استاد: دکتر رامتین خسروی
طراحان: بردیا اقبالی، حسنا سادات آزرم‌سا، رستا تدین

مهلت تحویل: دوشنبه ۲۲ اردیبهشت ۱۳۹۹، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با مفاهیم اولیهٔ وراثت[1] و چندریختی[2] است. انتظار می‌رود تکنیک‌های برنامه‌نویسی‌ای را که در کلاس درس یا در هنگام تحویل تمرین‌های قبلی فراگرفته‌اید به طور کامل در این تمرین به‌کار گیرید. این تمرین از دو سوال تشکیل شده است که سوال دوم آن امتیازی است. طراحی کلاس‌ها، نحوهٔ 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.

مدیریت استثناها[3]

در طول اجرای برنامه، ممکن است انواع خطا در برنامه شما رخ دهد. در این تمرین، انتظار می‌رود شما با پرتاب[4]کردن نمونه[5] هایی از کلاس‌هایی که در ادامه ذکر خواهد شد و گرفتن[6] آن‌ها به خطاها رسیدگی کنید. این کلاس‌ها از کلاس `std::exception` ارث برده‌اند و شما باید تابع عضو[7] `what` این کلاس‌ها را بازنویسی[8] کنید و در هنگام گرفتن این استثنا[9]ها، خروجی تابع `what` را در جریان خطای استاندارد[10] چاپ کنید.

خطاهایی که انتظار می‌رود که شما در برنامه‌تان به آن‌ها رسیدگی کرده باشید، در شرح تمرین آمده‌اند.

۱. تولیدکننده json

فرمت json

json یک قالب استاندارد متن‌باز[11] است که امکان تبادل داده‌ها را به صورت خوانا برای ماشین و انسان (مخصوصاً در وب) با استفاده از جفت‌های خصوصیت-مقدار[12] ممکن ساخته است. در ادامه آن را اندکی بیشتر توضیح می‌دهیم.

فرض کنید که شخصی یک بالن دارد و شما آن را می‌خواهید. یکی از روش‌های ارسال بالون این است که فرد دیگر عملاً آن را بسته‌بندی کند و از طریق پست برای شما ارسال نماید. اما یک روش دیگر آن است که فرد خصوصیات بالون خود را برای شما توضیح دهد تا شما بتوانید همان بالون را بدون دسترسی عملی فیزیکی به آن، به طور دقیق بازسازی نمایید. بالون آن فرد می‌تواند

[1] Inheritance
[2] Polymorphism
[3] Exception Handling
[4] Throw
[5] Instance
[6] Catch
[7] member function
[8] Override
[9] Exception
[10] Standard Error Stream
[11] Open-source
[12] attribute–value

خصوصیاتی از قبیل جنس: پلاستیک، رنگ: قرمز، قطر: ۲۵ سانتی‌متر، گاز: هلیوم داشته باشد. این چهار جفت خصوصیت: کلید برای این که بتوانید بالون را تجسم کنید کافی هستند. json نیز در زمان ارسال داده‌ها به همین ترتیب عمل می‌کند.
فرمت json برای این مثال را در زیر می‌بینید.

```
{
  "material":"Plastic",
  "color":"red",
  "diameter":25,
  "gas":"Helium"
}
```

ساختار json

محتوای داخل json با آکولاد باز شروع و با آکولاد بسته تمام می‌شود که این بلاک به عنوان آبجکت مادر نیز شناخته می‌شود.

آبجکت‌ها

هر شیء یا آبجکت در json شامل مجموعه‌ای نامرتب از داده‌ها به صورت نام:مقدار است. (مانند "material":"Plastic" که در این‌جا نام "material" و مقدار "Plastic" است.) نام یک رشتهٔ یکتاست. داده‌ها در آبجکت با کاما (,) از یکدیگر جدا می‌شوند. مقدار هر داده می‌تواند رشته، عدد، آبجکت یا آرایه باشد.

