

מכללת הדסה, החוג למדעי המחשב

מבוא לתכנות מונחה עצמים והנדסת תוכנה

סמסטר א', תשע"ט

תרגיל 1

תאריך אחרון להגשה:

יום ב, 5/11/2018 בשעה 23:59

מטרת התרגיל:

בתרגיל זה נתרגל את המונחים הבסיסיים בקורס, ובכללם מחלקות (Classes), משתני מחלקה (Constructors) ופונקציות מחלקה (Member Variables and Member Functions) בנאים (Member) במאים (Access Modifiers), תוך התאמת למשك שנקבע מראש.

נתונים לכם קבצים עם מספר פונקציות בסיסיות. עליהם ליצור פרויקט ב-Visual Studio, להוסיף את הקבצים הנתונים, כפי שהואר בתרגול, ולהוסיף את הקבצים שלכם עם המחלקות שאתם נדרשים ליצור. את הקבצים הנתונים אין לשנות בשום אופן, הקבצים שיוצרים לתרגיל שתגלוו צריכים להיות זהים לקבצים המקוריים שקיבלתם, אלא אם צוין בפירוש אחרת.

תיאור כללי:

בתרגיל תמשו אובייקטים פשוטים של צורות גיאומטריות. אתם צריכים למש 4 אובייקטים: מלבן, משולש שווה שוקיים, בית ועפיפון. אתם מקבלים מأتנו תכנית שימושת באובייקטים שימושתם ומצירתם אותם על המספר.

מה אתם צריכים לעשות:

אתם צריכים לכתוב 4 מחלקות (classes):

1. Rectangle – מלבן מקביל לצירים.
2. IsoscelesTriangle – משולש שווה שוקיים עם בסיס מקביל לציר ה-*x*.
3. House – בית המורכב מלבן ווגם משולש המקבילים לציר ה-*x*.
4. Kite – עפיפון המורכב משני משולשים שווים שוקיים.

מחלקות צרכות להיקרא בדיק בשמות האלה וכל אחת צריכה להיות מוגדרת בקובץ "h". בשם תואם. בהמשך יוגדרו הבנאים (constructors) והפונקציות הציבוריות בבדיקה עבור כל מחלקה. הממשק מוגדר בצורה מדינית – אין להשמיט או להוסיף פונקציות ציבוריות. החתימות של הפונקציות הציבוריות (Public) צרכות להיות בדיק כפי שהן מופיעות בהגדרת התרגיל (למעט מקומות שבהם מופיעים בתרגיל מספרים מפורטים מטעמי נוחות, ואצלם בתרגיל מצופה שתשתמשו בקבועים מתאימים). אין לחסוף משתנים פנימיים.

בכל מובן אחר, אתם חופשים למשם את המחלקות כרצונכם (בכפוף, כמובן, לכללי התכונות הנאות).
לכן תוכלו (ותצטרכו) להחליט בין השאר:

- איזה מידע יש להחזיק בתוך האובייקטים?
- איך ליצג מידע זה?
- אלו פונקציות פרטיות (Private) יש למשם?
- איזה שימוש חוזר ניתן לעשות בקוד שכבר כתבנו ולהימנע מכפל קוד?

מה אתם מקבלים מאייתנו:

1. אתם מקבלים את הקובץ `h` שבו הגדירה של `struct` שנקרא `Vertex` המייצג קדקוד
במישור על פי קואורדינטות (`y, x`).

אם יש פונקציות ציבוריות שנראה לכם הגיוני לשימוש בקובץ זהה (למשל, אם חשבתם על השוואה בין שני אובייקטים מסווג `Vertex` שתשתמשו בה במספר מקומות), אפשר להוציא את ההצעה על הפונקציה לקובץ זהה, ואת השימוש לשימוש בקובץ `Vertex.cpp`, כמובן.

