

שפת C – תרגיל 1

היכרות עם השפה, preprocessor, compiler, משתנים, לולאות, תנאים, פונקציות, קלט/פלט, טיפול בשגיאות

תאריך הגשה של התרגיל והבחון: יום ראשון 06.08.17 עד שעה 23:55¹

1. הנחיות חשובות:

1. בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
2. בכל התרגילים עליכם לכתוב קוד ברור. בכל מקרה בו הקוד שלכם אינו ברור מספיק עליכם להוסיף הערות הסבר בגוף הקוד. יש להקפיד על תיעוד (documentation) הקוד ובפרט תיעוד של כל פונקציה.
3. במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
4. עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
5. בכל התרגילים במידה ויש לכם הארכה, או שאתם מגישים באיחור. **חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק overdue טרם נפתח).** מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.
6. אין להגיש קבצים נוספים על אלו שתדרשו. ובפרט אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש).
7. עליכם לקמפל עם הדגלים `Wall -Wextra -Wvla -std=c99` ולוודא שהתוכנית מתקמפלת ללא אזהרות, תכנית שמתקמפלת עם אזהרות תגרוור הורדה משמעותית בציון התרגיל. למשל, בכדי ליצור תוכנית מקובץ מקור בשם `ex1.c` יש להריץ את הפקודה:
`gcc -Wextra -Wall -Wvla -std=c99 ex1.c -o ex1`
8. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקווריום, לוי, השרת river). **חובה להריץ את התרגיל במחשבי בית הספר לפני ההגשה.** (ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת bit-64 באמצעות הפקודה `"uname -a"` ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב `x86_64`)
9. לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מהpresubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן על מנת שלא לאבד נקודות.
10. **שימו לב ! תרגיל שלא יעבור את הpresubmission script ציונו ירד משמעותית** (הציון יתחיל מ-50, ויוכל לרדת) **ולא יהיה ניתן לערער על כך.**
11. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה. במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. **במהלך הבדיקה הקוד שלכם ייבדק מול קליטים נוספים לשם מתן הציון.**
12. **הגשה מתוקנת** - לאחר מועד הגשת התרגיל ירוצו הבדיקות האוטומטיות ותקבלו פירוט על הטסטים בהם נפלתם. לשם שיפור הציון יהיה ניתן להגיש שוב את התרגיל לאחר תיקוני קוד ולקבל בחזרה חלק מהנקודות - **פרטים מלאים יפורסמו בפורום ואתר הקורס.**

¹ ניתן להגיש באיחור עם קנס של 10 נקודות.

2. הנחיות ספציפיות לתרגיל זה:

1. חל איסור להשתמש במערכים בגודל דינמי (VLA).
2. עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להשתמש בתוכנת valgrind (ראו פירוט בהמשך).
3. אתם רשאים להשתמש בכל הספריות הסטנדרטיות של C.
4. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
5. אינכם רשאים להשתמש בפקודה scanf שהיא איננה פונקציה בטוחה ואינכם יכולים להשתמש בפונקציות atol, atoi, atof, atofl ודומותיהן משום שלא תמיד ניתן לדעת אם הפונקציה הצליחה.
6. שימו לב שאתם מכירים כל פונקציה בה אתם משתמשים ושאתם בודקים עבור כל פונקציה שהיא הצליחה.

3. מסלול ארנסטוף (Arenstorf):

ב-4 לאוקטובר 1957, שיגרה בריה"מ את ספוטניק 1 הלוויין המלאכותי הראשון שהגיע למסלול, כך נפתח שלב נוסף במלחמה הקרה, "המרוץ לחלל". 12 שנים מאוחר יותר, ב-21 ליולי 1969, צעד ניל ארנסטורנג את הצעד הראשון של אדם על הירח ושינה את ההיסטוריה והעתיד האנושי לנצח.

אבל איך הוא עשה את זה?

אחת הבעיות הקשות בשיגור לוויין ובפרט חללית לחלל, היא כמות הדלק שהחללית צריכה לשאת עמה. לכן נרצה למצוא את המסלול בעל האנרגיה המינימלית למסע בין כדור הארץ לשמש. הבעיה היא שבועד שמסלולים של שני גופים (לוויין וגוף מאסיבי) נוסחו על ידי קפלר ונפתרו תיאורטית ע"י ניוטון. הבעיה התלת-גופית (לוויין ושני גופים מאסיביים) מסובכת יותר. כאן נכנס לתמונה ריצ'ארד ארנסטוף (Richard Arenstorf) שמצא פתרון לבעיה זו בדמות "מסלולי ארנסטוף", מסלול תנועה לחללית/לוויין סביב שני הגופים, כזה שגם מחזיר את האסטרונאוטים הביתה במקרה של תקלה (כמו באפולו 13). מסלולים אלו שימשו בזמן המרוץ לחלל ומשמשים עד היום לתנועת חלליות בחלל. בתרגיל זה נחשב מסלול כזה.