آرایه‌ها

آرایه‌ها شامل چندین مقدار هستند که با براکت باز ([) شروع و با براکت بسته (]) تمام می‌شوند. مقادیر درون آرایه با کاما از یکدیگر جدا می‌شوند. مقادیر آرایه می‌تواند از نوع رشته، عدد، آبجکت یا آرایه باشد. مثال:

```
{
    "Ages": [25, 12, 65, 16]
}
```

شرح تمرین

در این تمرین یک ساختار json به گونه‌ای که در ادامه توضیح داده می‌شود می‌سازید و در نهایت این ساختار را با فرمت json چاپ می‌کنید. جزئیات توابعی که انتظار می‌رود پیاده‌سازی کنید در ادامه آمده است. در صورت بروز خطا برنامه با پرتاب خطای مناسب تمام می‌شود. این تمرین با فایل main تست می‌شود که هنگام تست در کنار فایل‌های پروژه‌تان قرار می‌گیرد و ترتیبی از تابع‌های زیر در main صدا زده می‌شوند. یک نمونه فایل main همراه پروژه آپلود شده‌است. در صورت وجود خطا ضمن پرتاب پیام مناسب برنامه خاتمه می‌یابد.

افزودن داده به آبجکت

با فراخوانی دو تابع زیر داده با نام key و مقدار value به آبجکت با شناسه parentId اضافه می‌شود.

```
void addStringToObject(int parentId, string key, string value);
void addIntegerToObject(int parentId, string key, int value);
```

افزودن آرایه یا آبجکت به آبجکت

درصورتی که مقدار داده از نوع آرایه یا آبجکت باشد از تابع زیر برای افزودن داده استفاده می‌کنیم. type نشان می‌دهد که مقدار داده از نوع آرایه یا آبجکت است که می‌تواند یکی از دو مقدار "array" و "object" را داشته باشد. در نهایت تابع شناسه‌ای یکتا مربوط به آبجکت یا آرایه جدید بازمی‌گرداند.

```
int addContainerToObject(int parentId, string key, string type);
```

خطاها

در هر یک از سه تابع بالا باید خطاهای زیر بررسی شود (خطای سوم مربوط به تابع سوم است).
۱) اگر parentId وجود نداشت یا مربوط به هیچ آبجکتی نبود یک استثنا با پیام "Invalid id." پرتاب شود.
۲) اگر key یکتا نبود و قبلا در آبجکت مورد بحث استفاده شده بود یک استثنا با پیام "Duplicate key." پرتاب شود.
۳) اگر مقدار type یکی از دو مقدار تعریف شده نبود یک استثنا با پیام "Undefined type." پرتاب شود.

افزودن مقدار به آرایه

همانند بالا سه تابع زیر را برای افزودن مقدار به آرایه تعریف می‌کنیم.

```
void addStringToArray(int parentId, string value);
void addIntegerToArray(int parentId, int value);
int addContainerToArray(int parentId, string type);
```

خطاها

در هریک از سه تابع بالا باید خطاهای زیر بررسی شود(خطای سوم مربوط به تابع سوم است).
۱) اگر parentId وجود نداشت یا مربوط به هیچ آرایه‌ای نبود یک استثنا با پیام "Invalid id." پرتاب شود.
۳) اگر مقدار type یکی از دو مقدار تعریف شده نبود یک استثنا با پیام "Undefined type." پرتاب شود.

چاپ با فرمت json

با فراخوانی تابع زیر آبجکت مشخص‌شده به فرمت json چاپ می‌شود. id مشخص‌کننده آبجکتی است که باید چاپ شود. شناسه آبجکت مادر را 0 فرض می‌کنیم. توجه کنید دو آبجکت یا آرایه با شناسه یکسان نباید وجود داشته باشد.

```
void print(int id);
```

خطاها

۱) اگر id وجود نداشت یا مربوط به هیچ آبجکتی نبود یک استثنا با پیام "Invalid id." پرتاب شود.

روش چاپ

روش چاپ خروجی باید به شکل pretty print باشد. در این روش برای پرینت آبجکت آکولاد باز، هریک از داده‌ها و آکولاد بسته هرکدام در یک سطر چاپ می‌شوند به صورتی که هریک از داده‌ها یک tab(چهار خط فاصله[13]) با شروع آبجکت فاصله دارد. برای پرینت آرایه هم به همین ترتیب عمل می‌شود. می‌توانید برای آشنایی بیشتر حالت‌های مختلف را در این لینک امتحان کنید.

