به نام خالق یکتا و به یاد یوسف زهرا

گزارش تمرین برنامه‌سازی پیشرفته

تمرین سری چهارم

سیدرضا هاشمی‌راد

۳۱ خرداد ۱۳۹۸

فهرست مطالب

لینک دستیابی به فایل ها در گیت هاب

۱ سوال یک

۱. زمانی که بخواهیم مقدار یک متغیر را در یک متغیر دیگر بریزیم بدون آنکه بعدا بخواهیم از متغیر مقصد استفاده نماییم (مثلا برای جابجایی مقدار دو متغیر با هم) و یا زمانی که بخواهیم یک مقدار rValue را داخل یک متغیر دیگر با اپراتور = بریزیم و بخواهیم از کپی شدن زیادی جلوی گیری نماییم از move semantics استفاده می نماییم. به این معنا که به برنامه اعلام می کنیم که به مقدار ورودی داده شده، پس از ذخیره سازی آن در متغیر جدید، نیازی وجود ندارد و می تواند پاک شود.

۲. polymorphism یا همان چندریختی به معنای این است که فرمت های مختلفی را در زمان های مختلف در نظر بگیریم. در برنامه نویسی C++ به این صورت است که می توان یک تابع از یک آبجکت را که با پوینتر و یا رفرنس صدا زده شده است را در زمان های مختلف به شکل های متفاوتی صدا زد و در نتیجه خروجی متفاوتی نیز مشاهده نمود. به عبارت دیگر با نوشتن یک خط مشابه، دو تابع متفاوت در دو کلاس که از یک کلاس بالاتر ارث می برند، اجرا می شود. نکته‌ی دیگر این است که در این که کدام تابع اجرا شود، در زمان اجرای برنامه و نه در زمان کامپایل آن مشخص می شود. در واقع ریخت پوینتر مشخص نیست و می توان فقط به تابعی دسترسی داشت که در کلاس بالاتر وجود دارد. (منبع: کتاب برنامه نویسی مقدماتی معرفی شده در کلاس، صفحه۳۸۴)

۳. با توجه به آنچه در برخی سایت های معتبر برنامه نویسی آمده است، مفهوم abstract به این معنی است که اگر یک تابع را بخواهیم در کلاس والد تعریف نماییم ولی ندانیم که جزئیات آن برای هر یک از کلاس هایی که از آن ارث می برند چگونه است آن را به صورت ابسترکت تعریف می نماییم.

```
virtual int f() = 0
```

برای مثال یک کلاس حیوان داریم که یک تابع حرکت دارد ولی بسته به نوع حیوان جزئیات و روش حرکت متفاوت است که باید در هر کلاس فرزند به طور دقیق تعریف شود.

اگر یک کلاس والد، تماما دارای توابع `pure virtual` باشد، به آن کلاس pure abstract گفته می شود.

۴. همانطوری که در برنامه نویسی پایتون هم خیلی کاربرد دارد، زمانی که یک تابع از کلاس والد را در کلاس وارث آن بازتعریف نماییم، به این عمل override گفته می شود. البته تابعی می تواند بازتعریف شود که از نوعِ virtual باشد. برای استفاده از این خاصیت باید دو تابع دقیقا یکی باشند. همچنین برای استفاده می توان

از override استفاده نمود که از تفاوت در تابع داخل کلاس مشتق شده و تابع ویرچوال داخل کلاس والد جلوگیری می کند و در این صورت خطا می دهد.

همچنین از final هم برای جلوگیری از مشتق شدن کلاس و هم برای جلوگیری از اورراید شدن تابع یک کلاس استفاده می شود.

۵. زمانی که قبل از یک تابع از این دستور استفاده می نماییم در واقع می خواهیم به کامپایلر بگوییم که به جای اینکه هر بار تابع را فراخوانی کند، آن را یک بار داخل کد قرار دهد. مثل اینکه کامپایلر، به جای فراخوانی هر بار تابع، آن را هر جا که لازم باشد، اجرا کند. (متن تابع داخل بدنه ی کد قرار می گیرد که حافظه کمتری نیز اشغال می شود.) البته در کامپایلرهای جدید عملا این کار را خود کامپایلر انجام می دهد.

