סדנת תכנות בשפות C ו-C++ (67312)

תאריך הגשה: יום חמישי, 18/11/2021 בשעה 23:55

תאריך הגשה באיחור בקנס של 10 נקודות: יום שישי, 12/18/2021 בשעה 23:55

נושאי התרגיל: מצביעים למצביעים, מערכים דו-מימדיים, ו-Makefile

1 הקדמה

בתרגיל זה נממש ספרייה התומכת בפונקציונליות מתקדמת של מערכים דו-מימדיים (מטריצות) של מספרים (מטיפוס double) תוך שמירה על יעילות. הספרייה תאפשר יצירת מטריצות, שרשור של שתי מטריצות לפי שורות/עמודות, גישה לפי אינדקס שורה ועמודה וגישה לתתי-מטריצות באינדקסים נבחרים (slicing). הספרייה מורכבת ממבנה נתונים matrix ומאוסף פונקציות שעליכם לממש. **קראו את ההוראות במלואן לפני תחילת המימוש.**

2 הספרייה

מבנה הנתונים, החתימות של פונקציות הספרייה ותיעוד על תפקידה של כל אחת מהן נמצאים בקובץ ex3.h שסופק לכם עם קבצי העזר, את הפונקציות תממשו בקובץ ex3.c.

2.1 מבני נתונים

המטריצה מורכבת מ-struct המכיל את השדות הבאים:

- שדה מטיפוס **double שמצביע למערך דו-מימדי דינמי.
- n_rows - מספר השורות במטריצה.
- n_columns - מספר העמודות במטריצה.

2.2 פונקציות הספרייה

```
matrix *create_matrix (size_t n_rows, size_t n_columns);
```

הפונקציה מקבלת את מספר השורות הרצוי ומספר העמודות ומחזירה מצביע למטריצה חדשה (הנמצאת על ה-heap). שדה ה-data של המטריצה החדשה יצביע למערך דו-מימדי דינמי עם n_rows שורות ו-n_colums עמודות המוקצה על ה-heap ומאותחל כולו ל-0.0, והשדות n_rows, n_cols יכילו את מספרי השורות והעמודות בהתאמה. אם הקצאת זיכרון נכשלה במהלך יצירת המטריצה, הפונקציה תדפיס הודעת שגיאה אינפורמטיבית ל-stderr, תשחרר את הזיכרון שהוקצה עד כה ותחזיר NULL. אם מספר השורות או מספר העמודות הוא 0, המטריצה ריקה והשדה data יכיל NULL, השדות n_rows ו-n_columns יהיו **שניהם** 0.

```
double *get_by_index (matrix *mat, size_t row_index, size_t col_index);
```

הפונקציה מקבלת מטריצה, אינדקס לשורה ואינדקס לעמודה ומחזירה מצביע לערך השמור במטריצה בשורה ובעמודה המבוקשים. אם התקבל אינדקס לא חוקי (אינדקס גדול ממספר השורות/העמודות), הפונקציה תדפיס הודעת שגיאה אינפורמטיבית ל-stderr ותחזיר NULL.

```
matrix *slice (matrix *mat, size_t row_start,
               size_t row_end, size_t col_start, size_t col_end);
```

הפונקציה מקבלת מטריצה, אינדקס התחלה וסיום עבור השורות ואינדקס התחלה וסיום עבור העמודות ותחזיר מטריצה **חדשה** המכילה את השורות במטריצה שהאינדקס שלהן גדול/שווה ל-row_start וקטן (ממש) מ-row_end כשכל שורה כזו מכילה את העמודות שהאינדקס שלהן גדול/שווה ל-col_start וקטן (ממש) מ-col_end (במילים אחרות, הפונקציה מחזירה את ערכי המטריצה במיקום (i, j) כך ש-$i \in [\text{row\_start}, \text{row\_end})$ ו-$j \in [\text{col\_start}, \text{col\_end})$).
לדוגמה אם המטריצה היא

$$\begin{bmatrix} 1 & 2 & 4 & 6 \\ 16 & 6 & 24 & 9 \\ 8 & 3 & 5 & 31 \end{bmatrix}$$