תוכן הקובץ הבסיסי הוא:

```
#pragma once
struct Vertex
{
    double m_x;
    double m_y;
};
```

2. קובץ שנקרא `macros.h` בו מוגדרים קבועים `X_MAX` ו-`Y_MAX`. אלה ערכי ה-`X` וה-`Y` המxisים שניתן לקבל. ערכי ה-`X` האפשריים בתרגיל הם המספרים בין 0 ל-`X_MAX` וערכי ה-`Y` האפשריים בתרגיל הם המספרים בין 0 ל-`Y_MAX`. אם מתכוונים לבנאים נתוניים החורגים מהתחומים הללו, צריך להעתלם מהנתון הבועתי ולהשתמש במקומו בערכי ברירת מחדל המוגדרים בהמשך לכל מחלוקת. ערכי ברירת המחדל הללו נמצאים, כמובן, בתחום החוק, ואין צורך לבדוק אותם שוב בקוד שלכם. **שים לב לא להתייחס בקוד שלכם למספרים הספציפיים המופיעים בקובץ אלא לקבועים הנ"ל.** נזכיר שוב, הקובץ `macros.h` שאתם מגישים צריך להיות זהה לזה שקיבלתם.

3. קובץ שנקרא `Shapes.cpp`, המכיל תכנית המקבלת נתונים מהמשתמש ועל פיהם יוצרת אובייקטים ממחלקות שאתם צריכים למשם, מנהלת לוח פנימי שעליו יצירוי הצורות ולבסוף מדפסה אותו בחלון המוסף (Terminal, Console). למרות שעלייכם להגיש את הקובץ זהה כדי שקייבתם, מומלץ לשנות אותו לעצמכם כדי להוסיף לעצמכם בדיקות נוספת אבל להגשה תצרכו את הקובץ המקורי. הקובץ הזה הוא לא בהכרח ה-`tester` שאתו התרגיל "בדק אבל מן הסתם הוא דומה לו". התוכנית נועדה להמחיש ולהדגים את צורת השימוש

אובייקטים שאתם צריכים למש. בתחילת התוכנית קיימים define-ים, RECTANGLE, HOUSE, ISOSCELESTRIANGLE, KITE ו-DIAGONAL הקובעים אילו אובייקטים לבדוק, כפי שהסביר בתרגול.

4. צמד קבועים בשמות h Board.cpp וה-Board.h המכיל את פונקציות הלוח הרלוונטיות. תCENTERכו לכלול את h.h (כלומר, להוסיף include אליו) במחלקות שלכם כדי שתוכלו למש את פונקציית drawLine, על ידי שימוש בפונקציה drawLine.

5. צמד קבועים בשמות Utilities.h Utilities.cpp ישמשו אותנו לפונקציות עזר כלליות למימוש פונקציות הלוח.

פירוט המשך:

כפי שהזכרנו לעיל, עליו למש 4 מחלקות. כתת נפרט את חתימות הפונקציות הציבוריות שיש לספק עבור כל מחלוקת.

עבור Rectangle

Rectangle (const Vertex& bottomLeft, const Vertex& topRight) – בניין המקביל שני קדקודים התוחמים את המלבן. כפי שהשמות מרמזים, הראשון מצין את הקדקוד השמאלי-תחתון והשני – את הקדקוד הימני-עליון. כפי שהוזכר לעיל, יש לוודא שערך ה-*X* של שני הקצוות נמצאים בתחום [MAX_X,0] וערך ה-*Y* נמצאים בתחום [0,MAX_Y]. אם אפילו אחד מהפרמטרים לא עונה על הקритריון הזה עליו לבנות Rectangle שקדקודיו כנ"ל הם (20,10) ו-(30,20). כמו כן יש לבדוק האם הקדקוד השמאלי-תחתון אכן שמאלי-תחתון. נציג כי יתכן שהקדקודים יתלכו באחת הקואורדינטות או יותר, אבל לא יתכן שהקדקדוד השמאלי יהיה מימין לימין או שהעלין מתחת לתחתון. גם עבור הבדיקה הזאת, אם היא נכשלה, ניצור את המלבן לפי ברירת המחדל כנ"ל.

Rectangle (const Vertex vertices[2]) – אותו דבר בדיקת כמו הבנייה הראשון, כולל הבדיקות וברירת המחדל במקרה שהן נכשלות, אלא שמקבלים את הקדקודים במערך ([0]vertices הוא הקדקוד השמאלי-תחתון ו-[1]vertices הוא הימני-עליון).

Rectangle (double x0, double y0, double x1, double y1) – בניית Rectangle שבו הקדקוד השמאלי-תחתון הוא הנקודה (x0,y0) והקדקדוד הימני-עליון הוא (y1,x1). כמובן שגם פה צריך לעשות את אותן בדיקות כמו בבנייה הקודמים וברירת המחדל כנ"ל.

