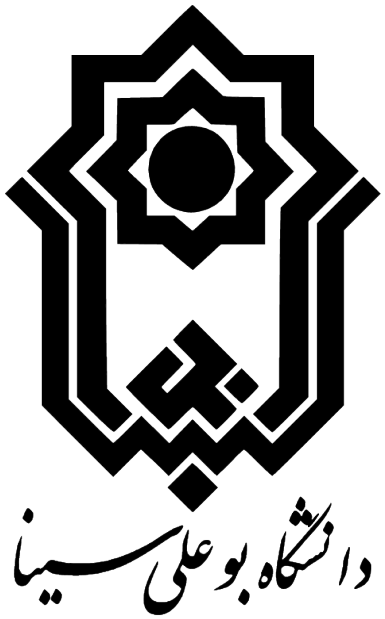

درس برنامه سازی پیشرفته

تمرین Classes and Operator Overloading

Matrix Class Implementation

بهار ۹۸

شرح کلی

امروزه ماتریس ها تقریبا در هر جایی کاربردی دارند.بسیاری از محاسباتی که در حل مسائل علمی امروز مطرح می شود با استفاده از ماتریس ها انجام می شود.با توجه به این موضوع تسهیل کار با ماتریس ها بسیار پراهمیت و مفید است. هدف این تمرین نوشتن یک کلاس برای کار با ماتریس ها است که بتوان محاسبات مرسوم ماتریسی را با استفاده از آن انجام داد.

جزئیات تمرین

در این تمرین شما باید کلاسی به نام Matrix بنویسید به طوریکه اعمالی که در ادامه خواسته شده با استفاده از آن قابل انجام باشد. تلاش کنید با استفاده صحیح از شی گرایی از دوباره نویسی کد خود پرهیز کنید.همچنین در توابعی که نیاز به پاس دادن یک ماتریس است، این کار را به نحوی انجام دهید که در هنگام ارسال ماتریس های بزرگ مشکل سرعت وجود نداشته باشد.

ساختار کلی

در فایلی به نام Matrix.h تعریف این کلاس و در فایلی به نام Matrix.cpp پیاده سازی این کلاس را انجام دهید.برای فایل Matrix.h هم include guard بنویسید.

ساخت ماتریس

کلاس Matrix را باید بتوان به صورت های زیر ساخت:

```
Matrix A(3, 2);
Matrix B(3, 2, 0.3);
Matrix C(5);
Matrix D = Matrix::eye(3);
Matrix E = Matrix::rand(4, 2);
array<array<float, 4>, 2> mat = {1, 2, 3, 4, 5, 6, 7, 8};
Matrix F(mat);
```

- A یک ماتریس 3×2 است که مقدار عناصر آن صفر است.
- B یک ماتریس 3×2 است که مقدار عناصر آن 0.3 است.
- در مواردی مثل ماتریس C که صرفا یک عدد داده شده است باید تعداد سطر ها و ستون ها را به اندازه همان عدد در نظر گرفت و مقدار اولیه عناصر را صفر در نظر گرفت.

- D یک ماتریس همانی 3×3 است.قطر اصلی این ماتریس 1 و دیگر عناصر این ماتریس 0 است.
- E یک ماتریس 2×4 است که عناصر آن اعداد تصادفی در بازه‌ی $[0, 1]$ هستند.
- F ماتریسی به صورت زیر است:

```
1 2 3 4
4 6 7 8
```

نمایش ماتریس

ماتریس باید با استفاده از output stream ها مثل cout قابل نمایش باشد.

```
1  Matrix A(3, 2, 0.4);
2  cout << A << endl;
```

```
0.4 0.4
0.4 0.4
0.4 0.4
```

دسترسی به عناصر ماتریس

برای کلاس توابعی به نام at و get پیاده سازی کنید که با دریافت دو عدد i و j عنصر متناظر در سطر i و ستون j را باز می گردانند.خروجی این توابع مانند کد زیر می تواند به عنوان L value هم استفاده شود.

```
1  cout << A.get(1, 2) << endl; // A[1][2]
2  cout << A.at(1, 1) << endl; // A[1][1]
3  A.at(1, 1) = 3; // A[1][1]
4  A.get(1, 1) = 2; // A[1][1]
```

تفاوت بین get و at در این است که اگر اعداد داده شده به at بزرگتر از اندازه ماتریس باشد آنگاه یک پیام خطا چاپ می شود. اما get بررسی نمی کند اندیس ورودی وجود دارد یا نه.

