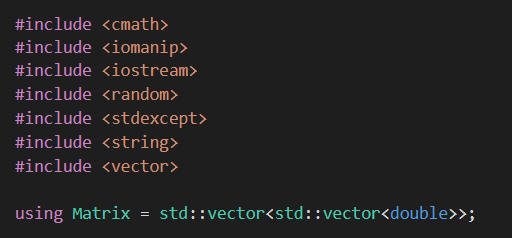


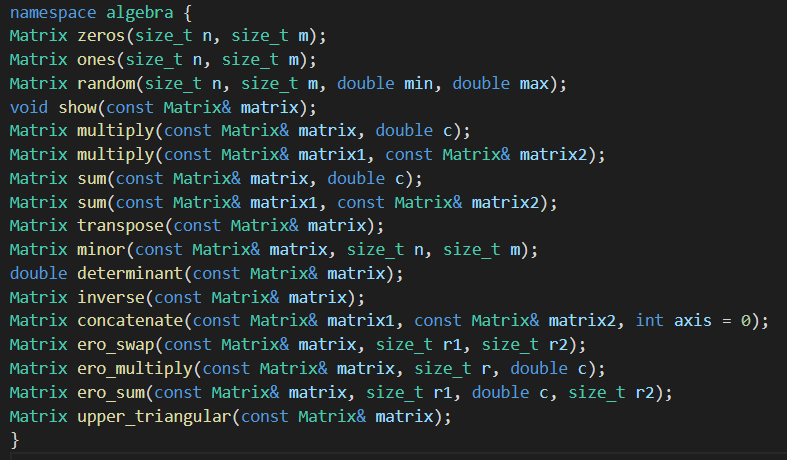
**گزارش­کار**

**دانش عبداللهی 9723053**

**HW1**

**در فایل hw1.h تمام کتاب­خانه­های مورد نیاز را اضافه می­کنیم. ( البته این اضافه کردن کتاب­خانه­ها به مرور با تکمیل شدن کد صورت گرفته است. )**

**همچنین نوع متغیر Matrix را با استفاده از vector ها مشخص می­کنیم.**

**همچنین در این فایل declarations های تمام توابع را در namespace algebra وارد می­کنیم.**

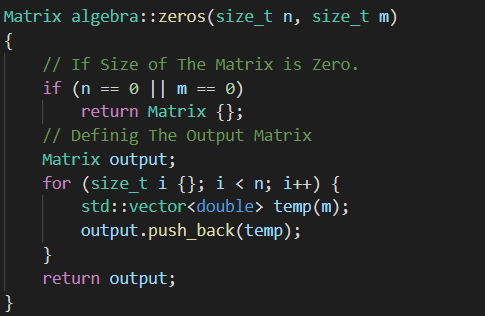
**در ادامه تعاریف تمام توابع را در فایل hw1.cpp می­نویسیم.**

1. **تابع zeros**

**برای ساختن یک ماتریس n\*m به طوری که تمام درایه­های آن صفر باشند، n تا بردار mتائی صفر می­سازیم و آنرا در متغیر به نوع Matrix ( که آن در hw1.h تعریف کرده بودیم ) می­ریزیم.**

**ورودی این تابع، دو عدد صحیح نامنفی و خروجی آن یک ماتریس n\*m با درایه­های تمام صفر است.**

**اگر هر کدام از ابعاد ماتریس ورودی صفر باشد ، با logic error مواجه می­شویم.**

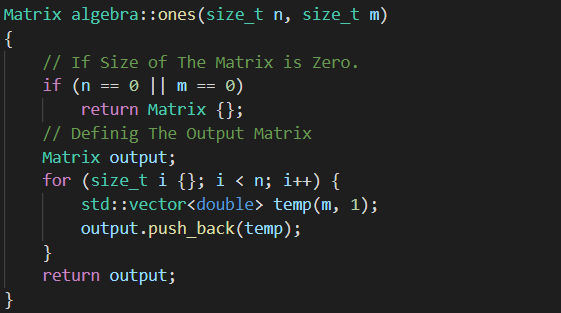
****

1. **تابع ones**

**برای ساختن یک ماتریس n\*m به طوری که تمام درایه­های آن یک باشند، n تا بردار mتائی با درایه­های تمام یک می­سازیم و آنرا در متغیر به نوع Matrix می­ریزیم.**

