گزارش تمرین سری دوم

هدف ازین تمرین نوشتن دو کلاس classroom و floor به گونه است که بتوان ابتدا اطلاعات هر کلاس شامل تعداد صندلی ها، دمای کلاس و غیره را با استفاده از کلاس classroom به دست اورد و سپس اطلاعات تمامی کلاس های درون یک طبقه را با استفاده از کلاس floor به دست اوریم و نمایش دهیم.

کلاس classroom

ابتدا در فایل h. متغیر ها و prototype توابع را مینویسیم.

در ابتدای فایل h. از #ifndef استفاده میکنیم تا اگر از کلاسی در کلاس دیگری استفاده کردیم یعنی include کردیم conflict یا error نداشته باشیم.

هنگام تعریف کردن توابعی که دارای ۱ خط میباشند که اصولا توابه getter و setter هستند میتوان در همان فایل h. تعریف کرد. اما من برای یکسان بودن ک و نوشته ها تعریف ان ها را همانند بقیه ی توابع در فایل cpp. انجام داده ام.

- علاوه بر متغیر ها و توابعی که در صورت سوال امده متغیرها و توابع دیگری به کلاس اضافه کرده ام.

Seq این متغیر ارایه ای است از کلاس های موجود بر اساس توالی کلاس ها

Place نشان دهنده ی اندیس object کنونی classroom در seq است.برای مثال در اینجا ابتدا اندیس a صفر است و سپس بعد از مشخص کردن کلاس های سمت چپ و راست و درست شدن ترتیب کلاس ها اندیس ان ۱ میشود.

```
-       auto a = std::make_shared<Classroom>("411", 40);
-       auto b = std::make_shared<Classroom>("413", 30);
-       auto c = std::make_shared<Classroom>("415", 20);
-       a->setLeft(c);
-       a->setRight(b);
```

Guass_output خروجی توزیع گاوسی با میانگین ۲۷ و انحراف معیار ۳ میباشد. مقدار اولیه ی این متغیر ماکزیمم حالتی است که این توزیع میتواند داشته باشد.

No متغیری است که تعداد کلاس ها را نشان میدهد و میتوان به صورت private تعریف کرد. در اینصورت نیاز به تابع getter برای دریافت ان می باشد.(دلیل نوشتن تابع totalObjects) اما در فایل تست به طور مستقیم به این متغیر دسترسی پیدا کرده پس باید ان را به صورت public تعریف کنیم که در اینصورت نیازی به نوشتن تابع totalObjects نیست. مقدار اولیه ی نامبر نباید در هنگام ان در فایل h. باشد زیرا هر بار که object جدیدی از این کلاس ساخته میشود تمامی این مقادیر به ان assign میشود و no هر بار ضفر میشود در حالیکه ما میخواهیم no مقدار قبلی خودش را حفظ کند و با هر object جدید مقدار ان اضافه شود. به همین دلیل ان را از نوع static int در نظر میگیریم به مقدار دهی ان را در فایل cpp. انجام میدهیم با این تعریف کامپایلر که در اول برنامه این فایل را بررسی میکند یکبار مقدار ان را ست میکند و سپس در تابع constructor مقدار ان را اضافه میکنیم.

توابع getName,gerSeq,setSeats نیز برای تغییر یا دریافت متغیر های خصوصی نوشته شده اند.

- عملکرد هر تابع در این کلاس به شرح زیر می باشد:

هر بار که یک کلاس ساخته میشود تابع constructor صدا شده پس برای داشتن کل کلاس ها باید no را در این تابع زیاد کنیم(متغیری از نوع static int) . در این تابع ابتدا قصد داشتم pc را به عنوان اولین عضو seq به ان اضافه کنم ولی با توجه به مشکلاتی که پیش می امد و ارور هایی که وجود داشت نتوانستم بنابراین ۰ را به عنوان نماینده ی pc به seq اضافه کردم. اگر راه حل اول درست نوشته میشد کد های بعدی برای استفاده کلاس Floor راحتتر بود اما الان برای هر مرحله باید index pc را بررسی کنم.

برای تابع destruct برای اطمینان seq را دیلیت میکنیم تا فضای ان ازاد شود.

