סדנת תכנות בשפת C++, מס׳ קורס 67317 - 2017

תרגיל 1

היכרות עם השפה, classes, const , references,
dynamic allocation, operators overloading

תאריך הגשה של התרגיל והבוחן:

1. הנחיות כלליות:

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 כלשהו 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=c++17 -pthread -g ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.cpp יש להריץ את הפקודה:

g++ -Wextra -Wall -Wvla -std=c++17 -pthread -g ex1.cpp -o ex1

8. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקוריום, 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.
שימו לב ! תרגיל שלא יעבור את ה presubmission script ציונו ירד משמעותית (הציון יתחיל מ-50, ויוכל לרדת) **ולא יהיה ניתן לערער על כך.**
10. 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 קלטים נוספים לשם מתן הציון.**
11. **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.

2. הנחיות חשובות לכלל התרגילים בקורס ++C

1. הקפידו להשתמש בפונקציות ואובייקטים של ++C (למשל new, delete, cout) על פני פונקציות של C (למשל malloc, free, printf).
בפרט השתמשו במחלקה string (ב-std::string) ולא במחרוזת של C (char *).
2. יש להשתמש בספריות סטנדרטיות של ++C ולא של C אלא אם כן הדבר הכרחי (וגם אז עליכם להוסיף הערה המסבירה את הסיבות לכך).
3. הקפידו על עקרונות Information Hiding – לדוגמא, הקפידו כי משתני המחלקות שלכם מוגדרים כמשתנים פרטיים (private).
4. הקפידו לא להעתיק by value משתנים כבדים, אלא להעבירם (היכן שניתן) by reference.
5. הקפידו מאוד על שימוש במילה השמורה const בהגדרות המתודות והפרמטרים שהן מקבלות:
המתודות שאינן משנות פרמטר מסויים – הוסיפו const לפני הגדרת הפרמטר.
מתודות של מחלקה שאינן משנות את משתני המחלקה – הוסיפו const להגדרת המתודה.
שימו לב: הגדרת משתנים / מחלקות ב- ++C כקבועים הוא אחד העקרונות החשובים בשפה.
6. הקפידו על השימוש ב- static, במקומות המתאימים (הן במשתנים והן במתודות).
7. הקפידו לשחרר את כל הזיכרון שאתם מקצים (השתמשו ב-valgrind כדי לבדוק שאין לכם דליפות זיכרון).
8. שימו לב שהאלגוריתמים שלכם צריכים להיות יעילים.
9. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.

1. הנחיות ספציפיות לתרגיל זה:

1. עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להשתמש בתוכנת valgrind (ראו פירוט בהמשך).
2. חל איסור להשתמש במבני נתונים מוכנים בתרגיל (כדוגמת STL) שימוש כזה יוביל לפסילת הסעיף הרלוונטי.
3. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
4. שימו לב שאתם מכירים כל פונקציה בה אתם משתמשים ושאתם בודקים עבור כל פונקציה שהיא הצליחה.

קצת רקע - 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 ארמסטרונג את הצעד הראשון של אדם על הירח ושינה את ההיסטוריה והעתיד האנושי לנצח.

אבל איך הוא עשה את זה?

אחת הבעיות הקשות בשיגור לווין 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 שבעוד שמסלולים של שני גופים (לווין וגוף מאסיבי) נוסחו על ידי קפלר ונפתרו תיאורטית ע״י ניוטון. הבעיה התלת-גופית (לווין ושני גופים מאסיביים) מסובכת יותר. כאן נכנס לתמונה ריצ׳ארד ארנסטוף (Richard Arenstorf) שמצא פתרון לבעיה זו בדמות ״מסלולי ארנסטוף״, מסלול תנועה לחללית/לווין סביב שני הגופים, כזה שגם מחזיר את האסטרונאוטים הביתה במקרה של תקלה (כמו באפולו 13). מסלולים אלו שימשו בזמן המירוץ 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 וכדוה״א (במקרה שלנו, יחס המסות הוא 0.012299) ו- $\beta = 1 - \alpha$
   b. D_1, D_2 בנוסחה בסעיף הבא מחושבים בנקודה ספציפית באופן הבא:
      - $D_2 = [(x-\beta)^2 + y^2\ ]^{3/2}\ ,\ D_1 = [(x+\alpha)^2 + y^2\ ]^{3/2}$
   c. נקבל את התאוצה בנקודה ספציפית:
      - $a_x = x + 2v_y - \beta\frac{x+\alpha}{D_1} - \alpha\frac{x-\beta}{D_2}$
      - $a_y = y - 2v_x - \beta\frac{y}{D_1} - \alpha\frac{y}{D_2}$
   d. שימו לב, <u>סעיפים b,c נכונים לנקודה ספציפית! ומחושבים בכל נקודה.</u>
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:
      - $y_{(t+dt)} = y_t + v_{y,\,t} \cdot dt$
      - $v_{x,\,(t+dt)} = v_{x,\,t} + a_{x\,,t} \cdot dt$
      - $v_{y,\,(t+dt)} = v_{y,\,t} + a_{y,\,t} \cdot dt$
   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.

