مدرس: رامتین خسروی موعد تحویل: ۹۰/۱۲/۲۲

حل N معادله با N مجهول

این پروژه از چند بخش تشکیل شده است و پیشنهاد می‌شود ابتدا یک‌بار کل آن را مطالعه کرده و سپس بخش‌ها را به ترتیب انجام داده و تست کنید. در نهایت پروژه‌ی شما می‌تواند جواب N معادله با N مجهول را، در صورت وجود بیابد.

بخش اول- نوشتن کتابخانه‌ای برای عملیات ماتریسی:

هدف از این بخش نوشتن یک کتابخانه برای انجام عملیاتی است که می‌توان روی ماتریس‌ها انجام داد. هر ماتریس به صورت یک بردار[1] دو بعدی از اعداد اعشاری تعریف می‌شود. شما باید توابع زیر را پیاده‌سازی کنید:

تابع

vector<vector<double> > add(const vector <vector<double> > &A,
const vector<vector<double> > &B)

این تابع باید ماتریس‌های A و B را با یکدیگر جمع کند. فرض کنید تعداد سطرها و ستون‌های ماتریس‌های A و B برابر هستند.

تابع

vector<vector<double> > multiply(const vector <vector<double> > &A,
const vector<vector<double> > &B)

این تابع ضرب دو ماتریس را برمی‌گرداند. ضرب دو ماتریس در حالتی تعریف شده است که A ماتریسی $m \times n$ و b ماتریسی $n \times k$ باشد. خروجی این تابع هم یک ماتریس $m \times k$ خواهد بود. برای محاسبه‌ی درایه‌ای که در سطر i و ستون j وجود دارد کافی است سطر i از ماتریس A را درایه به درایه در ستون j از ماتریس B ضرب کرده و حاصل آن‌ها را با یک‌دیگر جمع کنیم.

تابع

vector<vector<double> > scalar_multiply(double scale, const vector< vector<double> > &M)

این تابع یک عدد اعشاری را در یک ماتریس ضرب می‌کند. حاصل آن، ماتریسی با ابعاد ماتریس M خواهد بود که درایه‌های آن $scale$برابر درایه‌های M است.

[1] Vector

تابع

void row_operation (vector< vector<double> > &M, int i, int j, double scale)

این تابع $scale$برابر سطر iام ماتریس M را به سطر jام اضافه می‌کند. (دقت کنید که این تابع خروجی ندارد و تغییرات روی ماتریس داده‌شده اعمال می شود.)

تابع

double determinant(const vector< vector<double> > &M)

این تابع دترمینان ماتریس M را محاسبه می‌کند. فرض کنید ورودی یک ماتریس مربعی است. برای محاسبه‌ی دترمینان با استفاده از تابع row_operation (با روشی که در ادامه توضیح داده‌شده) ماتریس را قطری‌سازی کنید. دترمینان ماتریس حاصل، که برابر دترمینان ماتریس اولیه است، برابر با حاصلضرب درایه‌های قطر اصلی است.

تابع

vector<vector<double> > inverse(const vector< vector<double> > &M)

این تابع، معکوس ماتریس مربعی M را محاسبه خواهد کرد. برای این کار از روش زیر استفاده کنید:

فرض کنید M یک ماتریس $n \times n$ باشد. ماتریس $n \times 2n$ی با نام R به این شکل می‌سازیم که n ستون اول آن همان ماتریس M باشد و n ستون بعدی آن ماتریس همانی (مطابق شکل زیر). ثابت می‌شود که اگر با انجام عملیات row_operation بر روی ماتریس R ماتریس R' حاصل شود که n ستون اول آن ماتریس همانی باشد، آنگاه ماتریس $n \times n$ی که معادل n ستون دوم ماتریس R' است، ماتریس معکوس M خواهد بود.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{bmatrix}$$

بخش دوم – حل N معادله با N مجهول با استفاده از دو روش متفاوت

در این بخش قصد داریم با استفاده از کتابخانه‌ای که در بخش اول نوشته‌ایم N معادله با N مجهول را حل کنیم. فرض کنید که معادلات، دقیقاً یک جواب دارند.

