گزارش کار

HW6

دانش عبداللهی

9723053

- سوال 1 :

تابع gradient_descent را به صورت template ئی تعریف می‌کنیم و ورودی سوم آن را به صورت پیشفرض برابر Object ئی از جنس خود تابع قرار می‌دهیم. برای گرفتن مشتق تابع در یک نقطه ، متغیری بسیار کوچک به نام delta تعریف می‌کنیم. همچنین از مقدار delta برای متوقف کردن الگوریتم استفاده می‌کنیم. به این صورت که اگر فاصلهٔ دو مقدار متوالی X از هم کمتر از مقدار delta باشد، الگوریتم را متوقف می‌کنیم و آخرین X را به عنوان خزوجی از تابع بامی‌گردانیم. نکتهٔ خاص دیگری در این سوال وجود ندارد و الگوریتم پیاده‌سازی به سادگی قابل فهم است.

- سوال 2 :

ابتدا کلاس Patient را می‌نویسیم و Constructor آنرا به صورت دلخواه می‌نویسیم (با 6 ورودی ، 5 متغیر کلاس را مقداردهی می‌کنیم.)

در تعریف تابع read_file ، ابتدا فایل csv. مورد نظر را تبدیل به یک String می‌کنیم. سپس یک الگوی مناسب برای پیدا کردن اطلاعات هر بیمار می‌نویسیم و با استفاده از

اطلاعات هر بیمار ، یک Object از کلاس Patient می‌سازیم و آنرا به Vector patients اضافه می‌کنیم. در آخر هم Vector patients را به عنوان خروجی تابع بازمی‌گردانیم.

در تعریف تابع sort ، ابتدا یک lambda function برای مقایسهٔ دو بیمار تعریف می-کنیم و با استفاده از آن و تابع ()std::sort ، Vector ورودی را مرتب می‌کنیم. (به ترتیب کاهشی در شانس سرطان)

***** توجه شود که توابع به صورت Static تعریف شده اند. در غیر این صورت با Multiple Definition Error مواجه می‌شدیم. چرا که تمام header ها ، هم در فایل main.cpp و هم در فایل unit_test.cpp اضافه شده اند. (برای حل این مشکل همچنین می‌توانستیم include های header ها در فایل main.cpp را کامنت کنیم.)**

- **سوال 3 :**

ابتدا کلاس Flight و Constructor آنرا تعریف می‌کنیم.

سپس یک تابع lambda برای مقایسهٔ دو Object از کلاس Flight تعریف می‌کنیم و از آن در تعریف خروجی تابع gather_flights و ... که به صورت priority_queue هستند ، استفاده می‌کنیم.

در تابع gather_flights ابتدا فایل txt. مورد نظر را تبدیل به String می‌کنیم و با استفاده از الگوی نوشته شده ، اطلاعات هر پرواز را بدست می‌آوریم. همچنین برای بدست آوردن تمام Connection_times ها برای هر پرواز از یک الگوی دیگر استفاده

می‌کنیم و از اطلاعات بدست‌آمده Duration time و Totall_Connection_time هر پرواز را به دقیقه حساب می‌کنیم.

در ادامه با استفاده از این اطلاعات بدست آمده ، یک Object از کلاس Flight می‌سازیم و آنرا در priority_queue flights می‌ریزیم.

*** توجه شود که توابع به صورت Static تعریف شده اند. در غیر این صورت با Multiple Definition Error مواجه می‌شدیم. چرا که تمام header ها ، هم در فایل main.cpp و هم در فایل unit_test.cpp اضافه شده اند. (برای حل این مشکل همچنین می‌توانستیم include های header ها در فایل main.cpp را کامنت کنیم.)

- سوال 4 :

ابتدا کلاس‌های Vector2D و Sensor همراه با Constrcutor هایشان تعریف می‌کنیم.

در تعریف تابع kalman_filter ابتدا یک Object از کلاس Vector2D می‌سازیم که در این تابع مقدار متغیرهایش را اصلاح می‌کنیم و به عنوان خروجی تابع برمی‌گردانیم.

در ادامه با استفاده از تابع std::for_each() رو تمام المان‌های ورودی (Vector sensors) می‌گردیم و مجموع تمام accuracy ها را حساب می‌کنیم سپس با استفاده از آن و دوبهره تابع std::for_each() میانگین وزن دار موقعیت‌های گزارش شدهٔ sensor ها را محاسبه می‌کنیم و در متغیرهای Object ساخته شده از نوع Vector2D می‌ریزیم.

*** توجه شود که توابع به صورت Static تعریف شده اند. در غیر این صورت با Multiple Definition Error مواجه می‌شدیم. چرا که تمام header ها ، هم در فایل

main.cpp و هم در فایل unit_test.cpp اضافه شده اند. (برای حل این مشکل همچنین می‌توانستیم include های header ها در فایل main.cpp را کامنت کنیم.)

در نهایت :

لینک github