1. נחשב את המסלול על ידי חישוב התאוצה הפועלת על החללית בכל נקודה על פי שיטת ארנסטוף.

עבור חללית במיקום (x, y) (כל הבעיה פה תתואר במישור אחד) ובמהירות (v_x, v_y) .

- a. נגדיר α להיות יחס המסות ירח וכדור"א (במקרה שלנו, יחס המסות הוא 0.012299) ו-

$$\beta = 1 - \alpha$$

- b. בנוסחה בסעיף הבא מחושבים בנקודה ספציפית באופן הבא:

$$D_2 = \left[(x - \beta)^2 + y^2 \right]^{3/2}, D_1 = \left[(x + \alpha)^2 + y^2 \right]^{3/2} \quad \blacksquare$$

- c. נקבל את התאוצה בנקודה ספציפית:

$$a_x = x + 2v_y - \beta \frac{x+\alpha}{D_1} - \alpha \frac{x-\beta}{D_2} \quad \blacksquare$$

$$a_y = y - 2v_x - \beta \frac{y}{D_1} - \alpha \frac{y}{D_2} \quad \blacksquare$$

- d. שימו לב, סעיפים b,c נכונים לנקודה ספציפית! ומחושבים בכל נקודה.

2. כעת בהינתן התאוצה נוכל לקרב את המסלול באמצעות שיטת אוילר קדמית (Forward Euler)

- a. עבור התקדמות בפרק זמן dt , המיקום המשוער של החללית תהיה ב- $x_{(t+dt)} = x_t + v_{x,t} \cdot dt$, כלומר

אנחנו מבצעים קירוב לינארי, המיקום של החללית יהיה המיקום הקודם שלו פלוס תנועה בקו ישר

בכיוון המהירות שלו כפול פרק הזמן. באותו אופן שאר הערכים של החללית יהיו:

$$y_{(t+dt)} = y_t + v_{y,t} \cdot dt \quad \blacksquare$$

$$v_{x,(t+dt)} = v_{x,t} + a_{x,t} \cdot dt \quad \blacksquare$$

$$v_{y,(t+dt)} = v_{y,t} + a_{y,t} \cdot dt \quad \blacksquare$$

- b. כעת על ערכים אלו נחשב שוב את התאוצה ובעזרתה נחשב שוב פעם מיקום ומהירות חדשים בקפיצות של dt . על תהליך זה נחזור מספר רב של פעמים עד לפרק זמן של T כלשהו ואז נוכל לדעת את מיקום החללית לאחר T שניות.
- c. שימו לב שככל ש- dt קטן יותר, כך השגיאה במסלול קטנה יותר. בנוסף, ככל שמבצעים יותר פעולות euler השגיאה גדלה גם כן. לפרטים נוספים: [euler method](#).

4. הנחיות Ex1:

1. כתבו קוד המחשב את מסלול ארנסטוף עבור חללית הנמצאת במיקום x, y ובמהירות V_x, V_y , למשך T זמן, כאשר T מחולק ל- n צעדים של חישוב, ו- m צעדים יודפסו לקובץ output (כלומר כל $n \bmod m == 0$ תדפיסו את הערך, m מחלק את n)
2. הפלט לתוכנית יודפס לקובץ הפלט כאשר רק m שלבים מתוכו יודפסו

$\langle x_1 \rangle, \langle y_1 \rangle, \langle x_2 \rangle, \langle y_2 \rangle, \dots, \langle x_m \rangle, \langle y_m \rangle \backslash n$

ניתן לראות דוגמא בקבצים המצורפים לתרגיל

3. את התוכנית יהיה ניתן להריץ בצורות הבאות:

$ex1 \langle input \text{ file} \rangle \langle output \text{ file} \rangle$

$ex1 \langle output \text{ file} \rangle$

- a. במקרה הראשון על התוכנית לקרוא את הנתונים מקובץ:
הנתונים בקובץ יופיעו בפורמט הבא:

$\langle x, y, V_x, V_y \rangle \backslash n$

$\langle T, n, m \rangle \backslash n$

- b. במקרה השני, הקלט יגיע מהמשתמש באמצעות המקלדת.
עליכם לקרוא את הנתונים אחד אחרי השני, ולהדפיס הודעה בשורה חדשה לפני כל אחד מהם.
למשל:

"Enter initial pos x:\n"

0.994

"Enter initial pos y:\n"

0

"Enter initial vel x:\n"

0

"Enter initial vel y:\n"

-2.0016

"Enter total time T:\n"

17.0652

"Enter num of steps:\n"

8

"Enter num of steps to save:\n"

2

5. בתרגיל זה אינכם רשאים להניח כי הקלט תקין ועליכם להדפיס הודעת שגיאה עבור כל קלט שאינו תקין ולהחזיר ערך החזרה שאינו 0 (כישלון). אין צורך כי הודעת השגיאה תהיה זהה לשלנו, אבל עליה להיות אינפורמטיבית ולהישלח ל- `stderr`
6. שימו לב כי שימוש נכון ב- IO הוא העתקה של N תווים מהקלט למערך בגודל N כלשהו ורק לאחר מכן שימוש בנתונים. בשפת C ניתן לרשום מעבר לגודל המערך ולכן שימוש אחר יוביל ל- `buffer overflow` והתנהגויות לא תקינות ולא צפויות, מעניינות מאוד וקשות מאוד למציאה בשלב הדיבוג.
7. בבעיות בקריאת הקלט (למשל בעיות IO, שגיאות מערכת), על התוכנית להדפיס הודעה אינפורמטיבית ולהסגר באופן מיידי. בבעיות בתקינות הלוגית של הקלט (למשל זמן שלילי), על התוכנית לקרוא את כלל הקלט לפני שתדפיס הודעה אינפורמטיבית ותצא.
8. נזכיר בשלב זה כי תוכניות שמסיימות את ריצתן בהצלחה, יוצאות עם `exit code` של 0 (קרי, `return 0` בסוף ה-`main`). כאשר תוכניות אלו לא מסיימות את ריצתן בהצלחה, הן יוצאות עם `exit code` שונה מ-0, שבדרך"כ מתאר את מהות השגיאה.
9. שימו לב שהיות ותרגיל זה עוסק בחישובים נומריים והיות ונרצה למזער את השגיאה, נרצה להשתמש בטיפוסים מסוג `long double` ו- `long`, שימו לב שאתם משתמשים שימוש נכון בטיפוסים הנכונים במקומות הנכונים.
10. שימו לב, מיקום ומהירות יכולים להיות שליליים, פרק זמן כנראה שלא...
11. בנוסף, עליכם להריץ את התוכנה על שני קבצי ה- `input` שניתן לכם ולהציג את הפלט, מסלול החללית, ע"י הצגתו בגרף. אתם יכולים לייצר את הגרף במטלאב, פייתון או אקסל. דרך פשוטה להכין את הגרף בפייתון ניתנת באופן הבא:
 - a. `import matplotlib.pyplot as plt`
 - b. `import numpy as np`
 - c. `a = [<x1>, <y1>, <x2>, <y2>,]`
 - d. `x,y = np.array(a).reshape((len(a)/2, 2)).transpose()`
 - e. `plt.plot(x,y)`

5. עבודה עם `valgrind`:

4. ניהול זיכרון ב-C הוא נושא רגיש ומועד לפורענות – יש הרבה אפשרויות לטעות (לא להקצות מספיק זיכרון, לשכוח לשחרר זיכרון, להשתמש במצביעים שמצביעים לזבל וכו'). כמובן שהמהדר לא ידווח על שגיאה בכל המקרים הללו. יתכן שתגלו את השגיאות הללו בזמן ריצה, אך יתכן גם כי התוכנה תעבוד אצלכם "במקרה" והבעיות יתגלו דווקא בביתו של הלקוח.
5. ישנו מבחר די גדול של תוכנות בשוק שמטרתן לסייע באיתור בעיות זיכרון בקוד לפני שחרורו אל הלקוח. אנו נשתמש בתוכנת `valgrind`, שיחסית לתוכנה חנימית, נותנת תוצאות מעולות. אתם מתבקשים להריץ את `valgrind` עם התוכנה שלכם, ולהגיש את הפלט שלה בקובץ בשם `valdbg.out`.
6. כדי להריץ את `valgrind` עליכם לבצע קומפילציה ו-`linkage` לקוד שלכם עם הדגל '-g' (הן בשורת הקומפילציה והן בשורת ה-`linkage`). לאחר מכן הריצו `valgrind`:


```
> valgrind --leak-check=full --show-possibly-lost=yes
--show-reachable=yes --undef-value-errors=yes ProgramName
```
7. אם קיבלתם הודעת שגיאה, יתכן שתצטרכו לבצע שינוי הרשאות:


```
> chmod 777 ProgramName
```
8. כמובן שאם `valgrind` דיווח על בעיות עם הקוד שלכם, עליכם לתקן אותן.
9. היעזרו ב-`tutorial` הקצרצר של `valgrind` שבאתר הקורס.

