מבוא לתכנות מערכות
תרגיל בית 3

סמסטר חורף
2019-2018

תאריך פרסום: 24/12/18

תאריך הגשה: 14/01/19

משקל התרגיל: 12% מהציון הסופי (תקף)

מתרגל אחראי: אור אייזקס.

<u>הערות כלליות:</u>

- **שימו לב: לא יינתנו דחיות במועד התרגיל. תכננו את הזמן בהתאם.**
- לשאלות בנוגע להבנת התרגיל יש לפנות לסדנאות של התרגיל, או לשאול בפורום של הקורס במודל. לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר בFAQ - או במודל, ושהתשובה אינה ברורה ממסמך זה, מהדוגמא ומהבדיקות שפורסמו עם התרגיל.
- **קראו מסמך זה עד סופו לפני שאתם מתחילים לממש. יתכן שתצטרכו להתאים את המימוש שלכם לחלק עתידי בתרגיל. תכננו את המימוש שלכם לפני שאתם ניגשים לעבוד.**
- **חובה** להתעדכן בעמוד ה-FAQ של התרגיל.
- **העתקות בתוכנה תטופלנה בחומרה.**
- **אלא אם צוין אחרת, ניתן להניח שהמטריצות שמשומשות בתרגיל יהיו בגדלים של עשרות עמודות ושורות. כך שכמות הזיכרון הנדרשת היא בהתאם.**
- מומלץ לכתוב את המימוש של התבניות בקובץ ה-header, ניתן לבצע אותם גם כקובץ cpp אך זה דורש צעדים נוספים.

- מומלץ לעשות מימוש ראשוני עבור סוג ספציפי של משתנה. למשל לכתוב typedef int T.
  ורק כאשר הקוד מתקמפל ועובר טסטים להוריד את השורה ולהוסיף הגדרת תבנית.
- לתרגיל מצורף קובץ main.cpp, יש לוודא שכל הפונקציות שיצרתם מתקמפלות עם קובץ זה ומוציאות פלט זהה לקובץ main_out המצורף.
- ניתן להשתמש במבני STL בתרגיל.
- **אי שמירה על Code Conventions יגררו הורדה בציון.**

רקע:

אחרי מאמצים רבים הצלחתם להתקבל לעבודה בחברת מת"ם (מתמטיקה תכנותית מתקדמת). העבודה הראשונה שלכם בחברה היא בניית מערכת מתמטית לעבודה עם מטריצות דו מימדיות ווקטורים שאפשר יהיה להשתמש בה בפרויקטים אחרים.

המערכת המתמטית תורכב ממחלקות של וקטורים ומטריצות. כאשר המחלקות צריכות לעבוד בצורה של תבניות עם סוגי משתנים של float, double, int ו-Complex- מחלקה של משתנים מרוכבים שתצטרכו לממש.

בנוסף למטריצות רגילות ווקטורים, מבקשים ממכם לייצר באמצעות יכולת ההורשה ב-c++ מחלקות עבור מטריצות ריבועיות ומטריצות אלכסוניות ריבועיות שיירשו ממחלקת מטריצות רגילות.

חלק רטוב:

כל החלק הרטוב צריך להיכלל תחת namespace בשם MtmMath.

מסופק לכם קובץ בשם Auxilaries שמכיל מחלקה בשם Dimensions, מחלקה זו מכילה זוג מספרים המייצגים מימדי מטריצות ווקטורים וכמו כן פונקציות שימושיות.

מחלקת Complex

מחלקה זו מתארת מספרים מרוכבים כפי שלמדנו בקורס.

הבנאי שלהם צריך להיות: Complex(double re=0, double im=0)

יש לממש כל אופרטור הנדרש בשביל תרגיל זה.

מחלקת MtmMat

מחלקה זו מתארת מטריצה בסיסית.

הפונקציות הנדרשות ממחלקה זו בלבד:

i. בנאי:
mtmMat(Dimensions dim_t, const T& val=T())
מייצר מטריצה במימד של dim_t. הערך Val הוא הערך איתו מאתחלים את המטריצה.

ii. בנאי העתקה, הורס והשמה.

iii. void reshape(Dimensions newDim)
משנה את גודל המטריצה לגודל חדש כך שמספר האיברים במטריצה נשמר, ומיקום כל איבר מבחינת אינדקס לינארי נשמר (הגדרה של אינדקס לינארי והדגמה של פעולת זו נמצאים בסוף המסמך).
פונקציה זאת צריכה לעבוד רק באוביקט MtmMat וצריכה לזרוק שגיאה של ChangeMatFail בכל אחד מהמחלקות בנים של המטריצה.

פונקציות לוקטור MtmVec:

מחלקה זו מתארת וקטור בסיסי.