**ورودی این تابع، دو عدد صحیح نامنفی و خروجی آن یک ماتریس n\*m با درایه­های تمام صفر است.**

**اگر هر کدام از ابعاد ماتریس ورودی صفر باشد ، با logic error مواجه می­شویم.**

****

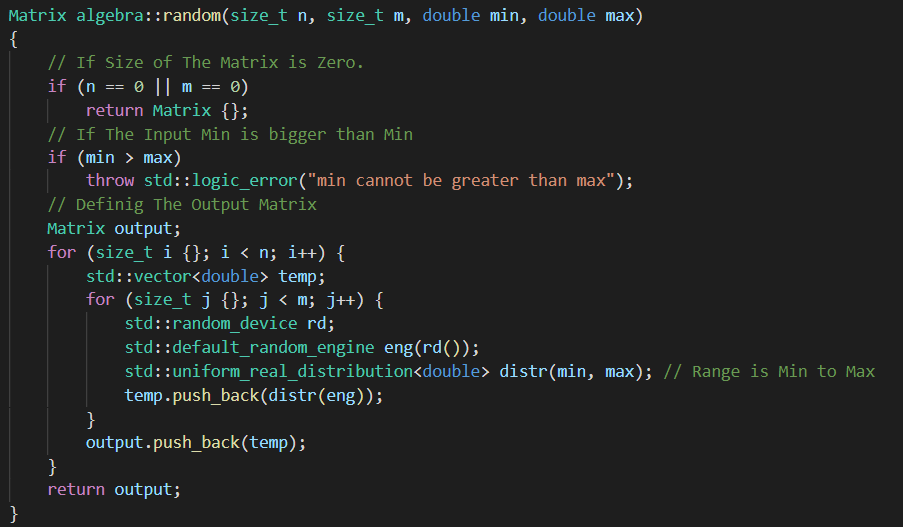
1. **تابع random**

**برای ساختن یک ماتریس n\*m به طوری که تمام درایه­های آن یک عدد رندوم بین min و max ( ورودی­های تابع ) باشند، n تا بردار mتائی با درایه­های تمام یک می­سازیم و آنرا در متغیر به نوع Matrix می­ریزیم.**

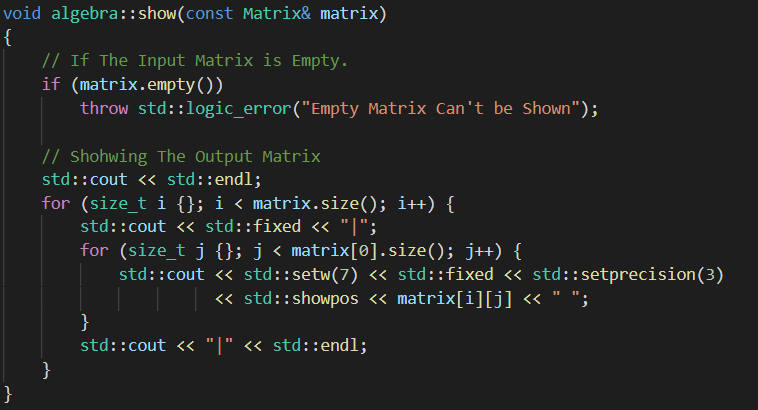
**برای ساختن اعداد رندوم در بازۀ [min : max] ، از توابع کتاب­خانۀ random استفاده می­کنیم.**

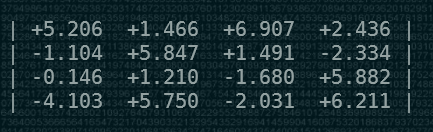
**ابتدا یک عدد به عنوان seed ، random engine می­سازیم و سپس با استفاده از توزیع احتمال یکنواخت در بازۀ معلوم ، یک عدد رندوم double در این بازه می­سازیم.**