مثال

تابع فراخوانی‌شده	نتیجه

`addStringToObject(0, "color", "red");`	`{`
	`  "color": "red"`
	`}`

`addIntegerToObject(0, "diameter", 25);`	`{`
	`  "color": "red",`
	`  "diameter": 25`
	`}`

`int coursesId = addContainerToObject(0, "courses", "array");`	`{`
	`  "color": "red",`
	`  "diameter": 25,`
	`  "courses": []`
	`}`

`addStringToArray(coursesId, "AP");`	`{`
	`  "color": "red",`
	`  "diameter": 25,`
	`  "courses": [`
	`  "AP"`
	`  ]`
	`}`

`int gradesId = addContainerToObject(0, "grades", "array");`	`{`
	`  "color": "red",`
	`  "diameter": 25,`
	`  "courses": [`
	`  "AP"`
	`  ],`
	`  "grades": []`
	`}`

`addIntegerToArray(gradesId, 20);`	`{`
	`  "color": "red",`
	`  "diameter": 25,`
	`  "courses": [`
	`  "AP"`
	`  ],`
	`  "grades": [`
	`  20`
	`  ]`

[13] space

	}

int infoId = addContainerToObject(0, "info", "object");
 	
	{
	  "color": "red",
	  "diameter": 25,
	  "courses": [
	  "AP"
	  ],
	  "grades": [
	  20
	  ],
	  "info": {}
	}

addStringToObject(infoId, "name", "hosna");
 	
	{
	  "color": "red",
	  "diameter": 25,
	  "courses": [
	  "AP"
	  ],
	  "grades": [
	  20
	  ],
	  "info": {
	  "name": "hosna"
	  }
	}

۲. ربات‌های اجتماعی (امتیازی)

الگوی طراحی مشاهده‌گر[14]

مشاهده‌گر یک الگو برای تعریف و پیاده‌سازی وابستگی چند به یک میان اشیا است به صورتی که زمانی که تغییری در یک شیء اتفاق می‌افتد، تمامی اشیایی که به آن وابسته‌اند خبردار می‌شوند و به صورت خودکار به‌روزرسانی می‌شوند. برای مطالعهٔ روش کارکرد این الگو به این لینک و برای دیدن یک پیاده‌سازی سادهٔ C++ از آن به این لینک مراجعه کنید.

شرح مسئله

در یک سیستم پیام‌رسان برای ربات‌ها، کانال‌های ارتباطی وجود دارند که ربات‌ها می‌توانند در آن‌ها عضو شوند و در آن‌ها پیام بفرستند. زمانی که پیام جدیدی در کانالی که ربات در آن عضو است منتشر می‌شود باید ربات از آن باخبر شود و با توجه به رفتار تعریف‌شدهٔ خودش عملی انجام دهد. توجه کنید برای این‌که ربات‌ها پیامی در کانالی بفرستند نیازی نیست در آن کانال عضو باشند و عضو شدن تنها برای اطلاع از پیام‌های ارسال‌شده به آن کانال است. در ادامه اعمال ممکن این سیستم و انواع ربات‌ها آمده است.

[14] Observer Design Pattern

در این تمرین باید برای پیاده‌سازی رابطهٔ میان کانال‌ها و ربات‌ها از الگوی مشاهده‌گر استفاده کنید. **توجه کنید** که درک کارکرد این الگو و فهم چرایی مناسب بودن این راه‌حل در این مسئله مهم‌ترین قسمت این سوال است و نوشتن کد آن به‌تنهایی هیچ نمره‌ای ندارد.

دستورها

اضافه‌کردن کانال

در این سیستم می‌توان با استفاده از دستور زیر یک کانال جدید با یک نام یکتا ساخت. فرض کنید نام کانال یک کلمه است. نیازی به بررسی این معیار نیست. (در ورودی‌های آزمون حتماً یک کلمه است.)

```
add_channel <name>
```

اگر نام کانال تکراری بود پیام "Channel already exists" داده شود.

اضافه‌کردن ربات‌ها

۱. ربات اکو

این ربات در کانال مبدا خود (src_channel) عضو می‌شود و هر پیامی را که در این کانال می‌آید در کانال مقصد خود (dest_channel) بازگو می‌کند. با دستور زیر یک ربات از این نوع ساخته می‌شود:

```
add_bot echo <src_channel> <dest_channel>
```

در صورتی که کانال مبدأ یا مقصد وجود نداشت باید پیام "Channel does not exist" داده شود و اگر این دو کانال یکی بودند باید پیام "Source and destination cannot be the same channel" چاپ شود.

۲. ربات واقعه‌نگار

این ربات به صورت خودکار در تمامی کانال‌های موجود در زمان ساختش عضو می‌شود و تمامی پیام‌هایی را که دریافت می‌کند در فایلی که به آن داده شده است می‌نویسد. با دستور زیر یک ربات از این نوع ساخته می‌شود:

```
add_bot logger <filename>
```

۳. ربات دوستانه

این ربات در یک کانال عضو می‌شود و اگر پیام "Hi" دریافت کند در جوابش یک پیام "Hello" در همان کانال می‌فرستد. با دستور زیر یک ربات از این نوع ساخته می‌شود:

```
add_bot fred <channel>
```

در صورتی که کانال وجود نداشت پیام "Channel does not exist" داده شود.