6. הערות למשימות התכנות:

1. התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של פתרונותיכם ישווה (השוואת טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות.
2. במידה וסיפקנו לכם קבצי קלט/פלט לדוגמה (אלו מהווים רק חלק קטן מקבצי הקלט-פלט שנשתמש בהם, כתבו לעצמכם בדיקות נוספות), עליכם לוודא שהתכנית שלכם נותנת את אותו הפלט בדיוק.
3. על מנת לעשות זאת הריצו את תכניתכם עם הקלט לדוגמה (באמצעות האופרטור "<") ונתבו את הפלט של תכניתכם לתוך קובץ (באמצעות האופרטור ">") באופן הבא:

```
ProgramName < inputFile > myOutputFile
```

4. ואז השוו את קובץ הפלט שנוצר לכם עם קובץ הפלט המתאים של פתרון בית הספר, באמצעות הפקודה

```
diff
```

הנה תוכנה להשוואה טקסטואלית של שני טקסטים שונים. בהינתן שני קבצי טקסט להשוואה (1.txt, 2.txt) הפקודה הבאה תדפיס את השורות אשר אינן זהות בשני הקבצים:

```
diff 1.txt 2.txt
```

במידה והקבצים זהים לחלוטין, לא יודפס דבר.
קראו על אפשרויות נוספות של diff בעזרת הפקודה `man diff`.
לחלופין אתם יכולים גם להשתמש בתוכנה `tkdiff` אשר מראה גם את השינויים ויזואלית.

5. כמו כן, אתם יכולים גם להשוות ישירות באופן הבא:

```
ProgramName < inputFile | diff expected.out
```

6. אם ישנם מקרים שהוראות התרגיל לא מציינות בבירור כיצד התכנית צריכה להתנהג, הביטו בקבצי הקלט וקבצי הפלט לדוגמה שניתנים לכם ובדקו אם התשובה לשאלתכם נמצאת שם. כמו כן, היעזרו בפתרון בית הספר, הריצו עליו את הטסטים שלכם והשוו להתנהגות תוכניתכם.

חומר עזר:

1. את הקבצים המסופקים לצורך התרגיל ניתן למצוא ב:

```
~plabc/www/ex1/ex1_files.tar
```

2. דוגמאות לקבצי קלט/פלט ניתן למצוא ב:

```
~plabc/www/ex1/ex1_examples.tar
```

3. מידע לגבי מצביעים לפונקציות ב-C ניתן למצוא ב:

```
http://www.newty.de/fpt/index.html
```

7. הגשה:

1. עליכם להגיש קובץ `tar` בשם `ex1.tar` המכיל לפחות את הקבצים הבאים:

- `Ex1.c`
- `output1e3.txt` - קובץ הפלט המתקבל מריצת התוכנה שלכם עם קובץ הקלט `input1e3.txt`
- `output1e9.txt` - קובץ הפלט המתקבל מריצת התוכנה שלכם עם קובץ הקלט `input1e9.txt`
- `graphs.pdf` - קובץ pdf עם 2 גרפים של המסלולים המתקבלים מריצת שני הקבצים הקודמים
- `extension.pdf` - בק במקרה שההגשה היא הגשה מאושרת באיחור (מכיל את האישורים הרלוונטיים להארכה).
- שימו לב! - על אף שאתם יכולים להוסיף קבצים נוספים כרצונכם, הימנעו מהוספת קבצים לא רלוונטיים (גם בכדי להקל על הבודקים, וגם בכדי שציונכם לא יפגע מכך).

2. לפני ההגשה, פתחו את הקובץ ex1.tar בתיקה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות.

3. מומלץ מאוד גם להריץ בדיקות אוטומטיות וטסטרים שכתבתם על הקוד אותו אתם עומדים להגיש. בנוסף, אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

```
~plabc/www/codingStyleCheck <file or directory>
```

כאשר <directory or file> מוחלף בשם הקובץ אותו אתם רוצים לבדוק או תיקייה שיבדקו כל הקבצים הנמצאים בה (שימו לב שבדיקה אוטומטית זו הינה רק חלק מבדיקות ה codingStyle)

4. דאגו לבדוק לאחר ההגשה את קובץ הפלט (submission.pdf) וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

```
~plabc/www/ex1/presubmit_ex1
```

בהצלחה!