שפת C++ – תרגיל 1

היכרות עם השפה, classes, const , references,

dynamic allocation, operators overloading

תאריך הגשה: יום חמישי 08.09.16 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): מוצאי שבת עד שעה 23:55[1]

תאריך ההגשה של הבוחן: יום חמישי 08.09.16 עד שעה 23:55

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. בכדי שתוכלו להגיש באיחור עם קנס, עליכם לסמן זאת בלינק התרגיל לפני מועד ההגשה.
4. למי שיש אישור להגשה מאוחרת, אין להגיש בקישור להגשה הרגילה. **מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.**
5. אין להגיש קבצים נוספים על אלו שתדרשו.
6. עליכם לקמפל עם הדגלים -Wall -Wextra -pthread -Wvla -std=c++11 ולוודא שהתוכנית מתקמפלת ללא אזהרות, תכנית שמתקמפלת עם אזהרות תגרור הורדה בציון התרגיל. למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.cpp יש להריץ את הפקודה:

g++ -Wextra -Wall -Wvla -pthread -std=c++11 ex1.cpp -o ex1

7. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר (מחשבי האקוריום, לוי, השרת river). בפרט, המחשבים שעליהם מתבצעת הבדיקה, צריכים להיות 64 ביט[2].
8. 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.

---

[1] ניתן להגיש באיחור נוסף, עד יום ראשון בקנס של 20 נקודות.

[2] ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת זו באמצעות הפקודה "uname -a" ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב x86_64.

9. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות עבורו היא אחריותכם. חישבו על מקרי קצה לבדיקת הקוד.
   קבצי בדיקה לדוגמה ניתן למצוא פה: ~slabcpp/www/ex1/tests_examples.tar
   שימוש בקבצים אלו הוא באחריותכם. במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.
10. הגשה מתוקנת - לאחר מועד הגשת התרגיל ירוצו הבדיקות האוטומטיות ותקבלו פירוט על הטסטים בהם התוכנית שלכם נכשלה. לשם שיפור הציון יהיה ניתן להגיש שוב את התרגיל לאחר תיקוני קוד קלים ולקבל בחזרה חלק מהנקודות - פרטים מלאים מופיעים בהנחיות הקורס באתר.

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 סטדנרטיות 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).

3. הנחיות ספציפיות לתרגיל זה:

1. שימו לב שהאלגוריתמים שלכם צריכים להיות יעילים.
2. בתרגיל זה אין להגיש קבצים נוספים.
3. חל איסור להשתמש במבני נתונים מוכנים בתרגיל (כדוגמת STL) שימוש כזה יוביל לפסילת הסעיף הרלוונטי.
4. טיפול בשגיאות ב-C++ מבוצע ע"י מנגנון חריגות (exceptions). אתם תלמדו על הנושא בהמשך הקורס. בתרגיל זה אנו מניחים שאין שגיאות (הקצאות זיכרון מצליחות תמיד, הפרמטרים לפונקציות חוקיים וכו').
5. מומלץ מאוד לקרוא ולהבין היטב את סעיפים 4-5 לפני שאתם מתחילים לממש את התרגיל. תכנון נכון עשוי לחסוך לכם זמן יקר.

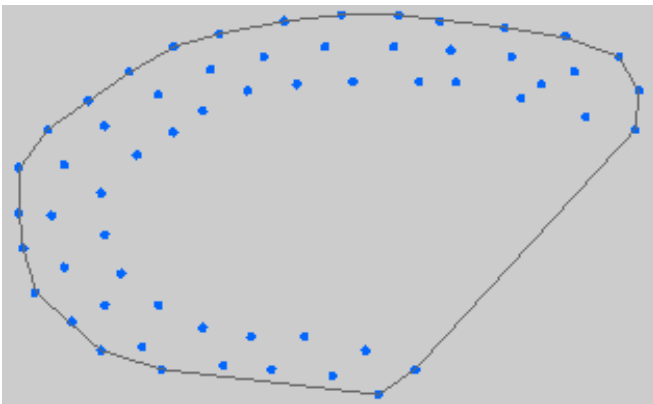
4. קמור - ConvexHull:

1. הקמור של גוף (או אוסף של גופים) הוא הגוף הקמור המינימלי המכיל אותם. קמור של קבוצת נקודות הוא הצורה הקמורה הקטנה ביותר שמכילה את הנקודות.
2. במקרה הפרטי שבו אוסף הגופים הוא אוסף של נקודות, קטעים ומצולעים במישור הדו-ממדי, הקמור הוא החיתוך של כל הקבוצות הקמורות שמכילות את קבוצת הנקודות.
3. לקמירות שימושים ברבים מתחומי המתמטיקה והיא עומדת בבסיסם של אלגוריתמים רבים בגיאומטריה חישובית, זיהוי כתב ותבניות, עיבוד תמונה, תכנון לינארי ועוד.
4. מבחינה אינטואיטיבית ניתן להגדיר את הקמור באופן הבא:

תהי X קבוצת נקודות במרחב הדו מימדי.

נמתח גומייה דמיונית סביב קבוצת הנקודות, הגומיה "תיתפס" במספר נקודות בקצוות, נסמנם ב H(X) .

אוסף נקודות זה מהווה פוליגון קמור המקיף את קבוצות הנקודות X.



שימו לב ש H(X) הינה תת קבוצה של קבוצת הנקודות X.
מטרתנו בתרגיל הינה למצוא את H(X).

5. למציאת הקמור קיימים אלגוריתמים רבים ומגוונים בעלי מאפיינים שונים ומעניינים. בתרגיל עליכם לממש את האלגוריתם Graham's scan.
6. עיינו בקישורים הבאים לצורך הבנת האלגוריתם ואופן פעולתו :
(גם אם קיימים הבדלים מינוריים בין הגרסאות השונות שלהאלגוריתם הינכם רשאים לממש את הגרסה הנוחה לכם).

http://www.gadial.net/2011/10/26/graham_scan/

https://en.wikipedia.org/wiki/Graham_scan

http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/GrahamScan.html

http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/ConvexHull/GrahamScan/grahamScan.htm

http://geomalgorithms.com/a10-_hull-1.html

שימו לב כי האלגוריתם כולל שני שלבים עיקריים:

a. מיון הנקודות לפי זווית.
b. איטרציה על הנקודות לשם בחירת נקודות הקמור מתוכן.

בקישור הבא קיימת הדגמת מציאת הקמור של קבוצת נקודות בצורה וויזואלית:

https://www.youtube.com/watch?v=dw120YcIav8

7. תכניתכם (שתמומש בקובץ ConvexHull.cpp) תקבל כקלט מהמשתמש רצף של קורדינטות (נקודות במרחב הדו-ממדי), ותחשב את הקמור של נקודות אלו.
8. פורמט הקלט הוא:

```
<x num>,<y num>
<x num>,<y num>
...
```

9. פלט התכנית יהיה: המחרוזת "result:\n" ורשימת נקודות המרכיבות את הקמור. כאשר:
   a. כל נקודה בשורה חדשה.
   b. הרשימה ממויינת לפי ציר ה-x (מיון ראשי), וציר ה-y (מיון משני).
10. לדוגמא, עבור הקלט הבא:

```
>ConvexHull
1,100
12,50
1,200
14,30
-12,-45
...
```

פלט התוכנית יהיה:

```
result:\n
-12,-45\n
1,100\n
1,200\n
...
```

11. תכנון ומבנה הנתונים:

(1) ככלל, תכנון התכנית מוטל עליכם מלבד המגבלות הבאות:

(a) על תכניתכם לכלול את המחלקות הבאות :

(i) Point - מייצגת נקודה במרחב.

(ii) PointSet - מייצגת קבוצת נקודות (ללא כפילויות).

(שימו לב! מחלקה זו מכילה גם את הפתרון לסעיף הבא בתרגיל)

(b) המחלקה Point תמומש בקבצים Point.h ו-Point.cpp, ותכלול (לפחות) את השיטות הבאות:

(i) בנאי והורס.

