האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנאות תכנות בשפת C ו־++C – קיץ (קורס 67320)
C – תרגיל 1

תאריך ההגשה של התרגיל והבוחן התיאורטי: יום שלישי, ה־6 באוגוסט, 2019 – עד השעה 23:55;

הגשה מאוחרת (בהפחתת 10 נקודות): יום רביעי, ה־7 באוגוסט, 2019 – עד השעה 23:55.

<u>נושאי התרגיל</u>: היכרות עם השפה, קומפילציה, משפטי pre-processor, משתנים, אריתמטיקה פשוטה, קלט & פלט, תנאים, לולאות, פונקציות ומערכים סטטים (<u>ללא</u> הקצאה דינמית).

1 רקע

בתרגיל זה נלמד על פעולת Convolution[1] (קונבולוציה, קיפול) ונראה שימוש אחד שלה. Convolution היא פעולה מתמטית בינארית המוחלת על שתי פונקציות או סדרות של ערכים, ומסמלת את סכום השטח הכלוא מתחת למכפלת שתי הפונקציות או סדרות הערכים, אך כשאחת מהפונקציות משוקפת סביב הציר האנכי ומוזזת בערך כלשהו, n.

בקונבולוציה נעשה שימוש נרחב בתחומים רבים במדעי הטבע, כגון פיזיקה, סטטיסטיקה, עיבוד אותות, עיבוד תמונה ועוד.
בתרגיל זה, אנו נבחן את השימוש בקונבולוציה בזמן דיסקרטי בתחום האקוסטיקה.

1.1 הניסוי

נניח שאנו נמצאים בחדר אשר מחזיר הד עבור קולות הנשמעים בו. אם נשמיע קול בחדר זה, ההד יתנהג בצורה הבאה: לאחר שנייה אחת, עוצמת הקול שנשמעת תהא חצי מהקול המקורי; לאחר שתי שניות, היא תהא רבע ממנו, וכן הלאה. באופן כללי נקבל: $h[t] = \frac{1}{2^t}$ – כאשר $h[t]$ היא סדרה המסמנת את ההגבר.
עתה, נניח כי מוצב תוף בחדר ואדם מכה בו כל שניה בעוצמה אחרת. $g[t]$ תהא הסדרה המייצגת את עוצמת ההכאה בכל שנייה. אם אנחנו מעוניינים לדעת לאחר זמן מסוים מה תהיה עוצמת הקול בחדר, עלינו לסכום את התרומות של כל הקולות שנעשו מאז תחילת

[1] http://mathworld.wolfram.com/Convolution.html (מומלץ להביט באנימציות המופיעות בקישור כדי לקבל אינטואיציה).

התיפוף ועד לזמן שבו אנחנו מעוניינים. אלא מהי? שעבור כל הכאה בתוף עבר זמן שונה, וזהו פרמטר שנצטרך להתחשב בו. הדרך לחשב זאת היא לחבר את העוצמה שנשמעה ברגע מסויים עם חצי העוצמה שנשמעה בשנייה הקודמת, יחד עם רבע מהעוצמה שנשמעה בשנייה שלפני כן – וכן הלאה עד לתחילת התיפוף. כלומר נקבל:

$$(g * h)[\tau] = g[\tau]h[0] + g[\tau-1]h[1] + g[\tau-2]h[2] + \cdots = \sum_{n=0}^{\tau} h[n] \cdot g[\tau-n]$$

כאשר האופרטור $*$ מסמן את **הקונבולוציה** של $h[n]$ עם $g[n]$ (קונבולוציה היא פעולה קומוטטיבית, לכן אפשר לקרוא זאת גם הפוך).

<u>הערה</u>: ביטוי זה שקיבלנו הוא הקונבולוציה בזמן בדיד, אך אפשר בקלות לקבל ביטוי דומה עבור זמן רציף.