برای تابع temperature باید هر بار عددی به صورت رندم از تابع توزیع گوسی با میانگین ۲۷ و انحراف معیار ۳ به ما بازمیگرداند. برای ورودی های این تابع یک مقدار Boolean را اضافه کردم. به این دلیل که وقتی کلاس ساخته میشود و تابع صدا زده میشود یکبار عدد رندم بگیرد و در بقیه ی پروسه که ممکن است نیاز باشد دوباره تابع صدا زده بشود با ورودی ۰ یا false صدا بشود که تا دما مقدار جدیدی اخذ نکند. البته برای اینکه مقدار دما با چندین بار صدا زدن تابع تغییری نکند میتوان از srand((time(0)) استفاده کرد و دیگر ورودی Boolean نداشته باشیم. اما مشکلی که در این حالت وجود دارد این است که دما برای همه ی کلاس ها یکسان میشود بنابراین برای داشتن دماهای متفاوت برای کلاس ها و ثابت ماندن ان ها در طول برنامه ناگزیر به اضافه کردن ورودی به تابع شدم. نکته ی دیگر در این تابع این است که با توجه به خروجی مشخص شده در صورت سوال منظور انتخاب رندم دما در بازه ی دمایی منایع برای توزیع گوسی است. بازه ی مناسب معمولا بین $\mu \pm 3\sigma$ می باشد. البته رابطه ی تابع گوسی هم نوشته شده در صورت نیاز می توان مقدار آن را به عنوان خروجی تابع در نظر گرفت.

کلاس Floor

به دلیل مشکل گفته شده مجبور به اضافه کردن تابع findInd و متغیر index شدم.

برای constructor یک حالت وجود دارد که نام طبقه و پوینتر را که میفرستیم. که در این حالت باید اندیس پوینتر را بیابیم.

برای حالت دیگر نام طبقه، تعداد کلاس ها و به تعداد کلاس ها پوینتر های classroom object را به عنوان ورودی میدهیم. برای دریافت وروردی های متغیر که به صورت لیست قرار بگیرد از کتابخانه ی cstdarg استفاده میکنیم. برای استفاده ازین کتابخانه لیستی از نوع va_list مینویسم و ابتدای لیست با مشخص کرده و سپس درون لیست iterate میکنیم و نوع متغیری که وجود دارد و انتظار دریافت ان را داریم مشخص میکنیم و سپس va_end را صدا میزنیم. برای این حالت اولین متغیر را به عنوان پوینتر در نظر میگیریم و کلاس های بعدی را در سمت راست پوینتر قرار میدهیم.

برای تابع findInd متغیری از نوع iterator در نطر میگیریم که در بردار به دنبال المان ۰ باشد و اندیس ان را برگرداند. زیرا ۰ را نماینده ی پوینتر در seq قرار دادیم. این اندیس بسایر با اهمیت است زیرا بقیه ی اعمال و مکان ها نسبت به این اندیس صورت میگیرد.

برای تابع operator دو حالت + یا – بودن i وجود دارد.اگر مثبت باشد، باید به تعداد گفته شده به سمت راست برویم البته از نقطه ی شروع پوینتر و کلاس ها را به صورت حلوقی در نظر بگیریم. برای این حالت مکان مورد نظر باقیمانده ی جمع i و اندیس پوینتر به تعداد کل کلاس ها می باشد. اگر i منفی باشد، باید از پوینتر به تعداد گفته شده به سمت چپ حرکت نماییم. در این

حالت ابتدا باید باقیمانده ی i به تعداد کلاس ها محاسبه سپس حاصل را با تعداد کلاس و اندیس پوینتر جمع کنیم. باقیمانده ی حاصل جمع به تعداد کلاس ها مکان مورد نظر میشود.

برای جابه جایی صندلی ها، اگر مبدا همان پوینتر باشد یا کلاس های چپ و راست ان باید طریقه ی دسترسی برای یافتن تعداد صندلی ها متفاوت است و این حالت برای زمانی که مقصد pc باشد یا نه نیز صادق است.

برای تعداد کل صندلی ها باید در برداری که در کلاس classroom تشکیل دادیم iterate کنیم و تعداد صندلی ها را با هم جمع کنیم. البته باز هم دو حالت برای اندیس پوینتر و بقیه ی کلاس ها به وجود می اید.

برای یافتن صندلی iام با شروع از pc ، همان روشی که برای []operator استفاده کردیم مورد استفاده قرار میدهیم.

در انتها نیز تابع getTemperature میانگین دماهای کلاسها را نمایش میدهدو که تابع دمای کلاس ها را با ورودی false صدا زدیم تا دمای رندم جدیدی به کلاس ها تعلق نگیرد.