$\text{row\_start} = 0$, $\text{row\_end} = 2$, $\text{col\_start} = 1$, $\text{col\_end} = 4$ אז המטריצה שתוחזר תהיה

$$\begin{bmatrix} 2 & 4 & 6 \\ 6 & 24 & 9 \end{bmatrix}$$

שימו ♡: אין דרישה כלשהי על row_start, row_end, col_start, col_end וכל רביעיית מספרים היא קלט חוקי, אך יתכן שהמטריצה המתקבלת תהיה ריקה (השדה data יהיה NULL, ומספר השורות והעמודות הוא 0).

```
matrix *concatenate_vertically (matrix *top, matrix *bottom);
```

הפונקציה תקבל זוג מטריצות top, bottom כאשר top היא מטריצה $m \times k$ ו-bottom היא מטריצה $n \times k$ ותחזיר מטריצה **חדשה** $(m+n) \times k$ כש-m השורות הראשונות הן השורות של top ו-n השורות האחרונות הן השורות של bottom. לדוגמה אם

$$\text{top} = \begin{bmatrix} 1 & 7.6 & 4 \\ 0.3 & 0 & -3 \\ 5 & 11 & 21 \end{bmatrix}$$

ו-

$$\text{bottom} = \begin{bmatrix} 6 & 135 & 1 \\ 21.8 & 15 & 9 \end{bmatrix}$$

אז המטריצה החדשה תהיה

$$\begin{bmatrix} 1 & 7.6 & 4 \\ 0.3 & 0 & -3 \\ 5 & 11 & 21 \\ 6 & 135 & 1 \\ 21.8 & 15 & 9 \end{bmatrix}$$

אם מספר העמודות של top שונה ממספר העמודות של botoom או שהקצאת זיכרון כלשהי נכשלת, הפונקציה תדפיס הודעת שגיאה אינפורמטיבית ותחזיר NULL.

```
matrix *concatenate_horizontally (matrix *left, matrix *right);
```

הפונקציה מקבלת זוג מטריצות left, right כאשר left היא מטריצה $k \times m$ ו-right מטריצה $k \times n$ ותחזיר מטריצה **חדשה** $k \times (m+n)$ כש-m העמודות הראשונות הן העמודות של left ו-n העמודות האחרונות הן העמודות של right. לדוגמה אם

$$\text{left} = \begin{bmatrix} 0 & 1 & 4 \\ 1 & 0 & 9.5 \end{bmatrix}$$

$$\text{right} = \begin{bmatrix} 1 & 4 & 9 & 0 \\ 3 & 0 & 3 & 1 \end{bmatrix}$$

אז המטריצה החדשה תהיה

$$\begin{bmatrix} 0 & 1 & 4 & 1 & 4 & 9 & 0 \\ 1 & 0 & 9.5 & 3 & 0 & 3 & 1 \end{bmatrix}$$

אם מספר השורות של left שונה ממספר השורות של right, או שהקצאת זיכרון כלשהי נכשלת, הפונקציה תדפיס הודעת שגיאה אינפורמטיבית ותחזיר NULL.

```
matrix *transpose (matrix *mat);
```

הפונקציה מקבלת מטריצה $m \times k$ ומחזירה את השחלוף (transpose) שלה (כמטריצה **חדשה** $k \times m$).
תזכורת: שחלוף מטריצה הוא פעולת ההחלפה בין השורות והעמודות, לדוגמה השחלוף של המטריצה

$$\begin{bmatrix} 1 & 3 & 4 \\ 2 & 7 & 9 \end{bmatrix}$$

הוא

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \\ 4 & 9 \end{bmatrix}$$

```
double *ravel (matrix *mat);
```