בדיוק כך, היינו יכולים להמשיך למדל את עוצמת הקול, אך הפעם מחוץ לחדר בו מכים בתופים. נניח שמחוץ לחדר עוצמת הקול (של כל הקולות שנשמעים בחדר, ללא רלבנטיות למקורם או טיבם) נחלשת פי שתים **בכל שניה**. אם כך, כבר בשניה השניה נשמע קול החלש ב־$\frac{1}{2}$ מעוצמת הקול שבחדר, בשניה השלישית ב־$\frac{1}{4}$ וכך הלאה. באופן כללי, נקבל שבזמן t עוצמת השמע מחוץ לחדר תהיה $h[t] = \frac{1}{2^t}$ – ותוצאה זו זהה לזו שקיבלנו מקודם! על כן, כדי לחשב את עוצמת הקול מחוץ לחדר במקרה זה, נדרש לחשב את:

$$((g * h) * h)(t)$$

עתה, נניח שהניסוי שתיארנו מתרחש בתוך חדר שנמצא בתוך חדר גדול יותר וכך הלאה והלאה... עד שנקבל מדובר ב־k חדרים מקוננים. על כן עלינו לחשב את

$$\underbrace{((((g * h) * h) * \cdots) * h)}_{k\ times}(t)$$

בתרגיל נעסוק בכתיבת מודל שכזה, ובייצוג גרפי שלו.

2 התוכנית DrumExperiment

בתרגיל זה תכתבו את התוכנית DrumExperiment, שמקבלת שני רשימות – האחת מייצגת הדגימות שביצענו לעוצמת ההכאה בתוף והשניה את הדגימות של ההגבר (אלו רשימות התואמות ל־$g[t]$, $h[t]$ שלעיל, בהתאמה). התוכנית תבצע את ניסוי ההכאה על התוף שתואר לעיל ותדפיס את עוצמות הרעש המנורמלת בזמן t, ביחס לעוצמה. התוכנית תתנהל כך:

1. התוכנית תקלוט מהמשתמש שתי רשימות של מספרים רציונלים חיוביים, כל רשימה בתורה. אלו יהיו הרשימות שייצגו את $g[t]$, $h[t]$. הבחירה לקבלת מספרים חיוביים נובעת מהסיבה שלא ניתן להגדיר עוצמת תיפוף שלילית או הד שלילי.

2. נוסף לכך, התוכנית תקלוט מספר שלם, שנסמן $n \in \mathbb{N} \cup \{0\}$, והוא ייצג את כמות החדרים המקוננים שבניסוי.

3. התוכנית תחשב את ערך הקונבולוציה של $g[t]$ עם $h[t]$, לאחר נירמול, n פעמים.

4. התוכנית תדפיס את ההיסטוגרמה המנורמלת של תוצאת הקונבולוציה.

להלן נציג הוראות מדויקות למימוש כל חלק ולק.

2.1 קלט

2.1.1 קליטת ערכי $g[t]$ ו־$h[t]$:

תחילה, נרצה לקלוט מהמשתמש (דרך ה־stdin) שתי רשימות של ערכים שיכילו את הדגימות של $g[t]$, $h[t]$. **מצופה שלא יהיו יותר מ־100 דגימות ב־$g[t]$ ולא יותר מ־100 דגימות ב־$h[t]$.**
כל רשימה תיקלט בשורה נפרדת כאשר בין כל ערך וערך ברשימה יהיה רווח.
כל מספר בכל אחת מהרשימות יהיה **רציונלי וחיובי** – ולכן התווים היחידים שניתן להרכיב מהם מספר הם הספרות 0 – 9 (כשהספרה 0 אינה יכולה להופיע בתחילת המספר) והנקודה העשרונית (שתופיע פעם אחת בלבד ולא בסוף המחרוזת, כלומר לא כשאינה מלווה בערכים עשרונים לאחריה).
למשל הקלט:

```
1 0.3 2 4
1 3.14 3 23.4
```

ייצג את רשימות הערכים:

$$g[t] = \{1,\ 0.3,\ 2,\ 4\},\ \ h[t] = \{1,\ 3.14,\ 3,\ 23.4\}$$

לשימושכם, באתר הקורס תמצאו קוד מקור לתוכנית לדוגמה הקוראת שורה של מספרים, המפורדים ברווח, מה־stdin ומדפיסה אותם בלולאה, מספר אחר מספר, **כמחרוזות**, ל־stdout. זו יכולה להיות נקודת התחלה למימוש חלק זה (עם זאת, מותר לממש פתרון בדרכים שונות אחרות, כל עוד הפתרון עומד בדרישות התרגיל).
עתה, לאחר שבידכם מחרוזת שאמורה לייצג מספר – עליכם להמירה למספר תקין.

