگومانی از نوع بولی[8] برای تعیین مسیر عملکرد کد و اجراشدن یک سناریو و یا دیگری. به طور مثلا پاس دادن یک متغیر به نام flag به تابع، فقط برای اجرای یک بخش از کد در حالتی خاص. (G15)

مشکلات کلی

- تکرار کد. سعی کنید تا حد امکان کدهای خودتان را تکرار نکنید و از ابزارهای زبان C++ که تا کنون یاد گرفته‌اید برای جلوگیری از تکرار استفاده کنید. (G5)
- کدهای مرده: کدهایی که دیگر در هیچ سناریوی اجرای برنامه فرخوانی نمی‌شوند ولی کماکان در متن برنامه موجودند. (G9)
- عدم ثبات و هماهنگی در کد. سعی کنید که همواره از یک الگو و روند در پیاده‌سازی خود استفاده کنید. (G11)
- دندانه‌گذاری[9] و فاصله‌گذاری نامنظم. این مشکل در کدهای شما به‌وفور دیده می‌شود. همانند کدهایی که تاکنون در کلاس دیده‌اید، همواره از یک روند معقول و استاندارد در دندانه‌گذاری استفاده کنید تا کدهای شما خوانا بمانند. (G16)
- انجام عمل نابه‌جا، به این شکل که عملکردی که انتظار داریم با یک تابع و در یک محدوده از کد پیاده‌سازی شود در مکانی دیگر پیاده‌سازی شده باشد. هر عملکرد با توجه به ساختار برنامه‌ی شما مکان مشخصی پیدا می‌کند و در طراحی بالا به پایین، یک موقعیت یکتا خواهد داشت. انجام یک عمل در لایه‌ی اشتباه باعث سردرگمی خواننده خواهد شد. (G17)
- نام توابع باید عبارت هایی امری باشند که به‌وضوح عنوان کنند عملکرد تابع متناظرشان چه خواهد بود. استفاده از عبارت‌های اسمی، عبارت‌های عادی بی‌محتوا و ... فقط باعث ناخوانایی کد می‌شود. (G20)
- استفاده از اعداد جادویی[10]. در کدهایتان همواره به‌جای اعداد بی‌هویت از عبارت‌های ثابت[11] استفاده کنید؛ برای مثال، اگر باید پاسخ مسئله را در عدد π ضرب کنید، به ضرب‌کردن مستقیم در عدد ۳/۱۴، آن را در یک متغیر ثابت با نامی معنی‌دار که در ابتدای برنامه‌ی خود تعریف کرده‌اید ضرب کنید. به این ترتیب، همواره می‌توانید با تغییر فقط یک متغیر عملکرد برنامه را تغییر دهید و تصحیح کنید. (G25)
- عدم پیاده‌سازی عملکرد بدیهی. گاهی عملکرد بدیهی و اصلی‌ای که از یک تابع انتظار داریم پیاده‌سازی نشده است، ولی همان عملکرد به شیوه‌ای پیچیده در کد حضور دارد. (G2)
- انجام بیش از یک کار در توابع. هر تابع باید فقط یک عمل را انجام دهد. (G30)

---

[8] Boolean

[9] Indentation

[10] Magic Numbers

[11] Constants

- استفاده از نام‌های بی‌ارتباط. در نام‌گذاری باید توجه کنید که فهمیدن کاربرد و مکان استفاده از متغیر بسیار پراهمیت است و باید بتوان هر چه سریع‌تر این را دریافت. نام‌گذاری صحیح باعث می‌شود که این فرآیند تسریع شود. شما هم احتمالاً با متغیرهایی با نام‌های a، aa، aaa و aaaa روبه‌رو شده‌اید که هیچ توضیحی ارائه نمی‌دهند و خواننده را گیج می‌کنند. (N1)

آزمودن

یکی از مراحل مهم و قابل‌توجه در حین بازسازی کد آزمودن درستی عملکرد آن است. داشتن یک تابع آزمون به‌ازای هر تابع از برنامه به ما این امکان را می‌دهد که بعد از هر تغییر در هنگام بازسازی، عملکرد تابع را بازبینی کنیم و مطمئن شویم که برنامه همچنان درست کار می‌کند.

