מבוא לתכנות מערכות – 234124

תרגיל בית 2
סמסטר אביב 2025

תאריך פרסום: 20.05.2025
זמן אחרון להגשה: 2.6.2025 עד השעה 23:55
מתרגל אחראי לתרגיל: אגבאריה מוחמד בראא

1 הערות כלליות

- תרגיל זה מהווה 6% מהציון הסופי בקורס.
- התרגיל להגשה בזוגות בלבד.
- כל ההודעות והעדכונים הנוגעים לתרגיל זה יפורסמו באתר הקורס ב-GR.
- מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל בפיאצה (קישור באתר הקורס) או בסדנאות. אין לשלוח דוא"ל לגבי התרגיל.
- שימו לב – לא תינתנה דחיות למועד הגשת התרגיל.
- על כל הקוד שאתם כותבים לעמוד בקונבנציות המפורטות בקובץ "קונבנציות לכתיבת קוד" הנמצא באתר הקורס. אי עמידה בקונבנציות עלולה לגרור הורדת ניקוד.
- קבצי התרגיל מסופקים לכם ב-GitHub.
- החומר הנדרש כדי לפתור את התרגיל הינו החומר הנלמד עד הרצאה 5 ותרגול 5 (כולל). אין צורך להשתמש בחומרים שנלמדו בתרגול ובהרצאות שאחרי.
- תיקונים למסמך התרגיל מסומנים בצהוב.

2 הקדמה

המטרות של תרגיל בית זה הן כתיבת מחלקות ראשונות בשפת C++, תרגול העמסת אופרטורים והמשך התנסות ב-Git.
מומלץ לפתור את התרגיל לפי הסדר – תחילה את החלק היבש ולאחר מכן את החלק הרטוב. לכל שאלה או בקשה לעזרה בתרגיל ניתן לכתוב לנו בפורום הקורס בפיאצה או להגיע לסדנאות לעזרה בשיעורי הבית (זמנים ומיקומים מפורסמים באתר הקורס) ונשמח לעזור.
לפני פרסום שאלה בפורום אנא בדקו אם כבר נענתה – מומלץ להיעזר בכלי החיפוש שהוצגו במצגת האדמיניסטרציה בתרגול הראשון.

3 חלק יבש

לופי וחבריו יוצאו לחפש את האוצר הגדול מכולם! בשביל להקל על מציאת האוצר, לופי החליט לכתוב קוד שיאגד את חברי צוות הפיראטים. בנוסף, מכיוון שלופי יודע דבר או שניים על עבודת צוות, החליט לנהל את פיתוח הקוד ב-Git. מטרת חלק זה היא להתנסות בעבודה עם branches, ביצוע merge ופתרון קונפליקטים הנוצרים במהלך ה-merge.

את התשובות לחלק זה יש לצרף בקבצים dry.pdf ו-log.txt (פירוט על כל אחד מהם בהמשך).

<u>שימו לב</u> – חלק זה לא יבדק בבדיקה אוטומטית, אלא רק בבדיקה ידנית.

3.1 יצירת Repository פרטי ומשיכת קבצי התרגיל

תוכלו למצוא את הקבצים מסופקים המסופקים לתרגיל זה ב-GitHub. בדומה לתרגיל בית 1, עליכם ליצור Repo פרטי בחשבון ה-GitHub שלכם, המכיל את קבצי התרגיל כפי שסופקו.

3.2 תיקון שגיאת קומפילציה

פתחו את ה-Repo הלוקאלי על המחשב שלכם. תחילה, ודאו שאתם תחת Branch הנקרא "master", זהו ה-Branch המרכזי של הפרויקט. ב-Repo תוכלו למצוא תיקיה בשם "dry" ובה מספר קבצי קוד. פתחו פרויקט חדש ב-CLion בתיקיה "dry" ונסו לקמפל את הקבצים הנמצאים בה (כולם יחד).