<u>הערה מאוד חשובה:</u> ב־C קיימות מספר פונקציות מובנות שאפשר להיעזר בהן כדי להמיר מחרוזת למספר רציונלי, למשל `strtod`. **עם זאת, מימוש חלק זה מהווה אחת ממשימות התרגיל – לכן <u>אין</u> לעשות שימוש בפונקציות מובנות כדי לבצע את ההמרה של המחרוזת למספר רציונלי.** תוכלו להמיר char כלשהו ל־int על ידי חיסור התו '0' מה־char שבידיכם (למה?). כלומר, בהנחה ש־"c" הוא משתנה מסוג char שערכו אחד התווים מ־'0' עד '9':

```
int value = c - '0';
```

ממילא, שימוש לא זהיר בפונקציות אלו יאפשר קבלת קלט לא תקין – ולכן גם לא יעמוד בחלק מהבדיקות האוטומטיות.

דרישות והנחות באשר לקלט:

- **לא** ניתן להניח כי התוכנית תקבל קלט כלשהו לאף אחת מהרשימות.

- **לא** ניתן להניח כי הקלט תקין או בטווח הנדרש. קלט שאינו תקין הוא קלט שאינו מייצג מספר **רציונלי וחיובי** (שימו לב כיצד הגדרנו לעיל מחרוזת של מספר רציונלי).
- על הרשימות $g[t]$ ו־$h[t]$ להכיל **לכל היותר** 100 איברים. עם זאת, **לא ניתן להניח** כי תקבלו כקלט רשימה הקטנה מ־100 איברים.
- על כל מספר להיות לכל היותר באורך 9 תווים, **לרבות** הנקודה העשרונית, אם זו קיימת. עם זאת, **לא ניתן להניח** כי המספרים שתקבלו יהיו תקינים במובן זה.
- עליכם לוודא שהאורך של $h[t]$ קטן (במידה חלשה) מהאורך של $g[t]$ (כלומר $|h| \leq |g|$).
- **ניתן** להניח שאורך השורה המקסימלי יהא 1024 תווים.

2.1.2 קליטת הערך n (כמות החדרים)

לאחר קליטת שתי שורות המספרים המייצגות את $g[t]$ ו־$h[t]$, נרצה לקלוט **בשורה נפרדת** את כמות החדרים המקוננים – כלומר את ערכו של n.
לא ניתן להניח שיקלט ערך כלשהו או שהוא יהיה מספר טבעי תקין. קלט תקין במקרה זה יכיל **ספרות בלבד.** עליכם לוודא זאת.
<u>רמז:</u> כדאי להשתמש בפונקציות עזר במימוש הפונקציה שקוראת מספרים רציונלים, אותה נדרשתם לכתוב לעיל (בחלק 2.1.1). חלוקה נכונה לפונקציות עזר תאפשר לכם לוודא את תקינות ערכו של n בקלות. גם כאן **ניתן** להניח שאורך השורה המקסימלי יהא 1024 תווים.

2.1.3 טיפול בשגיאות בקלט

אם נתקלתם בקלט שאינו עומד באחת ההגדרות שלעיל עליכם להדפיס ל־`stderr` את הפלט:

```
ERROR\n
```

כאשר "`\n`" מסמן ירידת שורה. לאחר פקודה זו, עליכם לסגור באופן מיידי את התוכנית עם קוד סיום `EXIT_FAILURE`.

2.2 מיקום איברי $g[t]$ ו־$h[t]$ במרכז המערך

לאחר קבלת הקלט, עליכם **למרכז** את הקלט שהוזן ל־$g[t], h[t]$ כך שבכל מקום "מיותר" נציב 0. את מרכוז האיברים נבצע על ידי חישוב נקודת האמצע, לפי הערך התחתון השלם.
נביט בדוגמה הבאה: נניח שכמות האיברים המקסימלית המותרת הייתה 10 ושהמשתמש הזין 4 ערכים - $1, 2, 3, 4$. אם כך, **ללא** פעולת המרכוז היינו מקבלים את המערך הבא:

$$[1, 2, 3, 4, 0, 0, 0, 0, 0, 0]$$

לאחר שנמרכז את ארבעת הערכים במערך לפי הערך השלם התחתון, נקבל את המערך הבא:

$$[0, 0, 0, 1, 2, 3, 4, 0, 0, 0]$$

2.3 נורמליזציה

לצורך פשטות הדברים, נרצה להציג בתוצאת הניסוי שלנו אך ורק את אחוז העוצמה, מתוך סך העוצמה שנמדדה לאורך כל הניסוי (כלומר לא את הערכים הנומינלים). לשם כך נבקש לנרמל את תוצאת הקונבולוציה בכל צעד.
נורמליזציה (normalization, נירמול) היא התאמה של ערכים הפרוסים על סקלה מסויימת (אפילו על כל $\mathbb{R}$) לסולם ערכים קבוע בקטע $[0,\,1]$. הרשימה המנורמלת עשויה להכיל, אפוא, ערכים שונים (שיהיו בטווח $[0,\,1]$) מהערכים המקוריים, אך היחס שביניהם ישאר זהה.
ישנן מספר שיטות לנרמול נתונים. לשם הנוחות, אנו ננרמל בתרגיל זה את הערכים בשיטת L1. לפי שיטה זו, בהינתן וקטור $\vec{x}$, שגודלו $n \in \mathbb{N}$, נוכל לנרמל את ערכיו כך:

$$\forall i \in \mathbb{N} \;\; 0 \leq i \leq n \quad x_i = \frac{x_i}{\sum_{k=0}^{n} x_k}$$

כזכור, איננו מקבלים בקלט ערכים שליליים, לכן נוכל לנרמל מבלי להשתמש בערך מוחלט.

2.4 מימוש הקונבולוציה

עתה נבצע את הניסוי. עליכם להחיל את הקונבולוציה של $g[t]$ ו־$h[t]$ n־פעמים. למשל:

$$n = 3 \quad \Rightarrow \quad result = (((g * h) * h) * h)\,(t)$$

בכל צעד נבצע קונבולוציה דיסקרטית על f ו־g (כלומר, נחשב את $(f * g)$) כך:

$$(f * g)[t] = \sum_{m=-\lceil \frac{M}{2} \rceil}^{\lfloor \frac{M}{2} \rfloor} f[t-m] \cdot g[m]$$

מפני שמדובר במערכים, הרי שאין לנו אינדקס שלילי. לכן, תצטרכו לעשות התאמת אינדקס, מתיחה, או בעגה מקצועית – פרמטריזציה אפינית:

$$\phi_f(t) = t + \left\lfloor \frac{N}{2} \right\rfloor, \quad \phi_g(t) = t + \left\lfloor \frac{M}{2} \right\rfloor$$

כלומר, נקבל שעבור מערכי־היצוג של הפונקציות $(f,\,g)$ שיסומנו ב־$f_{arr},\,g_{arr}$, ערך הקונבולוציה $(f * g)\,(t)$ (במקומות בהם היא מוגדרת) יחושב כדלקמן:

$$(f * g)[t] = \sum_{m=-\lceil \frac{M}{2} \rceil}^{\lfloor \frac{M}{2} \rfloor} f_{arr}[\phi_f(t-m)] \cdot g_{arr}[\phi_g(m)]$$
$$S.t. \quad 0 \leq \phi_f(t) \leq N \quad \wedge \quad 0 \leq \phi_g(t) \leq M$$

כאשר את t נחשב בטווח $\{\lceil \frac{N}{2} \rceil - 1, ..., 0, ..., \lfloor \frac{N}{2} \rfloor - 1\}$, כדי למנוע בעיות הזחה ($shifts$).

שימו לב:

- קונבולוציה אינה מוגדרת על רשימות ריקות.
- בסיום כל קונבולוציה, עליכם לנרמל את התוצאה.
- יש לבצע את הקונבולוציה n פעמים. אם $n = 0$, אזי אין לבצע את הקונבולוציה כלל.
- **על האלגוריתם שתממשו לעמוד בדרישת זמן ריצה מקסימלי של** $O(n^2)$ (האם ישנו פיתרון יעיל יותר? האם תוכלו לממש אותו? בונוס מובטח למצליחים ☺).

2.5 פלט

כעת, נרצה להדפיס את תוצאות הניסוי. נניח ש־$r[t]$ מייצג את מערך התוצאה של n פעולות הקונבולוציה על $g[t]$ ו־$h[t]$. על התוכנית להדפיס את ההיסטוגרמה (Histogram)[2] של $r[t]$. בכל איטירציה על r נדפיס את האיבר ה־i (קרי, $r[i]$), נקודותיים, רווח יחיד, ורשימה של כוכביות שמייצגות את היחס של $r[i]$ אל מול שאר הערכים ב־r (כלומר **המשקל** של $r[i]$ ביחס לכל איברי המערך). כמות הכוכביות תחושב באופן שבו $max\{r\}$ יקבל 20 כוכביות ובהתאם אליו יקבעו כמות הכוכביות שינתנו לכל ערך אחר. נוכל לחשב את יחס זה כך:

$$\forall 0 \leq i < |r| \quad numberOfStars(r[i]) = \left\lfloor \frac{r[i]}{max\{r\}} \cdot 20 \right\rfloor$$