הפונקציות הנדרשות ממחלקה זו בלבד:

i. בנאי:
mtmVec(size_t m, const T& val=T())

מייצר וקטור בגודל m. הערך Val הוא הערך איתו מאתחלים את הוקטור.הוקטור מאותחל כוקטור עמודה.

ii. בנאי העתקה, הורס והשמה.

iii. T vecFunc(Func f)
מבצע את האובייקט פונקציה f על כל אחד מהאיברים בוקטור בנפרד ומחזיר אוביקט T של התוצאה.

iv. אופרטור [] להשמת איבר והחזרת איבר בתור const ולא const.

נא שימו לב שוקטורים מאותחלים בצורה של וקטור עמודה שבו הם m שורות ועמודה אחת. ביצוע פונקציית מחלקה transpose עליהם הופך אותם לוקטור שורה.

כמו כן מטריצה בעלת שורה או עמודה אחת אינה נחשבת וקטור.

<u>פונקציות למטריצה ריבועית MtmMatSq:</u>
מחלקה זו מתארת מטריצה ריבועית-מספר העמודות שווה למספר השורות בה.

הפונקציות הנדרשות ממחלקה זו בלבד:

i. בנאי:
MtmMatSq(size_t m, const T& val=T())
מייצר מטריצה ריבועית בגודל m שורות על m עמודות. הערך Val הוא הערך איתו מאתחלים את איברי המטריצה.

ii. בנאי העתקה, הורס והשמה.

iii. ~~המרה~~ בנאי למטריצה ריבועית ממטריצה רגילה. במידה והמטריצה הרגילה בקלט איננה מטריצה ריבועית תקינה, ייזרק השגיאה illegalInitialization.

<u>פונקציות למטריצה משולשת MtmMatTriag:</u>
מחלקה זו מתארת מטריצה משולשת-מספר העמודות שווה למספר השורות בה וכמו כן היא יכולה להיות משולשת עליונה- כל האיברים מתחת לאלכסון המטריצה (הערכים שאינדקס השורות בהם שווה לאינדקס העמודות) שווים לאפס.
או משולשת תחתונה- כל האיברים מעל לאלכסון המטריצה שווים לאפס.

הפונקציות הנדרשות ממחלקה זו בלבד:

i. בנאי:
MtmMatTriag (size_t m, const T& val=T(), bool isUpper_t=true)
מייצר מטריצה אלכסונית בגודל m שורות על m עמודות. הערך Val הוא הערך

איתו מאתחלים את איברי המטריצה החוקיים. upper_t קובע האם המטריצה היא משולשת עליונה או תחתונה.

ii. בנאי העתקה, הורס והשמה.

iii. ~~המרה~~ בנאי למטריצה אלכסונית ממטריצה רגילה וריבועית. במידה והמטריצה בקלט איננה מטריצה ריבועית תקינה, ייזרק השגיאה illegalInitialization. במידה והפלט יכול להיות מטריצה משולשת עליונה ותחתונה (למשל אם כל המטריצה היא אפסים) אז ייבחר כברירת מחדל שהמטריצה היא משולשת עליונה.

<u>פונקציות המשותפות לכל המחלקות:</u>

i. void transpose()
מבצע פעולת שחלוף (transpose) על האוביקט.

ii. אופרטורים של חיבור, חיסור וכפל של מטריצות וקטורים וסקלרים **מאותו סוג משתנה T** (int, double וכו') על פי חוקיות מתמטית (חיבור וחיסור של מטריצות רק עם מטריצות אחרות מאותה צורה או סקלרים, כפל סטנדרטי בין מטריצות ומטריצות/וקטורים/סקלרים אחרים (אין כפל pointwise).
אין דרישה עבור אופרטורים אונריים ('+=','-=' מלבד האופרטור האונרי '-'). סוג אובייקט של תוצאות בין פעולות נמצא בסוף המסמך.

iii. void resize(Dimensions dim, const T& val=T())
משנה את מימדי המטריצה כך שהאינדקסים הזוגיים שלהם נשמרים (כלומר מספר השורה ומספר העמודה). שימו לב שאין כאן הגבלה של שמירת מספר האלמנטים ויכול להיות שיווצרו אלמנטים חדשים (עם ערך val) או יימחקו ערכים ישנים. נדרש שהאובייקט עדיין יהיה מאותו סוג שהיה מקודם (למשל מטריצה ריבועית עדיין מטריצה ריבועית, או וקטור עמודה עדיין וקטור עמודה).
דוגמה לפונקציה נמצאת סוף המסמך.

