



# گزارش کار

دانش عبداللہی

9723053

HW6

• سوال 1 :

تابع `gradient_descent` را به صورت `template` تعریف می کنیم و ورودی سوم آن را به صورت پیشفرض برابر `Object` ئی از جنس خود تابع قرار می دهیم. برای گرفتن مشتق تابع در یک نقطه ، متغیری بسیار کوچک به نام `delta` تعریف می کنیم. همچنین از مقدار `delta` برای متوقف کردن الگوریتم استفاده می کنیم. به این صورت که اگر فاصله دو مقدار متوالی `X` از هم کمتر از مقدار `delta` باشد، الگوریتم را متوقف می کنیم و آخرین `X` را به عنوان خزرجی از تابع بامی گردانیم. نکته خاص دیگری در این سوال وجود ندارد و الگوریتم پیاده سازی به سادگی قابل فهم است.

• سوال 2 :

ابتدا کلاس `Patient` را می نویسیم و `Constructor` آنرا به صورت دلخواه می نویسیم ( با 6 ورودی ، 5 متغیر کلاس را مقداردهی می کنیم. )

در تعریف تابع `read_file` ، ابتدا فایل `csv`. مورد نظر را تبدیل به یک `String` می کنیم. سپس یک الگوی مناسب برای پیدا کردن اطلاعات هر بیمار می نویسیم 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 باز می‌گردانیم.

در تعریف تابع sort ، ابتدا یک lambda function برای مقایسه دو بیمار تعریف می‌کنیم 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 شده‌اند. 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 شده‌اند. ( 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 مقایسه 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 بدست آمده `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 شده‌اند. 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 شده‌اند. ( 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 شده `sensor` ها را محاسبه می‌کنیم و در متغیرهای `Object` ساخته شده از نوع `Vector2D` می‌ریزیم.

\*\*\* توجه شود که توابع به صورت `Static` تعریف شده‌اند. 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 می توانستیم include های header ها در فایل main.cpp را کامنت کنیم. )

در نهایت :

```
root@b8347e4521a0:/usr/src/app/build# make && ./main
Scanning dependencies of target main
[ 33%] Building CXX object CMakeFiles/main.dir/src/main.cpp.o
[ 66%] Building CXX object CMakeFiles/main.dir/src/unit_test.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
RUNNING TESTS ...
[=====] Running 9 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 9 tests from HW6Test
[ RUN ] HW6Test.TEST1
[ OK ] HW6Test.TEST1 (0 ms)
[ RUN ] HW6Test.TEST2
[ OK ] HW6Test.TEST2 (0 ms)
[ RUN ] HW6Test.TEST3
[ OK ] HW6Test.TEST3 (0 ms)
[ RUN ] HW6Test.TEST4
[ OK ] HW6Test.TEST4 (0 ms)
[ RUN ] HW6Test.TEST5
[ OK ] HW6Test.TEST5 (4 ms)
[ RUN ] HW6Test.TEST6
[ OK ] HW6Test.TEST6 (4 ms)
[ RUN ] HW6Test.TEST7
[ OK ] HW6Test.TEST7 (7 ms)
[ RUN ] HW6Test.TEST8
[ OK ] HW6Test.TEST8 (0 ms)
[ RUN ] HW6Test.TEST9
[ OK ] HW6Test.TEST9 (0 ms)
[-----] 9 tests from HW6Test (16 ms total)

[-----] Global test environment tear-down
[=====] 9 tests from 1 test suite ran. (16 ms total)
[ PASSED ] 9 tests.
<<<SUCCESS>>>
root@b8347e4521a0:/usr/src/app/build# |
```



[لینک github](#)