תוכלו לראות שיש שגיאת קומפילציה בקובץ main.cpp. המשימה הראשונה שלכם לתרגיל זה היא להבין מדוע השגיאה מתרחשת ולתקן אותה. לאחר שתיקנתם את השגיאה, ודאו שהתכנית מתקמפלת ורצה בהצלחה. הוסיפו Commit חדש המכיל את התיקון שביצעתם ל-Branch "master".

מדוע התרחשה השגיאה? כיצד תיקנתם אותה? פרטו את תשובתכם בקובץ dry.pdf.

3.3 הרחבת התכנית

כעת, נרצה להוסיף תכונה (פיצ'ר) חדש למערכת של לופי. עבור כל אחד מהפיראטים בצוות מוגדר פרס כספי (Bounty) למי שיסגיר את הפיראט ויעזור לתפוס אותו.

הוסיפו למחלקה Pirate את התכונות הבאות:

- שדה מסוג מספר שלם המייצג את ה-Bounty.
- אתחול של השדה בבנאי (באמצעות קבלה כארגומנט).
- מתודות המאפשרות גישה ושינוי של השדה.
- הדפסה של השדה באופרטור הדפסה.

בנוסף, התאימו את המימוש של פונקציית ה-main כך שה-Bounty של לופי הוא 1000000 ושל זורו 500000.

הוסיפו Commit חדש המכיל את כלל השינויים שביצעתם ל-Branch "master".

3.4 מיזוג Branches וקונפליקטים

במקביל לכך שהוספתם את הפיצ'ר החדש לתכנית, זורו הרחיב גם הוא את התכנית. זורו הוסיף למערכת enum בשם "DEVIL_FRUIT" ובו מספר סוגים של פירות שיכולים להיות בבעלותם של הפיראטים. זורו גם הוסיף למחלקה Pirate שדה מסוג ה-enum, מתודות חדשות וכו'. את השינויים זורו ביצע ב-Branch שנקרא "devil_fruit".

כעת עליכם לשלב את השינויים שביצעתם בסעיף הקודם עם השינויים שזורו ביצע. מזגו את השינויים הנמצאים ב-devil_fruit אל ה-Branch עליו עבדתם קודם (master) באמצעות פקודת merge.

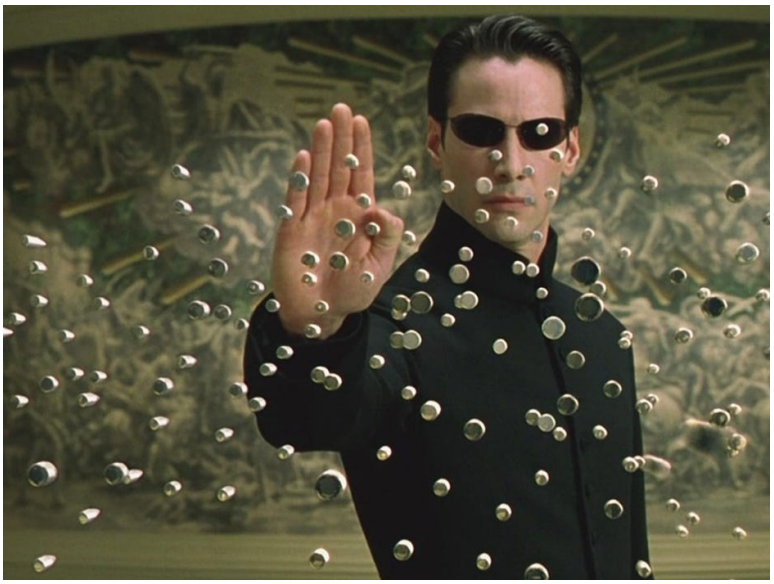
אבוי, נראה שנתקלנו בקונפליקטים! פתרו את הקונפליקטים כפי שלמדנו בתרגול 3, כך שלאחר פתרונם ישארו בתכנית גם השינויים שביצעתם וגם השינויים שזורו ביצע (כלומר המחלקה Pirate צריכה לתמוך גם ב-Bounty וגם ב-Devil Fruit). לפני המשך המיזוג, וודאו שהפתרון של הקונפליקטים עובד ושהתכנית מתקמפלת.