برای تعریف یک متغیر استاتیک در فایل هدر، لازم است که آن را به صورت inline تعریف نماییم. همچنین زمانی که یک متغیر یا یک تابع را به این صورت تعریف می نماییم به راحتی می توانیم در یک فرایند ترجمه (یک هدر و چند cpp) از آن استفاده نماییم. به طور کلی زمانی که بخواهیم به یک متغیر در فایل هدر مقدار بدهیم لازم است از این کلیداژه استفاده نماییم.

۶. با توجه به متن کتاب در صفحه ۳۹۶، در می یابیم برای کلاس های دارای کانستراکتور با یک ورودی، اگر احیانا بین کلاس و یک عدد مقایسه ای انجام شود، کامپایلر عدد از روی عدد یک آبجکت ساخته و عمل مقایسه را انجام می دهد که مطلوب ما نیست. برای اینکه این اتفاق رخ دهد و فقط در جایی که واقعا کانستراکتور صدا زده می شود، آبجکت ساخته شود، از کلیدواژه explicit قبل از کانستراکتور استفاده می نماییم.

۲ سوال دو

بر اساس منطقی که وکتور عمل می نماید، برای سرعت بیشتر در ذخیره‌سازی داده‌های جدید push_back شده، همواره capacity وکتور برابر است با توان دوم بعدی مقدار فعلی سایز. با اجرای این کد این منطق را که در تمرین قبلی نیز پیاده سازی نموده بودیم، مشاهده می نماییم. مثلا در سایز ۷، capacity برابر ۸ و در سایز ۸ برابر ۱۶ خواهد بود.

اگر قبل از پوش بک کردن، از تابع reserve(1000) استفاده نماییم. در واقع مقدار اولیه capacity را برابر با ۱۰۰۰ قرار می دهد ولی سایز همچنان با پوش بک کردن اضافه می شود. این کار به سرعت دسترسی به حافظه بسیار می افزاید چرا که هر بار که capacity دو برابر می شود، کل وکتور دوباره کپی می شود که اصلا خوب نیست. حال اگر باز هم سایز از ۱۰۰۰ بیشتر شود، مقدار capacity دو برابر گشته و به همان ترتیب قبل کار می کند.

۳ سوال چهار

بله لازم بود که حتما به این صورت تعریف گردد زیرا در صورتی که این اتفاق رخ نمی داد، زمانی که با استفاده از یک پوینتر از نوع Shape می خواستیم این توابع را اجرا نماییم، خود توابع موجود در Shape اجرا می شد، نه توابعی که در کلاس های مشتق شده نوشته شده است. مشکل زمانی بیشتر می شود که مثل این سوال داخل توابع مشتق شده با هم متفاوت باشد، یعنی اگر پوینتر از نوع دایره باشد، باید تابع موجود در دایره اجرا گردد و به همین ترتیب. لذا تعریف به این صورت قطعا ضروری است.

۴ سوال پنج

با توجه به ویژگی پشته که قرار است در این سوال پیاده سازی شود، یکی از روش های پیشنهادی برای پیاده سازی استفاده از کلاس های تو در تو است. به این معنا که داخل کلاس پشته، کلاس دیگری را تعریف می نماییم که وظیفه آدرس دهی درون پشته را برای تک تک ورودی ها به عهده می گیرد و این مورد را مدیریت می نماید. داخل هر آبجکت، یک متغیر از جنس T که جنس پشته است و یک پوینتر از نوع Obj قرار دارد که ادرس آبجکت بعدی را نگه می دارد.

در ادامه به پیاده سازی توابع مورد نیاز در main داده شده می پردازیم که به سادگی انجام شد. (مطابق کتاب)

به این نکته نیز توجه نمایید که کلاس پشته برای اینکه درس کار کند، به کپی کانتستراکتور و اپراتور مساوی نیاز دارد زیرا در صورت عدم تعریف این دو به دلیل new کردن متغیرهای جدید در پشته، این مشکل پیش می آید که یک آدرس دوبار پاک شود. البته به دلیل اینکه در تابع main داده شده، این نکته استفاده نشده بود این مورد پیاده سازی نشد. (ولی پشته کامل حتما نیاز دارد این مورد را داشته باشد.)