**در این تابع، اگر ورودی min از ورودی max بیشتر باشد یا اگر هرکدام از ابعاد ماتریس ورودی صفر باشد، با logic error مواجه می­شویم.**

****

1. **تابع show**

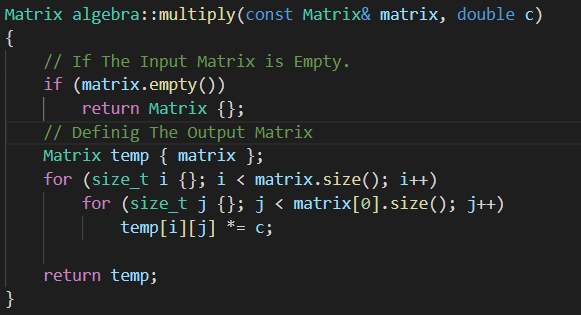
**در این تابع ، ماتریس ورودی به تابع را با دو حلقۀ for تو در تو چاپ می­کنیم. ( این تابع بدون خروجی است. )**

**نمونۀ خروجی این تابع به صورت زیر است :**

**در این تابع، اگر ماتریس ورودی تهی باشد، با logic error مواجه می­شویم.**

1. **تابع multiply**

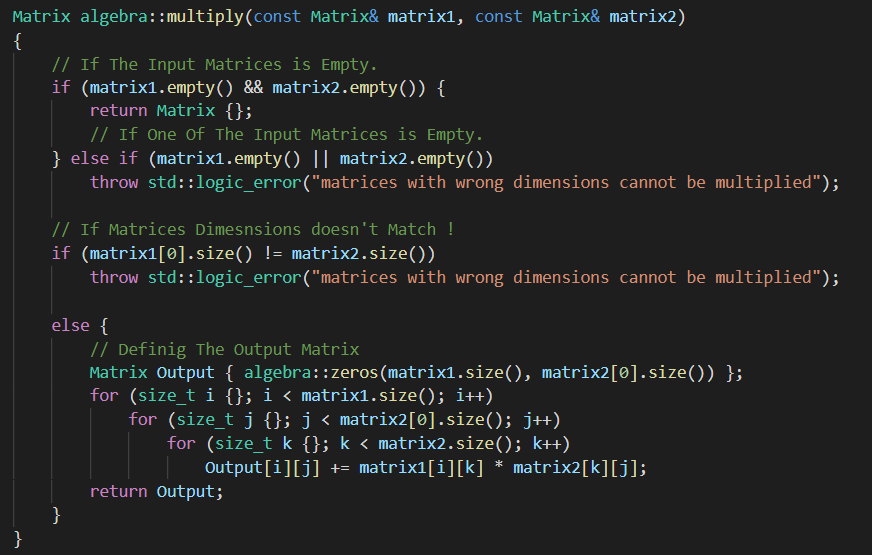
**این تابع ماتریس ورودی را در عدد ورودی ضرب می­کند. یعنی تمام درایه­های ماتریس ورودی را در عدد معلوم ضرب می­کند. برای اینکار در ابتدای تابع، ماتریس ورودی را در یک متغیر temp از نوع Matrix تعریف می­کنیم. زیرا ماتریس ورودی فقط قابل خواندن است.**

****

**در این تابع، اگر ماتریس ورودی تهی باشد، با logic error مواجه می­شویم.**

1. **تابع multiply**

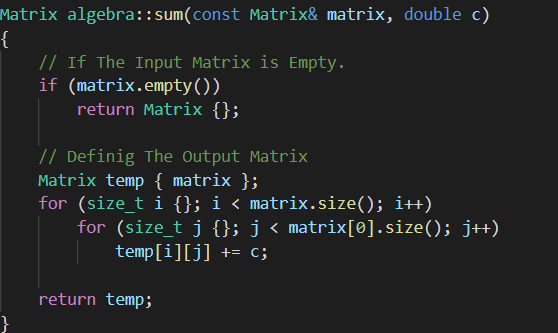
**این تابع، دو ماتریس ورودی را در هم ضرب ماتریسی می­کند و به عنوان خروجی تحویل می­دهد.**