Rectangle (const Vertex& start, double width, double height) – בניין המקביל קדקוד שמאלי-תחתון, רוחב וגובה ובונה על פיהם מלבן. גם כאן צריך לבדוק אם הקדקוד המועבר והקדקדוד שתחשבו מטור הנתונים נמצאים בטוויח שהוגדר למעלה ואם לא – לפעול באותה צורה כמו בבנייה הקודמים.

Vertex getBottomLeft() const – מחזירה את הקדקוד השמאלי-תחתון של המלבן.

– מוחזירה את הקודקוד הימני-עליון של המלבן.
double getTopRight() const
 – מוחזירה את הרוחב של המלבן (אורק הצלע המקבילה לציר ה-*x*).
double getWidth() const
 – מוחזירה את הגובה של המלבן (אורק הצלע המקבילה לציר ה-*y*).
double getHeight() const

עובר IsoscelesTriangle:

[3] **IsoscelesTriangle (const Vertex vertices[3])** – בניית משולש משלושה קודקודים. גם כאן, צריך לבדוק אם כל שלושת הקודקודים נמצאים בתחום החוק. בנוסף, צריך לוודא שאכן קיבלנו משולש כנדרש, כלומר שצלע הבסיס שלו מקבילה לציר ה-*X* ושתי הצלעות האחרות שוות זו לזו. אם לא – צריך ליצור משולש שלושת קודקודיו בנקודות (20,20), (25,25), (30,20). ניתן להניח שאנו מקבלים את הקודקודים משמאל לימין (על פי ערכי ה-*X* שלהם), כלומר הקודקוד השמאלי, המרוצחי והימני.

IsoscelesTriangle (const Vertex& center, double width, double height) – בניית משולש מקודקוד המרכז (הזווית המחברת בין שני השוקיים), רוחב צלע הבסיס, וגובה המשולש. בהינתן ערכים חיוביים לארוגמנט הגובה יפנה המשולש כלפי מעלה ▲, ואילו בהינתן ערכים שליליים יפנה המשולש כלפימטה ▼. גם כאן, צריך לבדוק אם כל שלושת הקודקודים נמצאים בתחום החוק. בנוסף, צריך לוודא שאכן קיבלנו משולש כנדרש, כלומר שיש לו צלע מקבילה לציר ה-*X* ושהוא שווה שוקיים. אם לא – צריך ליצור משולש ברירת מחדל כפי שראינו במבנה הקודם. שימו לב שגם הנחתם במבנה הקודם את הסדר משמאל לימין, עליו לוודא גם במבנה זהה שאתה שומרים על הסדר זהה (כלומר, הקודקוד שקיבלתם יהיה השני מבין השלושה).

Vertex getVertex (int index) const – פונקציה המוחזירה את הקודקוד *i*-index (כאשר האינדקס הוא בין 0 ל-2, כרגע במערכות) מבין שלושת קודקודיו המשולש. (במהשך הזמן נלמד איך למש זאת אופרטור [] כך שהיא אפשר לגשת אליהם כמו במערך רגיל: Triangle[0], Triangle[1], Triangle[2]). סדר הקודקודים הוא לפי שיורי ה-*X* שלהם (כלומר, בסדר שבו קיבלתם אותם, אם הנחתם לעיל תקינות...)

double getLength() const – מוחזירה את אורק צלע הבסיס של המשולש.
double getScalesLength() const – מוחזירה את אורק השוק של המשולש.
double getHeight() const – מוחזירה את הגובה של המשולש מהבסיס לקודקוד שבו מחוברים השוקיים השווים, כלומר הגובה לציר ה-*Z*.

עובר House:

House(const Rectangle& rectangle, const IsoscelesTriangle& triangle) – מקבלת מלבן ומשולש חוקיים. בנויסף עליהם לבדוק כי צלע הבסיס של המשולש (המאוזן ומקביל לציר ה-*X*) מציה

על אותו שיעורי Z של הבסיס העליון (המאוזן ומקביל לציר ה- X) של המלבן כך שתיווצר צורת בית כזו  . וכן שהקודקוד פניו כלפי מעלה.