<u>פונקציות המשותפות רק למטריצות השונות (לא לוקטורים):</u>

i. MtmVec<T> matFunc(Func f)
מבצע את האובייקט פונקציה f על כל אחד מהאיברים בטורי המטריצה בנפרד ומחזיר אובייקט MtmVec של וקטור שורה של התוצאות על כל טור.

ii. "אופרטור" [][] להשמת איבר והחזרת איבר בתור const ולא const. הגישה לאיבר היא על פי אינדקס שורות משמאל ואז אינדקס עמודה מימין. שימו לב שמדובר כאן בשימוש באופרטור [] על האובייקט מקורי ואז שימוש באופרטור [] נוסף על התוצאה.

דרישה נוספת על כל האובייקטים- איטרטורים:

איטרטורים הם אובייקטים המוגדרים בתוך המחלקה ומהווים כלי שעוזר במעבר על האלמנטים במחלקה. דורשים ממכם שני סוגי אופרטורים שונים:

- iterator שעובר על כל איברי המחלקה על פי סדר האינדקס הלינארי.
- nonzero_iterator שעובר על כל איברי המחלקה **ששונים מאפס** על פי סדר האינדקס הלינארי.

הפונקציות הצריכות להיכלל באיטרטורים הם:

i. אופרטור '++' (מצד שמאל כמו it++) המקדם האת האיטרטור צעד אחד.

ii. אופרטור '*' אונרי המבצע גישה לאינדקס עליו מצביע האיטרטור.

iii. אופרטור '=!' ו-'==' להשוואה בין המיקום אליו איטרטורים מצביעים.

האובייקטים של מטריצות ווקטורים אותם מגדירים בתרגיל צריכים לממש את הפונקציות מחלקה הבאות:

i. iterator begin() -מחזיר איטרטור לאינדקס הלינארי הראשון.

ii. nonzero_iterator nzbegin() -מחזיר איטרטור לאינדקס הלינארי שאינו אפס הראשון.

iii. iterator end() -מחזיר איטרטור לסוף האלמנטים באובייקט (שימו לב שזה לא האיבר האחרון אלא איבר דמה אחריו).

iv. nonzero_iterator nzend() -מחזיר איטרטור לסוף האלמנטים באובייקט (שימו לב שזה לא האיבר האחרון אלא איבר דמה אחריו).

שימו לב שכל שינוי על האובייקט של האיטרטור (למשל שינוי צורה) יכול לגרום לאי נכונות של האיטרטור ואין זה באחריות המתכנת לדאוג למקרים אלה.

חריגות המוגדרות:

כל החריגות יורשות מהמחלקה MtmException המוגדרת ב-MtmException.h.
החריגות צריכות להדפיס פלט ספציפי עבור פונקציית מחלקה ()what שלהן, יש לוודא שהפלא מתיישב עם קובץ main_out.

חריגות:

IllegalInitialization-במידה וערך אתחול של אובייקט לא חוקי.

OutOfMemory-במידה ונגמר זיכרון בעת פעולה כלשהי.

DimensionMismatch-במידה ופעולה נכשלת משום שמימדי מטריצות או וקטורים לא תואמים. התיאור של החריגה צריך להכליל מה הם המימדים של שתי המשתנים (וקטור עמודה נחשב כמימד 1 ב-col וההפך עבור וקטור שורה).

ChangeMatFail-במידה ופעולת reshape אינה בעלת אותו מספר איברים כמו האוביקט המקורי, הצורה החדשה מכילה מימד בגודל 0 או resize לא עונה על ההגדרה של וקטור/מטריצה ריבועית/מטריצה אלכסונית.

AccessIllegalElement-גישה לאלמנט לא חוקי.

נדרש ממכם שהפלט של פונקציית ()what של החריגות יהיה בפורמט מסוים שמתואר בקובץ MtmException.h.

אובייקט פונקציה:

חלק מהפונקציות הנדרשות בתרגיל מחייבות כקלט אובייקט פונקציה. עבור התרגיל, אובייקט פונקציה מוגדר כמחלקה אשר מקיימת את התנאים הבאים:

1. Constructor ו-destructor.
2. אופרטור () שמקבל כקלט משתנה בודד המתאים לסוג התבנית של המחלקה עליו הוא עובד (אין צורך שתחזיר ערך).
3. אופרטור * שמחזיר תוצאה של הפעולה למחלקה.

בקובץ main.cpp המצורף לתרגיל יש דוגמה למחלקה בשם maxAbsolute אשר שומרת את האיבר עם הערך המוחלט הגבוה ביותר.

קימפול:

התרגיל צריך להתקמפל תחת הפקודה הבאה:

```
g++ -std=c++11 –Wall –Werror –pedantic-errors –DNDEBUG *.cpp –o [program name]
```

משמעות הפקודה:

- **-std=c++11** שימוש בתקן השפה C++11

- **-o [program name]** הגדרת שם הקובץ המהודר
- **-Wall** דווח על כל האזהרות.
- **-errors-pedantic** דווח על סגנון קוד שאינו עומד בתקן הנבחן כשגיאות
- **-Werror** התייחס לאזהרות כאל שגיאות – משמעות דגל זה שהקוד חייב לעבור הידור ללא אזהרות ושגיאות.
- **-DNDEBUG** מוסיף את השורה #define NDEBUG בתחילת כל יחידת קומפילציה. בפועל מתג זה יגרום לכך שהמאקרו assert לא יופעל בריצת התוכנית.