****

**در این تابع، اگرهر دو ماتریس ورودی تهی باشند، خروجی تابع ماتریس تهی است اما اگر فقط یکی از ماتریس­های ورودی تهی باشد، یا ابعاد ماتریس­های ورودی برای ضرب ماتریسی با هم ، هم­خوانی نداشته باشند ، با logic error مواجه می­شویم.**

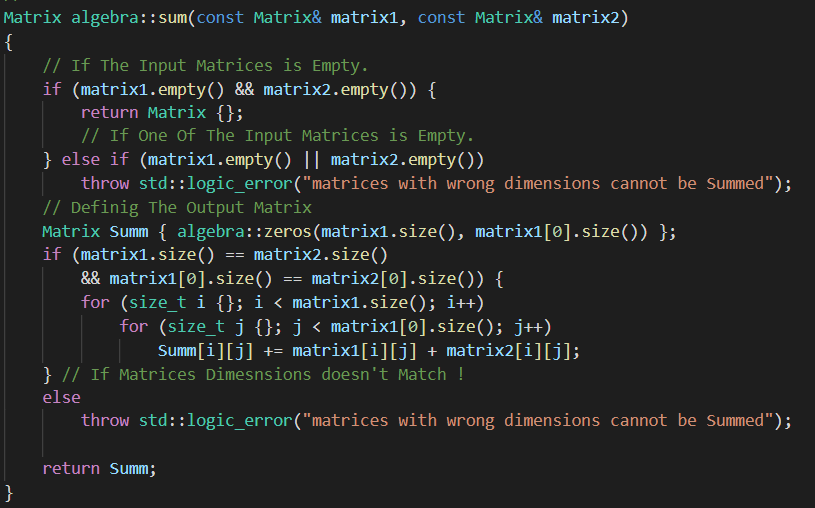
1. **تابع sum**

**این تابع ماتریس ورودی را با عدد ورودی جمع می­کند. یعنی تمام درایه­های ماتریس ورودی را با عدد معلوم جمع می­کند. برای اینکار در ابتدای تابع، ماتریس ورودی را در یک متغیر temp از نوع Matrix تعریف می­کنیم. زیرا ماتریس ورودی فقط قابل خواندن است.**

****

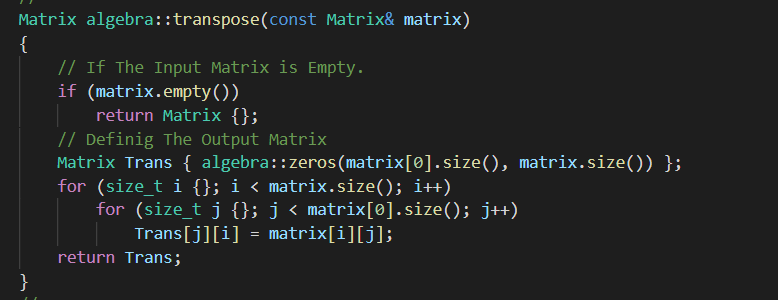
**در این تابع، اگر ماتریس ورودی تهی باشد، خروجی تابع ماتریس تهی است .**

1. **ماتریس sum**

**این تابع دو ماتریس ورودی را با هم جمع ماتریسی می­کند. یعنی درایه­های نظیر به نظیر هر دو ماتریس را با هم جمع می­کند.**

**در این تابع، اگرهر دو ماتریس ورودی تهی باشند، خروجی تابع ماتریس تهی است اما اگر فقط یکی از ماتریس­های ورودی تهی باشد، یا ابعاد ماتریس­های ورودی برای جمع ماتریسی با هم ، هم­خوانی نداشته باشند ، با logic error مواجه می­شویم.**