הפונקציה מקבלת מטריצה $m \times k$ ומחזירה מצביע למערך חד-מימדי (על ה-heap) בגודל $m \cdot k$ המכיל את כל איברי המטריצה המקורית. הסדר במערך החדש הוא לפי השורות, כלומר k הערכים הראשונים הם ערכי השורה הראשונה, k הערכים הבאים הם ערכי השורה השניה וכן הלאה. לדוגמה אם נקבל את המטריצה

$$\begin{bmatrix} 1 & 3 & 4 \\ 2 & 7 & 9 \end{bmatrix}$$

המערך יהיה

$$\{1, 3, 4, 2, 7, 9\}$$

שימו לב: הפונקציה לא מחזירה מטריצה אלא מערך חד-מימדי דינמי.

```
void free_matrix (matrix *mat);
```

הפונקציה משחררת את הזיכרון של המטריצה (המערך הדו-מימדי וה-struct). שימו לב לסדר השחרור כדי שלא תגיעו למצב שזיכרון עוד לא שוחרר אך המצביע אליו שוחרר.

2.3 Make וכתיבת Makefile

בחלק זה נתרגל שימוש בסיסי ב-Make. Make היא תוכנה לניהול אוטומטי של קומפילציית קוד, והיא חלק מפרוייקט התוכנה החופשית GNU. כדי להשתמש ב-Make ניצור קובץ טקסט בשם Makefile בו

יכתבו ההוראות לקומפילציה, כאשר הפורמט הבסיסי הוא

```
target_name: dependencies
        commands
```

כאשר target_name הוא שם כלשהו (לבחירתכם), במקום dependencies נשים את קובץ הקוד שנרצה לקמפל, או שם של target נוסף שעבורו גם מוגדרות הוראות קומפילציה, ואת commands נחליף בפקודת הקומפילציה (אותה פקודה שהיינו כותבים בטרמינל). בשבוע הקרוב תלמדו בהרצאה יותר לעומק ועל תכונות יותר מתקדמות של Make. בינתיים אתם מוזמנים לקרוא על השימוש הבסיסי במדריך https://makefiletutorial.com/. לאחר שהגדרנו את הוראות הקומפילציה עבור target_name נוכל להריץ בטרמינל את הפקודה make target_name ו-make יריץ את commands והקומפילציה תתבצע. בתרגיל הנוכחי עליכם להגיש Makefile כך שאם נריץ בטרמינל בתיקיה עם קבצי הקוד את הפקודה make matrix_lib, הקובץ ex3.c יקומפל לקובץ matrix_lib.o (שלב הקומפילציה בלבד, ללא Linkage, ב-gcc קומפילציה בלבד מתבצעת על ידי הוספת c- לפקודת הקומפילציה). על הפרטים של שלבי הקומפילציה השונים תלמדו בהרצאה בשבוע לאחר פרסום התרגיל.

2.4 הערות

- **שימו ♡: בכל הפונקציות המחזירות מטריצה חדשה, אין זיכרון משותף בין המטריצות שהתקבלו כקלט, ובפרט ניתן להשתמש במטריצה החדשה שהוחזרה גם אחרי ששאר המטריצות שוחררו על ידי free_mat**. לדוגמה, ניתן לשרשר זוג מטריצות top ו-bottom ולקבל מטריצה חדשה mat, לאחר מכן לשחרר את הזיכרון של top ו-bottom ולהשתמש ב-mat.
- אין לשנות את ה-struct matrix, ולא ניתן להניח שהמטריצה שהועברה אליכם בקריאה לפונקציה אכן נוצרה על ידכם. המטרה של דרישה זו היא שנוכל לבדוק את את מימוש הפונקציות שלכם בלי תלות בין הפונקציות, כך שגם אם הפונקציה create_matrix שלכם לא עובדת כנדרש, עדיין תוכלו לעבור את הטסטים של שאר הפונקציות בהצלחה.
- שימו לב שאין לכם דליפות זיכרון בתרגיל. בדקו זאת באמצעות valgrind.
- בכל הפונקציות שמקבלות *matrix ניתן להניח שהמצביע למטריצה איננו NULL.
- הימנעו משימוש במשתנים גלובליים.
- אין להשתמש בליטרלים מספריים או מחרוזות בגוף הקוד ("magic numbers"), השתמשו ב-#define.