۴. ربات کتابخانه‌دار

این ربات هر پیامی که در یک کانال عضو می‌شود و هر ۵ پیامی که دریافت کند یک پیام "!Quiet" در همان کانال می‌فرستد. با دستور زیر یک ربات از این نوع ساخته می‌شود:

```
add_bot librarian <channel>
```

در صورتی که کانال مبدا یا مقصد وجود نداشت پیام "Channel does not exist" داده شود.

ارسال دستی پیام

با این دستور می‌توان یک پیام از طرف کاربر در یک کانال منتشر کرد.

```
tell <channel> <message>
```

در صورتی که کانال مبدا یا مقصد وجود نداشت پیام "Channel does not exist" داده شود.

نحوهٔ تحویل

- کدهای هر بخش را در پوشه‌ای جداگانه با **شماره آن بخش** قرار دهید و این پوشه‌ها را در قالب یک پروندهٔ زیپ با نام A6-SID.zip در صفحهٔ CECM درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۷۹۹۹ باشد، نام پروندهٔ شما باید A6-810197999.zip باشد که شامل یک (یا دو پوشه در صورت پیاده‌سازی بخش امتیازی) با نام‌های 1و 2 است.
- **دقت کنید** که پرونده زیپ آپلودی شما باید پس از Unzip شدن شامل پوشه‌های پروژه شما باشد و از زیپ کردن پوشه‌ای که داخل آن پوشه‌های پروژه‌تان قرار دارد خودداری فرمایید.
- این پروژه حتما باید به روش شیءگرایی و به صورت Multi File پیاده‌سازی شود همچنین استفاده از makefile اجباری است.
- برنامهٔ شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++11 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود. **دقت کنید** که باید در makefile خود مشخص کنید که از استاندارد c++11 استفاده می‌کنید.
- **دقت کنید** برای بخش اول این تمرین، یک رابط در اختیار شما قرار داده شده است که درستی برنامهٔ شما از طریق فراخوانی توابع این رابط سنجیده می‌شود. در واقع تابع اصلی[15] برنامهٔ شما با توابع اصلی آزمون[16] جایگزین می‌شود و خروجی برنامه مورد بررسی قرار می‌گیرد. **دقت کنید** که نام پرونده‌ای که تابع اصلی شما برای هر یک از پرسش‌ها در آن قرار دارد باید **main.cpp** باشد که با توابع اصلی آزمون جایگزین می‌شود. برای آشنایی بیشتر با رابطه برنامه‌نویسی کاربردی[17]، می‌توانید از این لینک استفاده کنید.
- برنامه شما باید نسبت به خطاهای مختلف مقاوم باشد و در صورت وقوع خطا با چاپ کردن پیام مناسب به کاربر اطلاع دهد. پیام‌های خطایی که در صورت پروژه ذکر نشده است در آزمون خودکار بررسی نمی‌شوند و تنها در تحویل حضوری بررسی می‌شوند.
- **تأکید می‌شود** در پایان رشته‌هایی که در توابع what در استثناها بر می‌گردانید حتما کاراکتر '\n' وجود داشته باشد.

[15] Main

[16] Test

[17] Application Programming Interface

- **تأکید می‌شود** که هدف از پروژه طراحی و استفادهٔ صحیح از مفاهیم وراثت و چندریختی است و استفاده از **if یا switch case** به جای وراثت قابل‌قبول نخواهد بود.تنها هنگام تجزیه[18] ورودی و برای تشخیص آن که چه نوع کلاسی ساخته شود استفاده از آن ایرادی ندارد.
- تمیزی کد، شکستن مرحله‌به‌مرحلهٔ مسئله و طراحی مناسب، در کنار تولید خروجی دقیق و درست، بخش مهمی از نمرهٔ شما را تعیین خواهد کرد.
- درستی برنامهٔ شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید. همچنین دقت شود که نام پروندهٔ اجرایی شما برای هر بخش باید شامل **نام آن بخش** باشد. برای مثال، نام پروندهٔ اجرایی مربوط به بخش اوّل باید به صورت 1.out باشد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

[18] parse