1. **تابع transpose**

**خروجی این تابع، transpose شدۀ ماتریس ورودی است. اگر ماتریس ورودی n\*m باشد ، ماتریس خروجی m\*n خواهد بود.**

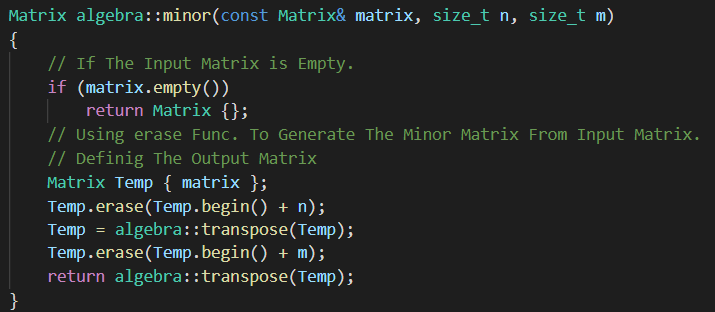
**در این تابع ، اگر ماتریس ورودی تهی باشد، ماتریس خروجی تهی خواهد بود.**

1. **تابع minor**

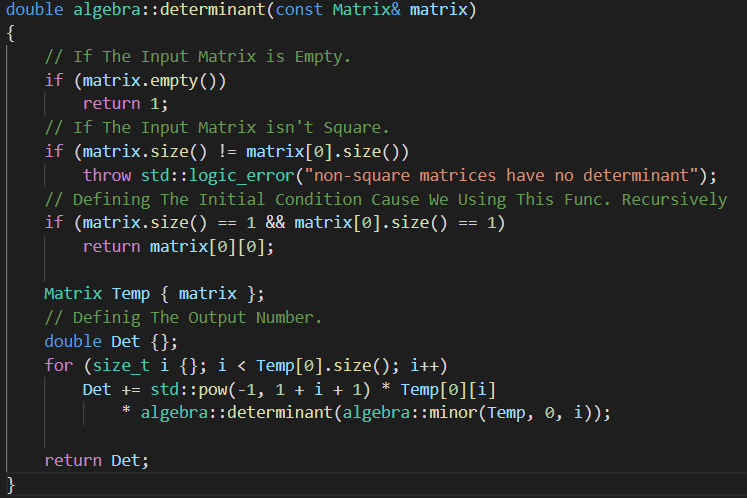
**در این تابع، ابتدا ماتریس ورودی را در یک متغیر temp از نوع Matrix میریزیم.**

**سپس با استفاده از تابع erase از کتابخانۀ vector ، سطر nام ورودی را پاک می­کنیم. سپس temp را transpose می­کنیم و سطر mام را پاک می­کنیم و در مرحلۀ آخر دوباره ماتریس temp ، را transpose می­کنیم و به عنوان خروجی تحویل می­دهیم.**

**در این تابع، اگر ماتریس ورودی تهی باشد، خروجی تابع ماتریس تهی است .**

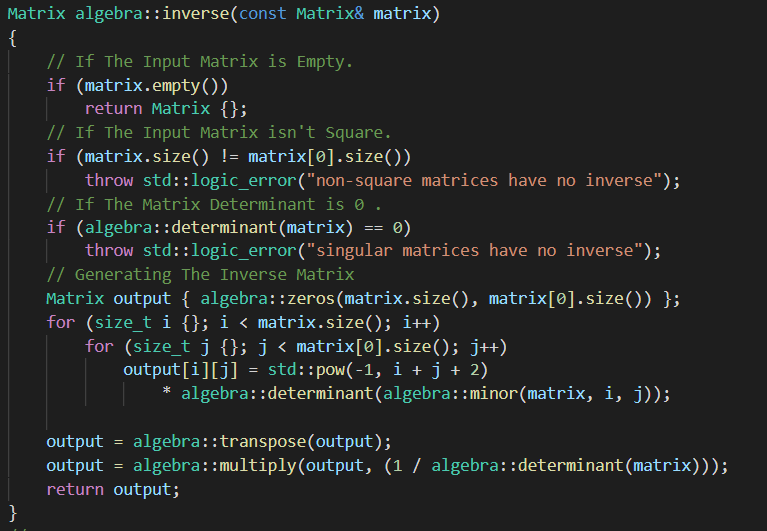
****

1. **تابع determinant**

**این تابع دترمینان ماتریس ورودی را به عنوان خروجی تحویل می­دهد. در تعریف این تابع، خود تابع را فرا می­خوانیم. ( تابع بازگشتی است. ) برای اینکه تابع بازگشتی بنویسیم ، یک شرط پایانی برای تعریف تابع می­نویسیم. ( دترمینان ماتریس 1\*1 برابر با خودش است. )**