יתכן שagg הבית רחוב יותר מהבית אך לא להיפר. (מיותר לציין כי עליים לוודא שagg הבית יפנה כלפי מעלה). כמו כן נניח כי אג הבית תמיד נמצא במרכז המלבן (אופקית). במקרה של אי חוקיות ניצרת בית שערci קודקודיו הם ((30,20) (20,10) (30,20) (25,25) (20,20))

House(const Vertex& roofTop, double width, double roofHeight, double baseHeight)
– מקבלת קודקוד של הנקודהגובה של האג, רוחב וגובה המלבן של הבית המבוקש, ותיצור בית אג שרוחב הבית יהיה זהה לרוחב האג והנקודה שתתקבל תהיה הנקודהגובה של האג. במקרה כישלון של החוקיות עליים ליצור את צורת ברירת המחדל.

extendRoof(double width) – תבדוק האם הרוחב החדש שיתקבל כARGUMENT הוא חוקי וכי האם הוא רחוב מהקיים. במקרה הצלחה היא תעדכן את שיעורי הקודקודים של האג לפני הרוחב החדש ותחזיר `true`. במקרה כישלון היא לא תעדכן ותחזיר `false`.

getHeight() const – ממחישה את גובה הבית מהרצפה עד לנקודהגובה בגג.
getWidthDifference() const – ממחישה את הפרש בין רוחב האג לרוחב הבית (0 במקרה שאין הפרש).

עובר Kite:

Kite(const IsoscelesTriangle triangles[2]) – בונה עפיפון משני משולשים שווים. עליים לבדוק את חוקיות המשולשים. בנוסף עליים לבדוק כי אכן האחד נמצא מתחת לשני בצורה שיתקבל דלתון (כלומר שני הקודקודים הגובה והנמוך נמצאים באותו שיעורי X , וכן שבסיסו הרוחב מתלכדים ושווים באורכם) במקרה של אי חוקיות עליים ליצור עפיפון ברירת מחדל מהנקודות הבאות:

(30,20) (25,25) (20,20)

(30,20) (25,15) (20,20)

Kite(const Vertex& top, double width, double topHeight, double bottomHeight) – בונה עפיפון מהקודקוד הגובה ביותר ביחס לנקודת הזיכרה למעלה. שימו לב כי סופקו הגבהים של שני המשולשים אך רוחבם זהה. עליים ליצור משולש תחתון כשהבסיס משותף לזה של העליון אלא שהתחתון פונה כלפי מטה לפני הגובה שסופק.

double getTotalHeight() const – מחזירה את הגובה הכולל של העפיפון, מהקודקוד הגבוה ביותר עד הנמוך ביותר.

שיטות שיהיו בכל אחת מהמחלקות:

void draw(Board& board) const – מצירת את הצורה על הלוח שהתקבל כפרמטר. כפי שהוזכר, אתם מצופים להשתמש בפונקציה drawLine() של מחלקת Board כדי לצייר את הצורה שלכם על הלוח.

הערה: בציור הבית והעפיפון אתם רשאים לצייר את קו הבסיס של המשולשים.

Rectangle getBoundingClientRect() const –מחזירה את המלבן המקביל לצירים (כפי שהגדכנו את Rectangle לעיל) החוסם את הצורה, ככלומר המלבן הקטן ביותר שמקביל לצירים ומכל את המלבן, המשולש, הבית או העפיפון.

double getArea() const – מחזירה את שטח הצורה הכולל.

double getPerimeter() const – מחזירה את היקף הצורה (במקרה של בית ועפיפון יש לחשב

ללא הבסיס המשותף וככלל את ההפרש ברוחב כלומר .

Vertex getCenter() const –מחזירה את מרכז הצורה (מרכז הקודקודים) עבור צורה פשוטה. במקרה של בית נגדיר את נקודת המרכז באמצעות צלע הבסיס של הגג (זהה גם אמצע תקרת מלבן הבסיס). ובמקרה של עפיפון נגדיר את נקודת המרכז במרכז האלכסון האופקי של הדלתון (אםצע הבסיס המשותף של המשולשים).

bool scale(double factor) – מגדילה או מקטינה את מרחק כל הקודקודים ממרכז הצורה לפי הפקטור המועבר וביחס למרכז הצורה (שהוגדרה למעלה) ומחזירה true. אם הצורה חורגת מהגבוקות שהוגדרו (אחד הקודקודים יהיה עם ערכים החורגים ממה שהוגדר לעיל) – להשאר את הצורה ללא שינוי ולהחזיר false. על מנת לפתע פעליה זו תוכל במקרה זה להוסיף העמסה של פונקציה זו כך שתקבל ארגומנט נוסף.