לאחר הפתרון של הקונפליקטים וסיום המיזוג, קמפלו והריצו את התכנית. צרפו צילום של הרצת התכנית ב-Shell עם הפלט המתקבל ל-dry.pdf.

בנוסף, עליכם להדפיס את ה-log של ה-Repo לקובץ log.txt באמצעות הפקודה הבאה:

```
git log --all -p > log.txt
```

יש לצרף להגשה את הקובץ log.txt.



4 חלק רטוב

ברכות על ההצטרפות ל- MataMvidia, כאן אנו פורצים את גבולות טכנולוגיית העיבוד הגרפי! ה-GPU שלנו ידועים בביצועים הגבוהים שלהם וביכולות החדישות שלהם, תוך שימוש בפעולות מתמטיות מורכבות המבטיחות טרנספורמציות גרפיות חלקות ויעילות.

כחברים חדשים בצוות הנדסת התוכנה שלנו, מוטלת עליכם המשימה לפתח מרכיב חיוני לטכנולוגיה שלנו: מודול מטריצות חזק וגמיש. מודול זה יהווה את עמוד השדרה של רבות מהטרנספורמציות הגרפיות שלנו, כולל תזוזות, סיבובים ופעולות מסובכות אחרות שגורמות לגרפיקה שלנו להתבלט.

4.1 הערות מימוש כלליות

- את כל הקבצים לחלק זה יש לכתוב בתיקיה wet ולצרפם ל-Repo שיצרתם בחלק היבש (אותו תגישו).
- תוכלו למצוא דוגמאות שימוש לכלל הדברים אותם אתם נדרשים לממש בסעיפים אלו בקובץ test.cpp המסופק לכם.
- על כל המחלקות שאתם מממשים בתרגיל זה יש לתמוך במופעים מסוג const.
- <u>אסור</u> לכם להוסיף לממשק המודולים דבר מעבר למינימום הנדרש.

4.2 טיפול בשגיאות

בקובץ Utilities.h תוכלו למצוא enum המייצג שני סוגים שונים של שגיאות אפשריות. בנוסף תוכלו למצוא בו פונקציה בשם "exitWithError" המקבלת ערך שגיאה, מדפיסה הודעה מתאימה ומסיימת את ריצת התכנית. בכל מקרה של שגיאה באחד המודולים שתממשו, יש לקרוא לפונקציה זו עם ערך שגיאה מתאים.

Matrix 4.3

תזכורת מאלגברה אמ' – מטריצה הינה מערך דו-מימדי של מספרים (או איברים מחוג כללי יותר). מטריצה מאופיינת באורך (מספר שורות) ורוחב (מספר עמודות).

בתחום עיבוד התמונה, מטריצות משמשות למגוון משימות כמו אחסון ומניפולציות שונות על תמונות. בסעיף זה, נממש מודול לעבודה עם מטריצות ובו מחלקה בשם Matrix שתייצג לנו מטריצה של מספרים שלמים. את הממשק והמימוש של המודול עליכם לכתוב בקבצים Matrix.h ו-Matrix.cpp בהתאמה.

4.3.1 שדות המחלקה והקצאות זכרון

עליכם להגדיר שדות למחלקה.
לצורכי יעילות, נרצה להימנע כמה שניתן מהקצאות זכרון דינמיות. לכן, עליכם לשמור את איברי המטריצה בתור מערך חד מימדי בגודל n*m, כאשר n ו-m הם אורך ורוחב המטריצה בהתאמה. על מנת לגשת לאיבר ה-i,j במטריצה (כלומר בשורה ה-i ובעמודה ה-j), נוכל לגשת לאינדקס i*width + j במערך (כאשר width הוא רוחב המטריצה).