על פלט התוכנית לעמוד בהנחיות הבאות:

- **לפני** ההדפסה והחישוב של $max\{r\}$, עליכם לעגל את כל איברי $r[t]$ לפי 3 נקודות לאחר הנקודה העשרונית. כלומר, יש לחשב זאת כך: $\forall v \in r[t] \;\; v = \frac{round(1000 \cdot v)}{1000}$.
- יש לחשב את $max\{r\}$ **לפני** שמחשבים את מספר הכוכביות. **אין** לעגל את $max\{r\}$.
- נשים לב שיש לחלק את $r[i]$ ב־$max\{r\}$. כביכול, הבדיקה $max\{r\} \neq 0$ לפני החלוקה אמורה להבטיח שאיננו מחלקים ב־0 – אלא שהמצב אינו כך:
  שכפי שנלמד בשיעורים, מספרים עשרוניים (float, double) נשמרים בזיכרון בייצוג בינארי. הדרך בה אלו מיוצגים מגבילה אותנו, בין היתר, **בכמות** הספרות שניתן לכלול לאחר הנקודה העשרונית. בפרט, לא ניתן לייצג מספרים הקטנים מ־ε כלשהו, התלוי בסוג מבנה הנתונים ובגרסת המחשב (32 או 64 ביט). במקרים בהם חילקנו במספר קטן מ־ε, החלוקה בו תיחשב **כאילו חילקנו ב־0** ולפיכך תייצר שגיאה. סוג שגיאות אלו נקרא Division Underflow. כדי לפתור קושי זה ננהג כך: לצורך תרגיל זה יהי $\varepsilon = 0.00000001$. ההיסטוגרמה תודפס אם"ם $max\{r\} > \varepsilon$, אחרת לא יודפס דבר.
- כל ערך יודפס עם הפורמט ''%0.3f''. אנא השוו את הפלט שאתם מקבלים לזה של פתרון בית הספר. הדפסה בפורמט שונה תביא לחוסר תאימות עם הבדיקה האוטומטית **ותגרור גריעת נקודות משמעותית מציונכם.**

3 דוגמה

ניתן לסכם את התרגיל בדוגמת הריצה הבאה של DrumExperiment:

[2] http://mathworld.wolfram.com/Histogram.html

```
$ ./DrumExperiment
3 4 5 6 5
1 2 2
4
...
0.029: ***
0.065: ******
0.115: ************
0.163: *****************
0.189: ********************
0.180: ********************
0.136: **************
0.077: ********
0.029: ***
...
```

כאשר שורה שנפתחת ב־$ מסמנת את הפקודה שהוקלדה, השורות בירוק מסמנות קלט מהמשתמש ושלוש הנקודות שמסומנות באדום מציינות שורות רבות שכולן הן הפלט `0.000:`‘‘ ‘‘. למען הסר ספק, **עליכם לכלול גם את שורות אלו, כלומר את** ‘‘ `0.000:`‘‘**, בפלט**).

4 נהלי הגשה

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס.
- כתבו את כל ההודעות שבהוראות התרגיל בעצמכם. העתקת ההודעות מהקובץ עלולה להוסיף תווים מיותרים ולפגוע בבדיקה האוטומטית, המנקדת את עבודתכם.
- בתרגיל זה, בנוסף לאיסור על שימוש ב־VLA, גם **חל איסור על הקצאת זיכרון דינמית**.
- בתרגיל חל איסור על שימוש בפונקציות מובנות הממירות מחרוזת למספר, כדוגמת `strtod, strtof` וכו'. עליכם לממש את המרת המחרוזת ל־double באופן עצמאי.
- פתרון בית הספר זמין בנתיב:

  `~proglab/www/c_ex1/DrumExperiment`

- עליכם ליצור קובץ tar הכולל **אך ורק** את הקובץ DrumExperiment.c וה־README שכתבתם (בפורמט הנדרש בהנחיות להגשת תרגילים). ניתן ליצור tar כדרוש על ידי:

  ```
  $ tar -cvf c_ex1.tar README DrumExperiment.c
  ```

  שימו לב: `DrumExperiment.c` יכלול את מלוא התרגיל, לרבות ה־main. על הקובץ להתקמפל כהלכה עם C99, כנדרש בהוראות להגשת תרגילים שפורסמו באתר הקורס.
- אנא וודאו כי התרגיל שלכם עובר את ה־Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה־Pre-submission Script זמין בנתיב.

  `~proglab/www/c_ex1/presubmission`

בהצלחה!!