دسترسی به عناصر با عملگر پرانتز

می خواهیم با استفاده از عملگر پرانتز به شکلی معادل با تابع get به عناصر دسترسی داشته باشیم. مثلا می خواهیم به عنصر واقع در سطر 0 و ستون 3 دسترسی داشته باشیم:

```
cout << A(0, 3) << endl;
A(0, 3) = 7.5;
```

به دست آوردن زیر ماتریس

تابعی به نام rowRange تعریف کنید که با دریافت دو عدد i و j ماتریسی متشکل از سطر های i تا j ماتریس بازگرداند.همچنین تابعی به نام colRange تعریف کنید که با دریافت دو عدد i و j، ماتریسی متشکل از ستون های i تا j ماتریسی که روی آن فراخوانی شده است را بدست آورد.خروجی این توابع از نوع Matrix است.برای مثال فرض کنیم ماتریس A به صورت زیر باشد:

```
1 2 3 4
5 6 7 8
9 1 1 3
1 2 2 2
```

در اینصوت خواهیم داشت:

```
cout << A.rowRange(0, 2) << endl;
```

```
1 2 3 4
5 6 7 8
9 1 1 3
```

```
cout << A.colRange(1, 3) << endl;
```

```
2 3 4
6 7 8
1 1 3
2 2 2
```

دسترسی به خصوصیات کلاس

برای کلاس توابعی به نام rows و cols و size بنویسید که به ترتیب تعداد سطر ها، تعداد ستون ها و تعداد عناصر ماتریس را بازگرداند.

توابع محاسباتی کلاس

توابعی برای کلاس در نظر بگیرید که اعمال محاسباتی را به شکل زیر برای ماتریس ها ممکن کند. دقت کنید که در این محاسبات عملوند ها تغییر نمی کنند.

```
// Matrix Addition
Matrix C = A.add(B);
Matrix D = A + B;
// Matrix Subtraction
Matrix E = A.subtract(B);
Matrix F = A - B;
// Matrix Mutiplication
Matrix G = A.mul(B);
Matrix H = A * B;
// Matrix Power
Matrix I = A.power(3); // A * A * A
Matrix J = A ^ 3; // A * A * A
// Matrix Scalar Multiplication
// Multiple Every Element By A Number
Matrix K = A.mul(2);
Matrix L = A * 2;
```

در هر یک از این محاسبات در صورتی که اندازه‌ی عملوند ها متناسب نباشد، باید Exception پرتاب شود.

عملگر تساوی

عملگر تساوی را برای این کلاس overload کنید به طوریکه در صورت مساوی بودن اندازه و تمامی عناصر دو ماتریس مقدار true و در غیر این صورت false بازگرداند.

```
if(A == B)
    cout << "A = B" << endl;
else
    cout << "A != B" << endl;
```

ترانهاده ماتریس

عمل ترانهاده یک ماتریس را به دو شکل زیر پیاده سازی کنید.

```
// Matrix Transpose
cout << A.transpose() << endl;
cout << ~A << endl;
```

نمره‌ی مثبت

در صورت پیاده سازی این امکانات امتیاز اضافه برای تمرین محسوب خواهد شد و انجام آن به افراد علاقمند توصیه می شود.

تابع apply

برای کلاس ماتریس تابعی به نام apply بنویسید که با دریافت یک تابع، آن را بر روی تک تک عناص ماتریس اعمال کند.برای مثال اگر تابعی که خروجی آن توان دوم ورودی است را به apply پاس دهیم تمام عناصر ماتریس به توان 2 خواهند رسید و یا اگر تابعی که خروجی آن عدد 5 است را به آن پاس دهیم تمام عناصر برابر 5 خواهند شد.apply باید تابع موردنظر را به عنوان ورودی دریافت کند.به عنوان نمونه بعد از اجرای کد زیر تمام عناصر A باید در عدد 2 ضرب و با 1 جمع شوند.

```
A.apply([](float i){
    return 2*i+1;
});
```

دریافت ماتریس به شیوه خاص

کلاس را به نحوی تغییر دهید که بتوان عناصر را به شکل زیر وارد ماتریس کرد:

```
Matrix A(3, 3);
A << 1, 2, 3,
     4, 5, 6,
     7, 8, 9;
cout << A << endl;
```

```
1 2 3
4 5 6
7 8 9
```