شما باید برای **تمامی** توابعی که در پروژه‌ی اول خود پیاده‌سازی کرده‌اید و توابعی که پس از بازسازی اضافه کرده‌اید، تابع آزمون بنویسید. در این تابع باید سعی کنید تا تمامی ظرافت‌ها و عملکردهای جزئی تابع اصلی را مورد بررسی قرار دهید و در پایان، یک پیام مبتنی بر موفقیت‌آمیزبودن یا نبودن چاپ کنید. (با توجه به این که تابع آزمون باید تمامی عملکردهای تابع موردنظر را بررسی کند، هر چه توابع بیشتر و کوچک‌تری داشته باشید، نوشتن توابع آزمون برای آن‌ها ساده‌تر خواهد بود)

پس از هر مرحله از بازسازی، تابع آزمون را صدا کنید تا از درستی کارکرد تابع خود پس از اعمال تغییرات مطمئن شوید.

برنامه‌ی شما بعد از اعمال بازسازی باید قادر به گذراندن تست‌کیس‌های مربوط به پروژه‌ی اول باشد.

git (امتیاز بیشتر!)

git یک سیستم کنترل نسخه است که به کمک آن می‌توان روند تغییرات اعمال‌شده بر روی انواع فایل‌ها را دنبال کرد. git همچنین این امکان را فراهم می‌سازد تا افراد یک گروه به‌صورت همزمان بر روی یک پروژه کار کنند و تغییرات را با یکدیگر به اشتراک بگذارند. از این رو، آشنایی با git و چگونگی استفاده از آن بسیار مهم و کاربردی است.

برای کسب اطلاعات درمورد چگونگی کار با git می توانید به لینک‌های زیر مراجعه کنید:

https://agripongit.vincenttunru.com
https://learngitbranching.js.org

در این پروژه، برای کسب امتیاز بیشتر، در سایت gitlab[12] عضو شده و یک repository خصوصی[13] جدید ایجاد کنید. سپس، فایلی را که برای پروژه‌ی اول خود ارائه دادید روی git بگذارید. پس از اعمال هر تغییر، فایل جدید را با پیامی مناسب ــ که نشانگر تغییرات اعمال‌شده در این نسخه نسبت به نسخه‌ی قبلی است ــ commit و سپس push کنید. همه‌ی commitها را روی branch اصلی (master) push کنید.

توجه کنید که هر commit باید تنها شامل یک تغییر باشد. با اجرای کامل این بخش می‌توانید ۵۰٪ نمره‌ی اضافی علاوه بر نمره‌ی همین پروژه دریافت کنید.

[12] http://gitlab.com/

[13] Private

نکته‌های پایانی

1. توجه کنید که کد نهایی شما توسط تست‌کیس‌های پروژه‌ی ۱ هم آزموده خواهند شد و برای آن نمره در نظر گرفته می‌شود.
2. یک نمونه از بازسازی کد را می‌توانید در repository زیر مشاهده کنید. این کد مربوط به مساله‌ی اول پروژه‌ی صفر است.

- آدرس repository
- کد اولیه
- کد نهایی (بازسازی‌شده)
- فهرست مراحل تغییرات (لیست commitها)

با کلیک کردن روی لینک مشخص شده می‌توانید تغییرات ایجادشده در هر مرحله را مشاهده کنید.

نحوه‌ی تحویل

اگر قسمت git را انجام نداده‌اید، فایل برنامه‌ی خود را با نام R-SID.cpp در صفحه‌ی CECM درس بارگذاری کنید که SID شماره‌ی دانشجویی شماست؛ برای مثال، اگر شماره‌ی دانشجویی شما ۸۱۰۱۱۲۳۴۵ است، نام فایل شما باید R-810112345.cpp باشد.

اگر از git استفاده کرده‌اید، کافی است آدرس repository و نام‌کاربری خودتان در سایت gitlab را در محل آپلود درس بنویسید. دقت کنید که زمان تحویل پروژه‌ی شما، زمان آخرین commitتان در نظر گرفته می‌شود.

- برنامه‌ی شما باید در سیستم‌عامل لینوکس و با مترجم g++ با استاندارد c++98 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- برنامه‌ی شما باید تست‌کیس‌های قبلی پروژه را هم با موفقیت بگذراند.
- از صحت ساختار ورودی‌ها و خروجی‌های برنامه‌ی خود مطمئن شوید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.
- توجه کنید که راه ارتباطی اصلی با دستیاران آموزشی فروم اصلی درس یا ایمیل است و دستیاران آموزشی موظف نیستند به سوال‌های شما در سامانه‌های دیگر مانند **تلگرام** پاسخ دهند.