روش اول (ماتریس معکوس)

vector <double> solve_inverse(const vector< vector<double> > &A, const vector <double> &B)

برای حل N معادله با N مجهول، از کتابخانه‌ای که در بخش اول نوشتید استفاده خواهیم کرد. معادلات زیر را در نظر بگیرید:

$$\begin{aligned} &a_{11}X_1 + a_{12}X_2 + a_{13}X_3 + \cdots + a_{1N}X_N = b_1 \\ &a_{21}X_1 + a_{22}X_2 + a_{23}X_3 + \cdots + a_{2N}X_N = b_2. \\ &\vdots \\ &\vdots \\ &a_{N1}X_1 + a_{N2}X_2 + a_{N3}X_3 + \cdots + a_{NN}X_N = b_N \end{aligned}$$

معادلات فوق را می‌توان به صورت $AX = B$ نیز نوشت:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

اگر دو طرف معادله‌ی فوق را در معکوس ماتریس A ضرب کنیم، داریم:

$$A^{-1} * (AX) = A^{-1}B \xrightarrow{A^{-1}A=I} X = A^{-1}B$$

روش دوم (گاوس-جردن)

vector <double> solve_GJ(const vector< vector<double> > &A, const vector <double> &B)

ماتریسی با N سطر و N+1 ستون به صورت زیر در نظر بگیرید:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} & b_1 \\ a_{21} & a_{22} & \dots & a_{2N} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} & b_N \end{bmatrix}$$

در این روش می‌خواهیم با یک سری اعمال بر روی ماتریس فوق، آن را به ماتریسی به صورت زیر تبدیل کنیم:

$$\begin{bmatrix} A_{11} & 0 & \cdots & 0 & B_1 \\ 0 & A_{22} & \dots & 0 & B_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_{NN} & B_N \end{bmatrix}$$

اگر بتوانیم به ماتریس فوق برسیم، تمامی X_iها از رابطه‌ی $X_i = \frac{B_i}{A_{ii}}$ به دست می‌آیند. حال برای به دست آوردن ماتریس فوق باید الگوریتم زیر را پیاده‌سازی کنید.
ابتدا باید همه‌ی A_{1i}ها($i > 1$) را صفر کنید. برای این کار کافی است سطر i را با ضریبی از سطر اول جمع کنیم. (این ضریب برابر با $\frac{-A_{1i}}{A_{11}}$ است.) پس از آن، نوبت به A_{2i}ها($i > 2$) می‌رسد و باید آن‌ها را صفر کنید که روش آن همانند فوق است. با ادامه‌ی این کار به ماتریسی می‌رسیم که تمام درایه‌های زیر قطر اصلی آن صفر است. اگر همین کار را (این بار از سطر آخر) برای درایه‌های بالای قطر اصلی هم انجام دهیم به ماتریس فوق می‌رسیم.

فرض کنید می‌خواهیم سه معادله با سه مجهول زیر را با این روش حل کنیم:

$$\begin{cases} X_1 + 2X_2 + 3X_3 = 5 \\ 2X_1 + X_1 + 5X_3 = 4 \\ X_1 + X_2 + X_3 = 3 \end{cases}$$

بنابراین ماتریس ذکر شده را تشکیل می‌دهیم و عملیات فوق را انجام می‌دهیم:

$$\begin{bmatrix}1&2&3&5\\2&1&5&4\\1&1&1&3\end{bmatrix}\xRightarrow{row[2]-=2row[1]}\begin{bmatrix}1&2&3&5\\0&-3&-1&-6\\1&1&1&3\end{bmatrix}\xRightarrow{row[3]-=row[1]}\begin{bmatrix}1&2&3&5\\0&-3&-1&-6\\0&-1&-2&-2\end{bmatrix}$$

$$\xRightarrow{row[3]-=\frac{1}{3}row[2]}\begin{bmatrix}1&2&3&5\\0&-3&-1&-6\\0&0&-\frac{5}{3}&0\end{bmatrix}\xRightarrow{row[2]-=\frac{3}{5}row[3]}\begin{bmatrix}1&2&3&5\\0&-3&0&-6\\0&0&-\frac{5}{3}&0\end{bmatrix}\xRightarrow{row[1]+=\frac{9}{5}row[3]}\begin{bmatrix}1&2&0&5\\0&-3&0&-6\\0&0&-\frac{5}{3}&0\end{bmatrix}$$