הערות חשובות:

- על מנת להשוות בין שני ערכי double אין להשתמש באופרטור ==. זאת מאחר ורבה פעמים שני ערכים מתקובלים מחישובים שונים והם שוויים בהתכנסותם למספר הרצוי, אף בפועל כל אחד מיוצג מעט אחרת בגליל אילוצי מקום בחוץ. (לדוגמה המספר 0.99999999

.... עלול להיות שווה למספר 1) لكن על מנת להשוות שני מספרים علينا להשתמש בחישוב

ההפרש ולקבוע אפסיון כלשהו כך שאם ההפרש הוא קטן ממנו, יוכרזו המספרים כשוויים.

קובץ ה-README:

יש לכלול קובץ README שיקרא README.txt או README.doc, README.docx (ולא בשם אחר). הקובץ יכול להיות בעברית ובלבד שיכיל את הסעיפים הנדרשים.

קובץ זה יכול לכל הפחות:

1. סורתה.
2. פרטי הסטודנט: שם מלא כפי שהוא מופיע ברישומות המכלה, ת"ז.
3. הסבר כללי של התרגיל.
4. רשימה של הקבצים שנוצרו ע"י הסטודנט, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.
5. מבני נתונים עיקריים ותפקידיהם.
6. אלגוריתמים הרואים לצוין.
7. באגים ידועים.
8. הערות אחרות.

יש לתמצת כלثنון אך לא יותר על אף חלק. אם אין מה להגיד בנושא מסוים יש להשאיר את הכותרת ומתחתי פסקה ריקה. כתבו ב-README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.

אוף הגשה:

הקובץ להגשה: יש לדוחו כל קובץ הקשור לתרגום, למעט מה שיצוין להלן, לקובץ ששמו ex_name.zip, כאשר N הוא מספר התרגום וname הוא השם המלא. במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית ex_name1_name2.zip, עם שמות המגישים בהתאם (לא רוחחים; גם במקרים עצם יש להחליף רווחים בקו תחתוי או להצמיד את שני חלקיו השם).

לפני דחיסת תיקיית החסונים Solution שלכם יש למחוק את הפריטים הבאים:

- תיוקיות בשם debug או release (לרוב יש יותר מתיקייה אחת בשם זה, אחת בתיקיית ה-Solution ואחת בתיקייה כל פרויקט).
 - תיוקייה בשם 64x, אם קיימת.
 - תיוקייה בשם 32. שאמורה להיות בתיקייה עם החסונים Solution.
- זכרו שגם את הקבצים שננתנו לכם צריכים לצרף, והם צריכים להיות ללא שינוי, כפי שקיבלתם אותם.

ככלל אצבע, אם קובץ ה-.kōz שוקל יותר ממ"ב אחד או שניים, כנראה שלא מוחקתם חלק מהקבצים הבינאריים המוזכרים.

את הקובץ יש להעלות ל-Moodle של הקורס למשימה המתאימה.

הגשה חוזרת: אם מסיבה כלשהי סטודנט מחליט להגיש הגשה חוזרת יש לוודא שם הקובץ זהה לחלווטין לשם הקובץ המקורי. אחרת, אין הבודק אחראי לבדוק את הקובץ האחרון שיוגש.

כל שינוי ממה שמוגדר פה לגבי צורת ההגשה ומבנה ה-README עלול לגרום הורדת נקודות בציון.

מספר הערות:

1. שימוש לב לשם הקובץ שאכן יכול את שמות המגיישים.
2. שימוש לב שעיליכם לשלוח את תיוקית ה-solutions כולה, לא רק את קובצי הקוד שיצרתם. עקובו אחרי ההסבר המפורט באתר, במקרה שאתם לא בטוחים איך למצוא את התיקייה. תרגיל שלא יכול את ה-solutions, לא יתקבל וידרשו הגשה חוזרת (עם כללי האיכון הרגילים).

המליצה כללית: אחרי שהכנתם את הקובץ להגשה, העתיקו אותו לתיקייה חדשה, חלצו את הקבצים שבתוכו ובדקו אם אתם מצליחים לפתח את ה-solutions שבתוכו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!