אם המהדר בשקת לא מזהה את הדגל std=c++11– הריצו את הפקודה:

/usr/local/gcc4.7/setup.sh

חלק יבש:

1. ציירו סקיצה כיצד תכננתם את המערכת מבחינת יחסי הורשה וכיצד תכננתם את הפרמטרים באזור הפרטי של המחלקה של המטריצה. הסבירו מדוע בחרתם בבחירות אלה.

2. כעת התבקשתם לייצר מחלקה חדשה של מטריצות אלכסוניות. האם תיישמו את זה כמחלקה יורשת או מחלקת אב למה שכתבתם בחלק הרטוב? ואם כן איפה? הסבירו מדוע.

3. כעת התבקשתם לייצר מחלקה חדשה של מטריצות דלילות. זוהי מחלקה של מטריצות עם סדרי גודל שיכולים להגיע למאות אלפיי שורות ועמודות, אך מעט מאוד איברים בהן ששונים מאפס. האם תיישמו את זה כמחלקה יורשת או מחלקת אב למה שכתבתם בחלק הרטוב? ואם כן איפה? הסבירו מדוע.

4. בחלק הרטוב ודאי שמתם לב שאנו ביקשנו רק אופרטורים של כפל וחיבור בין מטריצות עם אותו סוג משתנים. מה הבעיה עם בקשה של פעולה כמו כפל בין סוגי משתנים שונים?

5. בתרגיל ביקשנו שכל החלק הרטוב יהיה תחת ה-namespace של MtmMath, מדוע זהו מנהג תכנות נכון?

6. מדוע חשוב לעבוד עם רפרנסים לאוביקטים שלנו (מטריצות ווקטורים) במהלך התרגיל?

הגשה:

יש לבצע הגשה אלקטרונית בלבד דרך אתר הקורס בזוגות בלבד.

ההגשה הינה בתיקיית zip, כאשר כל הקבצים נמצאים בתיקיית השורש **(לא בתוך תיקיות פנימיות)**.

הקבצים אשר צריכים להיכלל בשביל החלק הרטוב צריכים להיות תחת השם: Auxilaries.h, Complex.h, MtmMat.h, MtmMatSq.h, MtmVec.h, MtmMatTriag, MtmExceptions.h. וקבציי cpp עם שמות מתאימים במידה והמימוש שלכם נמצא בהם.

עבור החלק היבש צריך להיות קובץ pdf בשם dry.pdf המכיל את כל הפתרונות לחלק היבש. **אין להגיש סריקות**.

על מנת לבטח את עצמכם נגד תקלות בהגשה האוטומטית:

- שימרו את קוד האישור עבור ההגשה (תמונת מסך). עדיף לשלוח גם לשותף.
- שימרו עותק של התרגיל על חשבון ה-csl2 שלכם לפני ההגשה האלקטרונית ואל תשנו אותו לאחריה (שינוי הקובץ יגרור שינוי חתימת העדכון האחרון)
- כל אמצעי אחר לא יחשב הוכחה לקיום הקוד לפני ההגשה.

ניתן להגיש את התרגיל מספר פעמים, רק ההגשה האחרונה נחשבת.

נספח:

תוצאת חיבור וחיסור מבחינת סוגי משתנים:

	וקטור	**מטריצה**
סקלר	וקטור	מטריצה
וקטור	וקטור	מטריצה
מטריצה	מטריצה	מטריצה

תוצאות כפל מבחינת סוגי משתנים:

	וקטור	**מטריצה**
סקלר	וקטור	מטריצה
וקטור	מטריצה	מטריצה
מטריצה	מטריצה	מטריצה

במקרה של פעולות בין אובייקטים בנים של מטריצה, התוצאה צריכה להיות מטריצה ולא מחלקת הבנים כמו מטריצה ריבועית. יש להתעלם ממקרים כמו כפל/חיבור של מטריצות מרובעות בהכרח יוצרות מטריצה מרובעת.

אינדקסים לינאריים

אינדקס לינארי מוגדר בתור "שורת האינדקס פלוס (עמודת האינדקס כפול מספר השורות)".

דוגמא:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

דוגמה לתהליך reshape:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

$$\xrightarrow{reshape(Dimensions(2,8))}$$

0	2	4	6	8	10	12	14
1	3	5	7	9	11	13	15

דוגמה לתהליך resize:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

$$\xrightarrow{resize(Dimensions(3,5),7)}$$

0	4	8	12	7
1	5	9	13	7
2	6	10	14	7