$$\xRightarrow{row[1]+=\frac{2}{3}row[2]}\begin{bmatrix}1&0&0&1\\0&-3&0&-6\\0&0&-\frac{5}{3}&0\end{bmatrix}\xRightarrow{X_i=\frac{B_i}{A_{ii}}}\begin{cases}X_1=\frac{1}{1}=1\\X_2=\frac{-6}{-3}=2\\X_3=\frac{0}{-\frac{5}{3}}=0\end{cases}$$

بخش سوم – خواندن ورودی و چاپ کردن نتیجه:

در این بخش باید تابعی با امضای readfile(string filename, string method) بنویسید که پارامتر اول آن نام فایل ورودی و پارامتر دوم آن مشخص کنندهی روش حل دستگاه معادلات است. پارامتر دوم یکی از دو مقدار "Inverse" یا "Gauss-Jordan" را خواهد داشت.

سطر اول فایل ورودی حاوی عدد صحیح و مثبت N است که بعد از آن، N سطر که هر کدام شامل N عدد اعشاری است نوشته شده است. این اعداد نشان‌دهندهی ماتریس A هستند. سپس N سطر که هر کدام حاوی یک عدد اعشاری است به عنوان ماتریس B داده می‌شود. $(AX = B)$

شما باید خروجی برنامه که همان ماتریس X است را در N سطر که هر سطر آن حاوی یک عدد اعشاری (با دقیقاً دو رقم اعشار) است، در خروجی استاندارد چاپ کنید.

خروجی نمونه	ورودی نمونه
1.00 2.00 0.00	3 1 2 3 2 1 5 1 1 1 5 4 3

نحوه‌ی تحویل

شما باید کدی که برای هر کدام از بخش‌ها می‌نویسید را به شکل دو فایل، شامل یک فایل منبع و یک هدر فایل بنویسید. نام این فایل‌ها برای بخش‌های مختلف باید به این شکل باشد:

بخش اول: matrix.h و matrix.cpp. بخش دوم: equationsys.h و equationsys.cpp. بخش سوم: read.h و read.cpp.

دقت کنید که هیچ کدام از این فایل‌ها نباید تابع ()main داشته‌باشند. ما برای آزمودن برنامه‌ی شما از یک فایل که فایل هدر مورد نظر را include نموده و دارای تابع ()main باشد، استفاده کرده و این فایل را در کنار فایل‌های منبع شما کامپایل و لینک می‌کنیم.

شایان ذکر است که شما باید برای کامپایل کردن فایل‌های مختلف تمرین خود از make استفاده کرده و Makefile مربوطه را همراه با تمرین‌تان آپلود کنید.

تحویل این تمرین به صورت حضوری خواهد بود. البته شما باید در مهلت مشخص‌شده، فایل‌های خود را در قالب یک فایل فشرده‌شده با نام A3-SID.zip در مکان مناسب در سایت درس آپلود کنید. (SID، پنج رقم آخر شماره‌ی دانشجویی شما است. مثلاً اگر شماره دانشجویی شما ۸۱۰۱۹۰۱۲۳ است، نام فایل شما باید A3-90123.zip باشد.)

دقت کنید

- برنامه‌های شما باید به زبان ++C باشند و با کامپایلر ++g کامپایل شوند.
- به جز کتابخانه‌های استاندارد زبان ++C از کتابخانه‌ی دیگری استفاده ننمایید.
- سعی کنید با شکستن متن برنامه به توابع مناسب به برنامه‌ی خود نظم دهید. همچنین اسامی متغیرها و توابع داخلی را متناسب با کاربرد آن‌ها انتخاب کنید.
- استفاده از توابع زبان C، آرایه، اشاره‌گر، عملیات cast مجاز نیست. همچنین نباید این تمرین را به صورت شیءگرا مجاز بنویسید.