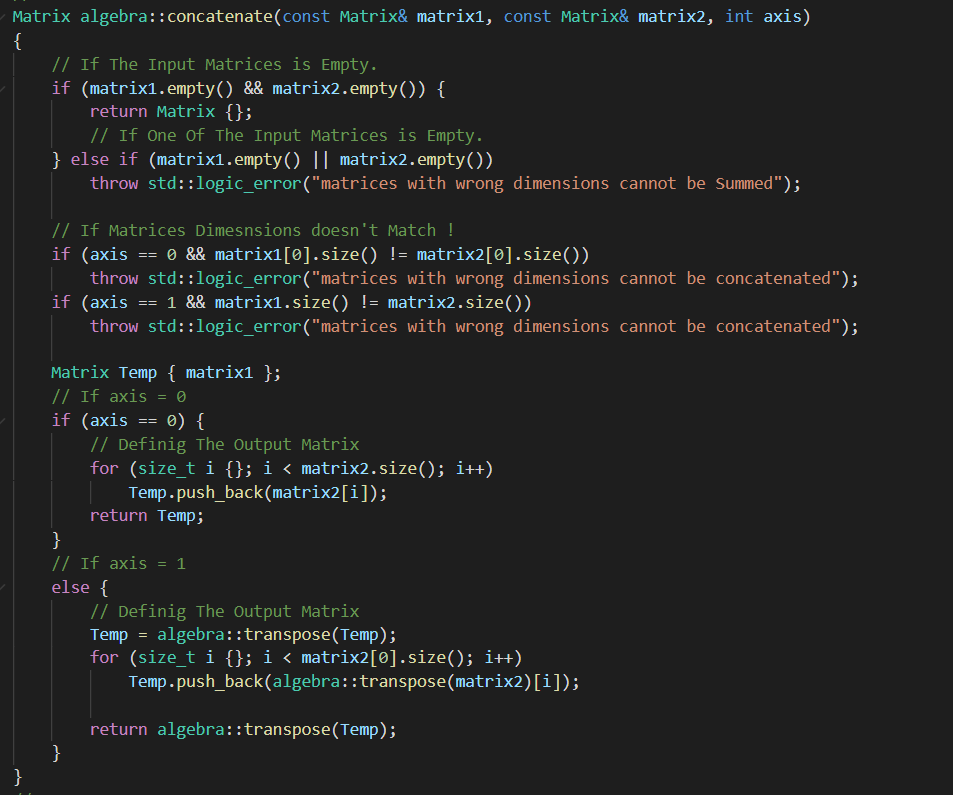
**در این تابع اگر ماتریس ورودی تهی باشد، خروجی برابر با عدد 1 است. همچنین اگر ماتریس ورودی مربعی نباشد، با logic error مواجه می­شویم.**