۵ سوال شش

نکته ای در این سوال باید در نظر گرفت این است که همانطور که در کتاب نیز توضیح داده شده است، دستور std::remove() به تنهایی برای حذف عدد ۲ از وکتور کافی نیست زیرا این تابع به این صورت عمل می نماید که صرفا ۲های موجود در وکتور را به انتهای آن انتقال می دهد و در خروجی خود اندیس آخرین خانه درست وکتور را به صورت شمارنده بر می گرداند. لذا باید پس از اجرای این دستور با گرفتن این شمارنده سایر المنت های وکتور از شمارنده تا انتها را حذف نماییم.

برای دو برابر کردن مقادیر وکتور از یک تابع بازگشتی استفاده شده است. البته با استفاده از توابع کتابخانه استاندارد هم می توانستیم با یک خط کد زدن این کار را انجام دهیم. به هر حال استفاده از تابع بازگشتی هم یک روش است.

ابتدا به کمک یک تابع بازگشتی میانگین را محاسبه نموده و سپس با کمک یک تابع بازگشتی دیگر المان ها را طوری مرتب می نماییم که خواسته سوال برآورده شود.

با کپی وکتور در یک `set` مقادیر تکراری آن را حذف می نماییم. بعد از حذف متغیرهای تکراری سایز وکتور را تغییر می دهیم.

با ترکیب تابع `erase` و `find(4)` درایه های بزرگتر مساوی ۴ (بزرگتر از سه) را حذف می نماییم.

توجه: برای چاپ وکتور بدون حلقه از دستور ذیل استفاده نمودیم.

```
std::copy(vec.begin(), vec.end(), std::ostream_iterator<int>(std::cout, " "));
```

۶ سوال هفت

ابتدا وکتور به طول ۵۰ با مقدار اولیه صفر را می سازیم. سپس با استفاده از تابع بازگشتی، مقادیر تصادفی بین صفر تا ۵۰ را در آن می ریزیم.

در مرحله بعد برای حذف عناصر تکراری آن را در یک سِت کپی می ریزیم.

حال در بخش بعدی با کمک تابع بازگشتی جدیدی، عناصر ۱ تا ۵۰ را به صورت تصادفی در یک سِت می ریزیم. ویژگی جالب این است که به دلیل وِیژگی ست بودن، اعداد تکراری وارد مجموعه نمی شوند.

برای قسمت آخر نیز از تابع `std::transform` استفاده می نماییم.

۷ سوال Django

روند حل این سوال مطابق آنچه در کلاس بیان شد و همچنین آنچه در راهنمای راه‌اندازی Django آمده است انجام می شود. در روند حل سوال به دلیل توضیحات کامل راهنما، مشکل خاصی پیش نیامد به سادگی آنچه در صورت سوال انتظار می رفت انجام شد. توجه به این نکته هم ضروری است که می توان برای زیبا سازی کار، از فایل css استفاده نمود.

۸ ارسال روی گیت

برای ارسال روی گیت، مطابق آنچه در کلاس گفته شد ابتدا یک ریپازیتوری[۱] به نام درس برنامه سازی پیشرفته درست کردیم و سپس در سیستم لوکال خود آن را دانلود کرده و فایل های مورد نظر را با پوشه بندی مناسب در دایرکتوری ایجاد شده قرار دادیم. با استفاده از دستور

[۱] Repositoriy.

```
git add DIRECTORYofFILES
```

فایل ها را اضافه کرده و سپس با دستور

```
git commit -m "MYCOMMENT"
```

آنها را روی سیستم لوکال کامیت یا ثبت می کردیم. حال با دستور

```
git push origin master
```

فایل های کامیت شده را روی سرور github.com و روی برنچ مستر بارگذاری نمودیم.

توجه نمایید که در ابتدا برای گرفتن ریپو از دستورهای ذیل به ترتیب استفاده نمودیم. (از طریق ssh)

```
git init
git clone git@github.com:SRHashemirad/AUT_AP_course.git
```

همچنین قابل ذکر است که فایل `.gitignore` را به به صورتی قرار می دهیم که فایل هایی که در حین کامپایل ایجاد می شوند[۲] را ثبت نکرده و همچنین نسخه هایی که نرم افزار emacs ایجاد می کند[۳] را نیز ثبت ننماید.

شما می توانید برای دستیابی به فایل ها در گیت هاب از این لینک اقدام نمایید.

با تشکر فراوان از حسن دقت نظر و توجه شما

[۲] *.o

[۳] *.*~