שימו לב – לאורך כל התרגיל, עבור כל המתודות של Matrix, האינדקס הראשון מייצג את מספר **השורה** והשני את מספר **העמודה**. כאשר מתייחסים לאורך המטריצה הכוונה היא למספר השורות, כאשר מתייחסים לרוחב המטריצה הכוונה היא למספר העמודות.

4.3.2 בנאים

עליכם להגדיר למחלקה בנאי המקבל אורך n ורוחב m ומאתחל מטריצה בגודל n*m, הערך ההתחלתי של המטריצה יתקבל בתור פרמטר שלישי בבנאי, אם לא מתקבל ערך התחלתי המטריצה תאותחל לאפסים.
בנוסף, עליכם להגדיר בנאי דיפולטי המאתחל מטריצה באורך 0 וברוחב 0, כלומר ללא איברים כלל.

```
Matrix matrix(3, 6);  // Creates a matrix with 3 rows and 6 columns, containing zeros
Matrix matrix2(2, 2, 4);  // Creates a matrix of size 2x2 containing 4 in all cells

Matrix matrix3; // Creates a matrix of size 0x0
```

4.3.3 העתקות והשמות

על המחלקה שלכם לתמוך בהעתקה והשמה של אובייקטים מסוג Matrix.

4.3.4 גישה לאיבר במטריצה

גישה לאיבר (קריאה וכתיבה) תתבצע באמצעות אופרטור ( )

```
matrix2(1,0) = 2; // Placement of 2 in index (1,0)
cout << matrix2(1,0) << endl; // Prints “2”

matrix(100,0) = 3; // Error
```

4.3.5 הדפסת המטריצה

הדפסת המטריצה תתבצע באמצעות אופרטור >>, לדוגמה:

```
cout << matrix2 << endl;
```

בתחילת ובסוף כל שורה, כמו גם בין כל שני איברים בשורה, יודפס קו אנכי "|". לדוגמה, עבור המטריצה matrix2 שהוגדרה בסעיפים הקודמים יודפס:

|0|0|
|2|0|

ניתן לראות דוגמאות להדפסות בטסטים המסופקים.

4.3.6 אופרטורים אריתמטיים

על המחלקה שלכם לתמוך באופרטורים האריתמטיים הבאים:

- +, -, * בין שתי מטריצות, לפי כללי חיבור, חיסור וכפל מטריצות בהתאמה.
- +=, -=, =* בין שתי מטריצות, בהתאם לאופרטורים בסעיף הקודם.
- מינוס אונארי (-) – מחזיר את המטריצה הנגדית.
- * בין מטריצה למספר שלם (משני הצדדים), בהתאם לכללי כפל בסקלר.
- =* בין מטריצה (משמאל) למספר שלם (מימין) בהתאם לאופרטור בסעיף הקודם.

4.3.7 אופרטורי השוואה

על המחלקה שלכם לתמוך באופרטורי ההשוואה == ו-=! בין שתי מטריצות. שתי מטריצות שוות זו לזו אם ורק אם הן באותו גודל ולכל i,j האיבר בשורה ה-i ובעמודה ה-j זהה בשתי המטריצות.

4.3.8 סיבוב מטריצה ב-90 מעלות

עליכם לממש מתודות בשם rotateClockwise ו-rotateCounterClockwise המחזירות מטריצה מסובבת ב-90 מעלות עם כיוון השעון או נגד כיוון השעון בהתאמה.
לדוגמה, עבור המטריצה הבאה:

|1|2|3|
|4|5|6|

סיבוב עם כיוון השעון יחזיר את המטריצה הבאה:

|4|1|
|5|2|
|6|3|

4.3.9 Transpose

עליכם לממש מתודה בשם transpose המחזירה עותק של המטריצה משוחלפת (לפי הגדרת ה-Transpose שאתם מכירים מאלגברה אמ').