1. **تابع inverse**

**خروجی این تابع ، inverse ماتریس ورودی است. در این تابع، از تابع­های قبلی ( determinant و minor ) استفاده می­کنیم.**

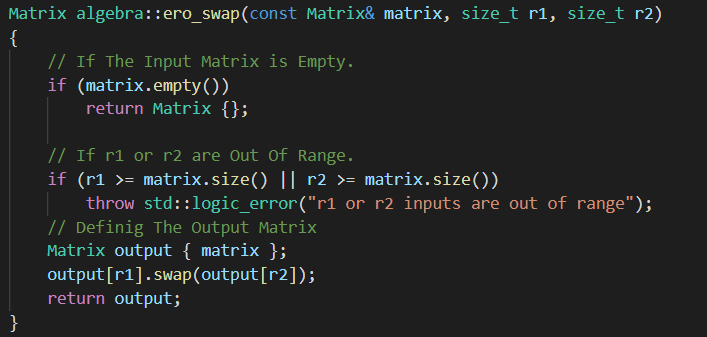
**در این تابع، اگر ماتریس ورودی مربعی نباشد یا دترمینان ماتریس ورودی 0 باشد ، با logic error مواجه می­شویم. همچنین اگر ماتریس ورودی تهی باشد ، خروجی ماتریس تهی خواهد بود.**

1. **تابع concatenate**

**این تابع ، دو ماتریس ورودی را از کنار یا از زیر، به هم دیگر می­چسباند.اگر ورودی axis تابع ، 0 باشد ، ماتریس2 را از راست به ماتریس1 می­چسباند و اگر ورودی axis تابع ، 1 باشد ، ماتریس2 را از پایین به ماتریس1 می­چسباند.**

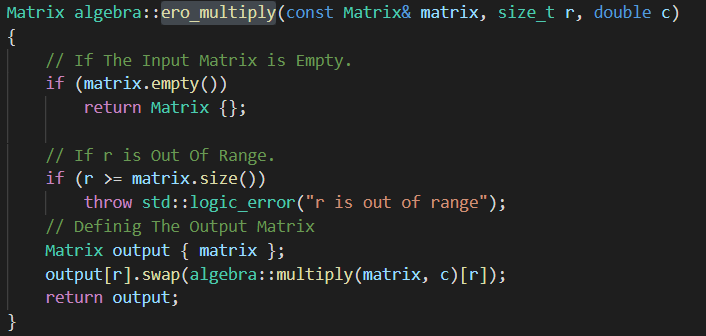
**در این تابع، اگرهر دو ماتریس ورودی تهی باشند، خروجی تابع ماتریس تهی است اما اگر فقط یکی از ماتریس­های ورودی تهی باشد، یا ابعاد ماتریس­های ورودی برای به هم چسباندن در راستای افقی یا عمودی ، هم­خوانی نداشته باشند ، با logic error مواجه می­شویم.**

1. **تابع ero\_swap**

**در این تابع با استفاده از تابع swap از کتاب­خانۀ vector ، دو سطر r1ام و سطر r2ام را با هم عوض می­کند.**

**در این تابع اگر ماتریس ورودی تهی باشد، خروجی تابع ماتریس تهی است. همچنین اگر ورودی r1 و r2 از سایز ابعاد ماتریس ( تعداد سطرهای ماتریس ) بیشتر باشد، با logic error مواجه می­شویم.**

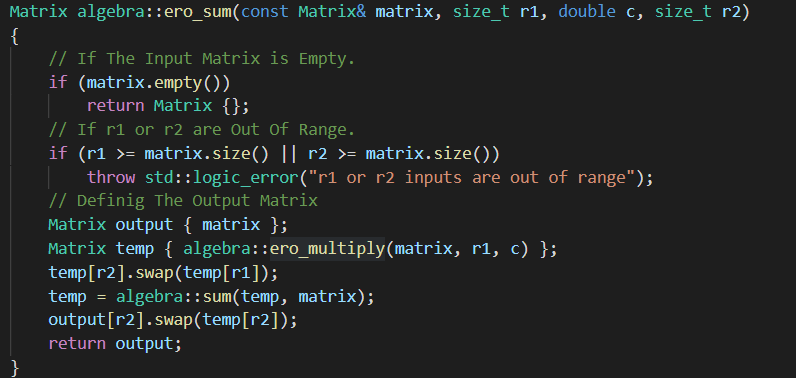
1. **تابع ero\_multiply**

**در این تابع ، فقط سطر rام ماتریس ورودی را در عدد c ضرب می­کنیم.**

**ابتدا ماتریس ورودی را در یک متغیر temp از نوع Matrix می­ریزیم و ماتریس ورودی را در عدد c ضرب می­کنیم. سپس با استفاده از تابع swap، سطر rام ماتریس ضرب شده را در سطر rام متغیر temp می­ریزیم.**