הנחיות מסלולים לירח:

1. <u>הדרכה והנחות לתרגיל:</u>
   - בתרגיל זה תכתבו 2 מחלקות (ArenstorfPoint,Arenstorf) וקובץ הרצה (ex1) עבור התרגיל.
   - כתבו קוד המחשב את מסלול ארנסטוף עבור חללית הנמצאת במיקום x,y ובמהירות Vx, Vy, למשך T זמן, כאשר T מחולק ל-n צעדים של חישוב, ו- m צעדים יודפסו לקובץ output (כלומר כל n mod m == 0 תדפיסו את הערך, m מחלק את n)
   - 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.
   - זכרו לסגור כל קובץ שאתם פותחים.
   - שימו לב, מיקום ומהירות יכולים להיות שליליים, פרק זמן כנראה שלא...
   - 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:
     - import matplotlib.pyplot as plt
     - import numpy as np
     - a = [<x1>, <y1>, <x2>, <y2>, ….]
     - x,y = np.array(a).reshape(np.round((len(a)/2), 2)).transpose()
     - plt.plot(x,y)

2. <u>דוגמאות הרצה:</u>
   - הפלט לתוכנית יודפס לקובץ הפלט כאשר רק m שלבים מתוכו יודפסו

<x1>, <y1>, <x2>, <y2>, …., <xm>, <ym>\n

ניתן לראות דוגמא בקבצים המצורפים לתרגיל

- את התוכנית יהיה ניתן להריץ בצורות הבאות:

ex1 <input file> <output file>

ex1 <output file>

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

<x, y, Vx, Vy>\n

<T, n, m>\n

- 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

3. <u>את תרגיל זה אנו נממש במחלקות שכוללות את הפונקציות הבאות:</u>

<u>Class Arenstorf:</u>

- `static int computeArenstorf(const ArenstofPoint& init, unsigned int n, unsigned int m, long double dt, std::ostream& out)`
  - ➢ which will compute the trajectory points and write it to the output stream
  - ➢ Returns 0 on success, -1 on an error

<u>Class ArenstorfPoint:</u>

- `ArenstorfPoint(long double x , long double y, long double vx, long double vy)`
  - ➢ inits the class with the x, y coordinates, vx, and vy
- Operator <<(ostream &os, const ArenstorfPoint& a) : <u>friend method</u>
  - ➢ Writes to the os a string representation of the point in the format x, y
- Getter functions for x, y, vx, vy

<u>Ex1.cpp :</u>

- בנוסף למחלקות עליכם לממש קובץ הרצה בשם Ex1 עם פונקציית main המריץ את הקוד כפי שמתואר לעיל. עם הקלט של הנתיב לקבצי הקלט\פלט.
- השתדלו לשמור על קוד מינימלי בקובץ זה - כל קוד השייך לחישוב הoutput אמור להימצא באחת מהמחלקות. הקוד של main צריך לקרוא את הקלט ולקרוא לפוקציה computeArendof של מחלקה Arendof
- תשתמשו ב streams עבור קלט ופלט.
- תוכלו להוסיף עבור כל מחלקה מתודות ומשתנים משלכם. רק זכרו לשמור על כללי OOP. (מותר להוסיף גם מתודות פומביות).

4. <u>הנחות קלט וטיפול בשגיאות:</u>

- 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- cerr
- שימו לב כי שימוש נכון ב- IO הוא העתקה של N תווים מהקלט למערך בגודל N כלשהו ורק לאחר מכן שימוש בנתונים.

- 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.
- 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- cerr.
- 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, שבדר"כ מתאר את מהות השגיאה.

הערות כלליות למשימות התכנות:

1. התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של פתרונותיכם יושווה (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 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. דוגמאות לקבצי קלט/פלט ניתן למצוא ב:

~labcpp/www/ex1/test_examples.tar

הגשה:

1. עליכם להגיש קובץ tar בשם ex1.tar המכיל **<u>לפחות</u>** את הקבצים הבאים:
   - Ex1.cpp
   - Arenstorf.cpp
   - Arenstorf.h

- ArenstorfPoint.cpp
- ArenstorfPoint.h
- 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
- קובץ Makefile התומך לפחות בפקודות הבאות:
  - make ex1 -קימפול ויצירת תוכנית ex1 (ללא בדיקות debug).
  - make - קימפול ויצירת תוכנית ex1 (ללא בדיקות debug).
  - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)

- extension.pdf - רק במקרה שההגשה היא הגשה מאושרת באיחור בקישור ex1_late (מכיל את האישורים הרלוונטים להארכה).
- שימו לב! - על אף שאתם יכולים להוסיף קבצים נוספים כרצונכם, הימנעו מהוספת קבצים לא רלוונטים (גם בכדי להקל על הבודקים, וגם בכדי שציונכם לא יפגע מכך).

2. לפני ההגשה, פתחו את הקובץ ex1.tar בתיקיה 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.

~plabcpp/www/ex1/presubmit_ex1

בהצלחה!