4.3.10 נורמת פרובניוס של מטריצה

נורמת פרובניוס – Frobenius – על מטריצה A באורך n ורוחב m מוגדרת באופן הבא:

$$\|A\| = \sqrt{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} |A_{i,j}|^2}$$

כתבו פונקציה סטטית של המחלקה, בשם CalcFrobeniusNorm אשר מקבלת מטריצה ומחשבת את נורמת פרובניוס עבורה.

4.3.11 (בונוס, 5 נקודות) חישוב דטרמיננטה של מטריצה

כתבו פונקציה סטטית בשם CalcDeterminant אשר מקבלת מטריצה ומחזירה את ערך הדטרמיננטה שלה.
רמז: רקורסיה
הערה: העתקה של החלק הזה תגרור פסילה של כל תרגיל הבית וגם היא תטופל בחומרה.

4.4 MataMvidia

במודול זה עליכם להגדיר מחלקה בשם MataMvidia המייצגת סרט. סרט הוא רצף סדור של פריימים (Frames), כל פריים בסרט מיוצג על ידי מטריצה של מספרים שלמים. את הממשק והמימוש של המודול עליכם לכתוב בקבצים MataMvidia.h ו-MataMvidia.cpp בהתאמה.

סרט מאופיין על ידי:

- אורך הסרט (כמות פריימים).
- אוסף הפריימים בסרט.
- שם הסרט.
- שם היוצר.

שימו לב – הפריימים בסרט לא נדרשים להיות אחידים בגודל.

4.4.1 בנאי

עליכם להגדיר בנאי המקבל את שם הסרט, שם היוצר, מערך של פריימים ואת כמות הפריימים במערך. על האובייקט לשמור עותקים של הפריימים אותם קיבל בבנאי.

```
Matrix matrix(2, 2);
matrix(0,1) = 100;
Matrix matrix2(2, 2);
Matrix array[] = {matrix, matrix2} // array[0]=matrix, array[1]=matrix2
MataMvidia movie("MataMatrix", "Baraa Egbaria", array, 2);
```

4.4.2 העתקות והשמות

על המחלקה שלכם לתמוך בהעתקה והשמה של אובייקטים מסוג MataMvidia.

4.4.3 אופרטורים

על המחלקה לתמוך באופרטורים הבאים:

- אופרטור [ ] – מאפשר גישה לפריים בודד בסרט (קריאה וכתיבה).
- אופרטור =+ בין שני סרטים – משרשר את הפריימים של הסרט המתקבל בצד ימין לסוף הסרט בצד שמאל.
- אופרטור =+ בין סרט (שמאל) ומטריצה (ימין) – משרשר את המטריצה בסוף רשימה הפריימים של הסרט.
- אופרטור + בין שני סרטים – מחזיר סרט חדש המהווה שרשור של שני הסרטים. השם והיוצר של הסטר המתקבל הם שם הסרט ושם היוצר של הסרט שהתקבל מצד שמאל של האופרטור.

4.4.4 הדפסה

עליכם לממש אופרטור >> המדפיס סרט בפורמט הבא:

Movie Name: *movie_name*
Author: *author_name*

Frame 0:
|…|…|…
|…|…|…
…

Frame 1:
…

Frame *n*:
|…|…|…
|…|…|…
…

-----End of Movie-----

למשל, עבור הסרט שהוגדר קודם יודפס:

Movie Name: MataMatrix
Author: Baraa Egbaria

Frame 0:
|0|100|
|0|0|

Frame 1:
|0|0|
|0|0|

-----End of Movie-----

4.5 בדיקות מסופקות

בקובץ tests.cpp תוכלו למצוא תכנית פשוטה הבודקת את המודולים Matrix ו-MataMvidia. יש לקמפל את הבדיקות עם הקבצים שכתבתם באמצעות הפקודה (בתוך התיקיה wet):

```
g++ -DNDEBUG -std=c++17 -Wall -pedantic-errors -Werror -o Mvidia *.cpp
```

ניתן להריץ את כל הבדיקות על ידי הפקודה:

```
./Mvidia > test.out
```

יש להשוות את פלט התכנית עם קובץ הפלט המסופק באמצעות הפקודה:

```
diff --strip-trailing-cr -B -Z test.out test.expected
```

5 הגשה ובדיקה

5.1 הגשה

עליכם להגיש את הקבצים הבאים בתיקיה dry:

- dry.pdf
- log.txt
- Pirate.h
- Pirate.cpp
- main.cpp

עליכם להגיש את הקבצים הבאים בתיקיה wet:

- Matrix.h
- Matrix.cpp
- MataMvidia.h
- MataMvidia.cpp

את ההגשה יש לבצע במערכת ה-Gradescope.