**در این تابع اگر ماتریس ورودی تهی باشد، خروجی تابع ماتریس تهی است. همچنین اگر ورودی r از سایز ابعاد ماتریس ( تعداد سطرهای ماتریس ) بیشتر باشد، با logic error مواجه می­شویم.**

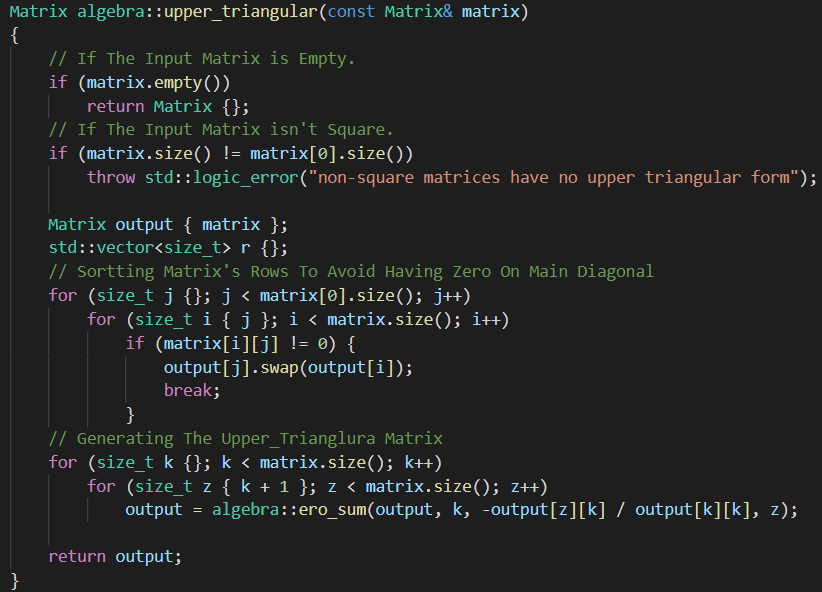
1. **تابع ero\_sum**

** این تابع، سطر r1ام ماتریس ورودی را در عدد c ضرب می­کند و آنرا با سطر r2ام ماتریس جمع می­کند.ابتدا با استفاده از تابع ero\_multiply سطر r1ام ماتریس را در عدد c ضرب می­کنیم. سپس سطر r1ام آنرا با سطر r2ام جابه­جا می­کنیم و بعد ماتریس بدست آمده را با ماتریس ورودی تابع جمع می­کنیم. در نهایت سطر r2ام ماتریس بدست آمده را در سطر r2ام ماتریس ورودی جایگزین می­کنیم.**

**در این تابع اگر ماتریس ورودی تهی باشد، خروجی تابع ماتریس تهی است. همچنین اگر ورودی r1 و r2 از سایز ابعاد ماتریس ( تعداد سطرهای ماتریس ) بیشتر باشد، با logic error مواجه می­شویم.**

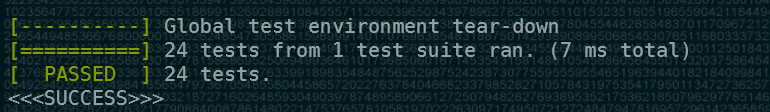
1. **تابع upper\_triangular**

**در این تابع ابتدا با جابه­جا کردن سطر­های ماتریس ورودی، ماتریس ورودی را به ماتریس تبدیل می­کنیم که هیچ کدام از درایه­های قطر اصلی آن صفر نباشد. ( اگر نیاز باشد! )**

**سپس با اعمال عملیات­های سطری مقدماتی ، ماتریس بالا مثلثی را می­سازیم.**

**در این تابع اگر ماتریس ورودی تهی باشد، خروجی ماتریس تهی است. همچنین اگر ماتریس ورودی مربعی نباشد، با logic error مواجه می­شویم.**

**در آخر هم برنامه را اجرا می­کنیم و خروجی به صورت زیر است :**

****