(ii) פונקציית toString שמחזירה string בפורמט הבא:

```
"<x num>,<y num>"
```

(iii) פונקציית set שמקבלת שתי מספרים מטיפוס int :

```
set(<x num>,<y num>)
```

(iv) בנאי שמקבל שני מספרים מטיפוס int:

```
Point(<x num>,<y num>)
```

(c) המחלקה PointSet, תתנהג כקבוצה מתמטית (ללא אברים כפולים), ותמומש בקבצים PointSet.h ו-PointSet.cpp, ותכלול (לפחות) את השיטות הבאות:

(i) בנאי והורס.

(ii) פונקציית toString שמחזירה string בפורמט הבא:

```
"<1st x num>,<1st y num>\n...<(size)st x num>,<(size)st y num>\n"
```

(קבוצה ריקה מחזירה "")

(iii) המתודה add המוסיפה נקודה לסוף הקבוצה. (מחזירה true אמ"מ האיבר אכן נוסף לקבוצה)

(iv) המתודה remove המוציאה נקודה מהקבוצה (מחזירה true אמ"מ האיבר היה קיים בקבוצה). המתודה הזו עשויה להשפיע על סדר האיברים בקבוצה.

(v) המתודה size המחזירה את גודל הקבוצה.

(2) מספר הנקודות איננו מוגבל, באפשרותכם לממש את קבוצת הנקודות באמצעות מערך (שיוגדל במידת הצורך), רשימה מקושרת, או כל מבנה נתונים אחר.

(3) הנכם רשאים להוסיף מחלקות נוספות לתכנית.

(4) בשאלה זו אין להשתמש במבני נתונים מוכנים בתרגיל (כדוגמת STL). אם אתם זקוקים לאחד, ממשו בעצמכם.

(5) הסבירו בקובץ ה- README (באנגלית בלבד) במפורט את מבנה תכניתכם והמחלקות בהן אתם משתמשים.

12. הדרכה והנחיות כלליות:

(1) לחלק זה (4- ConvexHull) של התרגיל פורסם פתרון בי"ס. בכל מקרה של שאלה בדקו כיצד פתרון בית הספר מתמודד איתה. על הפתרון שלכם לתאום את ההתנהגות של פתרון בית הספר.

(2) אתם רשאים להניח כי כל המספרים הינם מספרים שלמים מסוג int, וכי הזנת הקורדינטות תסתיים ב-EOF.

(3) הקלט אינו ממויין, והוא עשוי להכיל כפילויות.

(4) אתם רשאים להניח כי הקלט תקין.

(5) עליכם להתמודד גם עם רשימת נקודות בגודל <= 2 (חשבו כיצד).

(6) במקרה של נקודות הנמצאות על קו ישר ומשתייכות לקמור, עליכם להדפיס את כולן (ולא רק את הקיצוניות).

(7) 0 נקודות זה קלט תקין.

(8) עליכם לפרט בהערה לפני מימוש האלגוריתם:

(a) כיצד פועל האלגוריתם שלכם.

(b) הסבר על מידת הסיבוכיות שלו.

(c) הפנייה למקור האלגוריתם (במידה ולקחתם אותו ממקום כלשהו).

5. מימוש אופרטורים עבור PointSet:

1. עליכם לממש את האופרטורים הבאים עבור המחלקה PointSet:

(1) אופרטור השוואה:

(a) operator!=

(b) operator==

2 PointSet זהות אם הן מכילות את אותן נקודות (אין משמעות לסדר).

למשל:

```
A={(1,2);(2,6);(9,-1)}
B={(2,6);(9,-1);(1,2)}
C={(1,2);(2,4);(9,-1)}
A==B
A!=C
```

ערך ההחזרה של האופרטורים הוא bool.

(2) operator-

הפונקציה תחזיר קבוצה חדשה המכילה את הנקודות מה-קבוצה השמאלית אשר אינם מופיעות ב-קבוצה הימנית[3].

(3) operator &

זהו אופרטור המבצע את פעולת ה-Intersection על PointSets. הפונקציה תחזיר קבוצה חדשה המכילה את כל הנקודות שמופיעות גם בקבוצה השמאלית וגם בימנית [4].