שימו לב – על אחד השותפים (בלבד) להגיש את פתרון התרגיל במערכת. לאחר ההגשה ניתן להוסיף את השותף השני להגשה באמצעות לחיצה על כפתור "Group Members" ב-Gradescope.

הערות:

- ניתן להגיש מספר פעמים את התרגיל, ההגשה האחרונה (בלבד) היא זו שתיבדק ותקבל ציון.
- וודאו שהקבצים אותם אתם מגישים (בפרט קבצי PDF) ניתנים לפתיחה ולצפיה.

5.1.1 הגשה באמצעות GitHub

בתרגיל זה, יש להגיש את הפתרון שלכם במערכת ה-Gradescope באמצעות GitHub. עליכם להגיש את ה-Repo שיצרתם בחלק היבש ואליו הוספתם את הפתרון לחלק הרטוב. לשם כך בדף הגשת התרגיל בחרו באפשרות ההגשה באמצעות GitHub, תנו ל-Gradescope הרשאות גישה לחשבון ה-GitHub שלכם ובחרו את ה-Repo אותו תרצו להגיש.

5.2 בדיקה אוטומטית

מיד עם ההגשה שלכם במערכת ה-Gradescope, יתבצעו מספר בדיקות אוטומטיות על קובץ ההגשה:

1. בדיקות שפיות – בודקות שכל הקבצים הדרושים נמצאים ב-zip המוגש ושהקוד שלכם מתקמפל כראוי. בדיקות אלו נועדו לוודא שפורמט ההגשה שלכם תקין.
2. טסטים אוטומטיים – הבדיקה האוטומטית מריצה את התכניות שהגשתם עם הקלטים שסופקו לכם בקבצי התרגיל. הטסטים האוטומטיים בודקים שריצת התכנית הסתיימה בהצלחה ללא שגיאות זכרון ונתנה פלט נכון.

בדיקות אלו נועדו עבורכם לוודא שההגשה שלכם תקינה והקוד עובד כראוי, קראו היטב את הפלט והמשוב מהבדיקות!

על מנת להשוות תכנים של קבצים בבדיקה האוטומטית, נשתמש בפקודת ה-diff הבאה (כאשר file1 ו-file2 הם הקבצים אותם נרצה להשוות:

```
diff --strip-trailing-cr -B -Z file1 file2
```

<u>שימו לב</u> – בנוסף לבדיקות שסופקו לכם, ההגשה הולכת לעבור בדיקות אוטומטיות נוספות לפיהן יקבע הציון על התרגיל. אנו מצפים שתבדקו בעצמכם את התרגיל שלכם מעבר לבדיקות המסופקות, האחריות על נכונות ההגשה היא שלכם בלבד.

5.3 תחרות מימז

בתרגיל זה מתקיימת תחרות מימז (Memes).
כדי להשתתף בתחרות, עליכם להוסיף מימ לתחילת הקובץ dry.pdf. הזוג שיזכה בתחרות יקבל בונוס של 5 נקודות לציון התרגיל (לכל אחד מהשותפים).

חוקי התחרות:

1. מותר לכם להגיש לתחרות מימ אחד <u>בלבד</u>.
2. נושא המימ חייב להיות קשור לחומר הקורס.
3. המימ הזוכה יפורסם בתרגול לאחר לפרסום הציונים.

בהצלחה!

بالنجاح!

GOOD LUCK!