2.5 הגשת התרגיל

- **בתרגיל אתם לא מגישים פונקציית main**. לצורך בדיקת התרגיל שלכם, כדאי שתייצרו לעצמכם קובץ עם main שמשתמש בפונקציות הספרייה, תעשו include ל-ex3.h ותקמפלו את הקובץ ביחד עם ex3.c. אם תרצו לעשות זאת ב-CLion, תוסיפו ל-CMakeLists.txt את השורה add_executable(<target name> ex3.c <file with main>) כאשר במקום <target name> תכתבו שם כלשהו לבחירתכם, ולאחר מכן תראו ב-dropdown-list שליד אייקון ההרצה אופציה תחת השם <target name>, ואותה תריצו.

- פתרון בית הספר: סיפקנו לכם מימוש קבצים מקומפלים לפתרון בית הספר (קבצי o.), תוכלו לקמפל קובץ עם main שמשתמש בפונקציות הספרייה יחד עם קבצי פתרון בית הספר ולהריץ את המימוש שלנו לפונקציות כדי לבחון את ההתנהגות המצופה. הקבצים זמינים בנתיבים ~labcc/www/ex3/school_solution_1.o, ~labcc/www/ex3/school_solution_2.o במחשבי האקווריום. כדי לקמפל קובץ עם main שלכם, עליכם לבצע cd לתיקיה ~labcc/www/ex3 (בטרמינל במחשבי האקווריום). לאחר מכן לכתוב את הפקודה הבאה:

make school_sol_tests MAIN=<path to file with main that used the libaray functions>

פקודה זו תייצר קובץ executable בשם school_sol_tests בתיקייה בה נמצא הקובץ שמכיל את ה-main, תוכלו להריץ קובץ זה בטרמינל על ידי הרצת הפקודה ./school_sol_tests בתיקיה זו.

- הגשת התרגיל מתבצעת באמצעות git, עליכם להגיש את הקבצים ex3.h, ex3.c, Makefile ואותם בלבד.

- כדי לקמפל את התרגיל עם פונקציית main (לצורך בדיקה שלכם) הנמצאת בקובץ main.c ולייצר קובץ executable בשם ex3, יש להשתמש בפקודה הבאה (בתיקיה עם הקבצים):

gcc -Wextra -Werror -Wall -Wvla -std=c99 ex3.c main.c -o ex3

אנא ודאו שהקוד מתקמפל ללא שגיאות או אזהרות.

- מומלץ להגדיר ל-CLion לקמפל עם הגדרות זהות לקומפילציה הנעשית על ידינו (CLion יוכל לתת לכם אזהרות/שגיאות אינפורמטיביות בזמן הכתיבה). ניתן לעשות זאת על ידי הוספת השורה הבאה ל-CMakeLists.txt:

set(CMAKE_C_FLAGS "-Wextra -Werror -Wall -Wvla -std=c99")

- **וודאו כי התרגיל שלכם עובר את ה- Pre-submission ללא שגיאות או אזהרות.** כדי להריץ את הסקריפט על הקוד שלכם במחשבי בית הספר, עליכם לייצר קובץ tar המכיל את הקבצים שעליכם להגיש, בשם ex3.tar והריצו את הפקודה:

~labcc/presubmit/ex3/run <path to ex3.tar>

- וודאו שהרצת הפקודה make matrix_lib בתיקייה שמכילה את ex3.h,ex3.c,Makefile (על מחשבי בית הספר) אכן מקמפלת ומייצרת בתיקיה את הקובץ matrix_lib.o.

- **אין לבצע שינויים בקובץ ex3.h, הגישו את הקובץ שקיבלתם.**

בהצלחה! ☻