(4) אופטורי השמה:

עליכם לממש פונקציות (בנאי העתקה) ואופרטורים בכדי לתמוך בפעולות השמה.

למשל - "PointSet2=PointSet1".

עליכם לכתוב גם קובץ בשם PointSetBinaryOperations.cpp המכיל פונקצית main ויוצר קובץ הרצה המדגים את השימוש <u>בכל</u> האופרטורים שהוגדרו כאן.

2. הדרכה והנחיות כלליות:

(1) יש לממש את האופרטורים במחלקה עצמה בקבצים PointSet.h ו-PointSet.cpp כפונקציות מחלקה. הבנת החתימות הנדרשות של פונקציות האופרטורים היא חלק מהדרישות.

(2) אתם רשאים לשנות את PointSet.h.

(3) הקפידו שהפונקציות תעבודנה ללא דליפות זיכרון.

(4) הקפידו שלא לשכפל קוד בין האופרטורים השונים.

(5) מסופק לכם קובץ בדיקה בסיסי TestPointSetBinFuncs.cpp.

---

[3] מסומנת בדרך כלל "left\right"

[4] מסומנת בדרך כלל "left∩right"

6. בונוס (3 נקודות):

1. כל סטודנט שימצא שגיאה חדשה בפתרון בית הספר יקבל בונוס של 3 נקודות לציון התרגיל.
2. בכדי לקבל את הבונוס עליכם לפתוח דיון בפורום התרגיל שיכלול את:
   (1) קובץ קלט שגורם לשגיאה.
   (2) קובץ פלט שמכיל את פלט פתרון בית הספר שנוצר מריצתו עם קובץ הקלט.
   (3) הסבר מפורט מהי השגיאה.

7. חומר עזר:

1. את פתרון בית הספר ניתן למצוא ב:

```
~slabcpp/www/ex1/school_sol.tar
```

בדקו את תכניתכם וודאו שהפלטים שלכם זהים לאלה של פתרון בית הספר. אתם יכולים לייצר קבצי קלט רבים נוספים כדי לבדוק מקרים נוספים, ולהשוות את הפלט של התכנית שלכם עם של תלמידים אחרים, או עם הפלט שנוצר כשאתם נותנים את הקלט הזה לקובץ הריצה של פתרון בית הספר.

2. קבצי בדיקה לדוגמא ניתן למצוא ב:

```
~slabcpp/www/ex1/test_files.tar
```

3. ביצוע overloading ב-C++:

http://www.cprogramming.com/tutorial/operator_overloading.html

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B

http://www.cplusplus.com/reference/set/set/operators/

8. הגשה

1. עליכם להגיש קובץ tar בשם ex1.tar המכיל <u>רק</u> את הקבצים הבאים:
   - README
   - Point.h ו-Point.cpp
   - PointSet.h ו-PointSet.cpp
   - ConvexHull.cpp
   - PointSetBinaryOperations.cpp
   - קובץ Makefile התומך (לפחות) בפקודות הבאות:
     - make tar - יצירת קובץ tar בשם ex1.tar, המכיל <u>רק</u> את הקבצים שצריך להגיש בתרגיל.

- make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-makefile.
- make PointSet.o - קימפול, ויצירת PointSet.o.
- make Point.o - קימפול, ויצירת Point.o.
- make PointSetBinaryOperations - ,תבנה את קובץ ההרצה PointSetBinaryOperations.
- make ConvexHull - ,תבנה את קובץ ההרצה ConvexHull.
- הרצת make ללא פרמטרים תבנה את קבצי ההרצה ConvexHull ו-PointSetBinaryOperations, ותריץ את PointSetBinaryOperations.

- extension.pdf - <u>רק</u> במקרה שההגשה היא הגשה באיחור.

2. ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
tar cvf <tar name> <files>
```

3. <u>לפני ההגשה</u>, פתחו את הקובץ בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות. וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

```
~slabcpp/www/ex1/presubmit_ex1
```

4. אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

```
~slabcpp/www/codingStyleCheck <code 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)

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

בהצלחה!