**תרגיל 1**

**נושא: מחלקות, מתודות, בנאים והרסנים**

**הוראות כלליות**

במערכת הבדיקות יש להיכנס לקורס OOP ולבחור ex1.

יש להגיש למערכת 6 קבצים: Grid.cpp, Rectangle.cpp, Point.cpp, Grid.h, Rectangle.h, Point.h

* **תאריך הגשה אחרון: 07/04/2020 – לא יתקבלו הגשות אחרי תאריך זה מכל סיבה שהיא!**
* יש לקרוא היטב את התרגיל לפני שמתחילים לעבוד ולוודא שהבנתם את כל הפרטים.
* יש לקרוא את קובץ "דרישות סגנון תכנות" היטב לפני שמתחילים לעבוד, ולמלא אחר הדרישות.
* ההגשה הינה ביחידים בלבד
* **שימו לב שהבדיקה הינה אוטומטית! תכנית שלא מתקמפלת או מתקמפלת עם אזהרות תקבל ציון 0. שימו לב – לא יתקבלו ערעורים על שגיאות/אזהרות קמפול. אנא ודאו כי אין בעיות מהסוג הזה בתרגילים אותם אתם מגישים!**
* יש להקפיד על הגדרת private ו-public עבור משתני המחלקה והמתודות. לא לשכוח להוסיף הערות בפורמט הנכון (שם ופרטים בהתחלה, הערה לפני כל מתודה, הערה לכל מחלקה, הערות בקוד עצמו). תרגיל שיוגש ללא הקפדה כל כתיבת ההערות לפי הפורמט עלול לקבל הורדת ציון של 15 נק'.
* במסגרת התרגיל תקבלו קבצי H עם המתודות שעליכם ליישם. עליכם ליישם את המתודות המתאימות בקבצי ה-CPP התואמים וכמו כן להגדיר את משתני המחלקה החסרים בקבצי ה-H שניתנו.
* **אין לשנות חתימה של מתודות בשום צורה!**
* **אין לבצע פעולות קלט/פלט בשום פנים ואופן!**
* **אין לשנות קוד קיים! ניתן להוסיף קוד לפונקציה כתובה אך אין למחוק את מה שכתוב בה!**
* לאחר כתיבת התכנית מומלץ לעבור שוב על התרגיל ולוודא שמילאתם את כל ההוראות – בכל הסעיפים ובכל ההערות.

**תיאור התרגיל**

בתרגיל זה תתבקשו להגדיר 3 מחלקות Grid, Rectangle, Point.

ההגדרות של המחלקות מופיעות בקבצי H הניתנים לכם בהורדת המטלה.

שימו לב כי קבצים אלה חסרים את משתני המחלקות ועליכם להשלים אותם בעצמכם.

שימו לב כי אין לשנות/להוסיף מתודות בשום אופן!

**המחלקה Point:**

מייצגת נקודה במישור. המחלקה מכילה משתנים עבור שני השיעורים של הנקודה. כמו כן המחלקה מכילה משתנים נוספים עבור שיעורי הנקודה שיעזרו לנו בקנה מידה של הנקודה במישור (scaling).

**הערה**: get של ערך ה-X למשל חייב להכיל שילוב עם קנה המידה ולא להחזיר רק את ערך ה-X.

המחלקה מכילה גם setterים ו-getterים עבור משתנים אלה.

בנאי המחלקה מקבל כפרמטר את שיעורי הנקודה ומאתחל אותה כנדרש. ערכי קנה המידה צריכים להיות מאותחלים ל-1.

**המחלקה Rectangle:**

מייצגת מלבן צבעוני במישור. המחלקה מכילה 2 משתנים מסוג point המייצגים את הפינה השמאלית העליונה והימנית התחתונה של המלבן.

שמאלית עליונה – טופ LEFT

ימנית תחתונה – width \

טופ ולפט זה הערכים של הנקודה השמאלית העליונה,

את הערכים של התחתונה הימנית צריך לחשב בעזרת הטופ ולפט ועוד האורך והרוחב

בנוסף, המחלקה מכילה משתנה השומר את הצבע של המלבן. לצורך התרגיל הצבע ייוצג באמצעות מספר שלם. המחלקה מכילה setter ו-getter עבור משתנה זה.

כמו כן המחלקה מממשת מתודה contains שבודקת האם נקודה שהועברה לה כפרמטר נמצאת בתוך המלבן.

* כלומר אם אקבל X Y אצרך לבדוק שהגבולות שלהם הוא : של האיקס ה

מתודה נוספת היא moveRect המקבלת ערכי delta להזזה. הפונקציה תזיז את המלבן מהיכן שהיה למקום החדש לפי הערכים שקיבלה.

מתודה נוספת היא scaleRect שמקבלת ערכי גובה ורוחב חדשים ומתקנת את קנה המידה של המלבן בהתאם. הערה: הנקודה השמאלית העליונה גם אמורה לעבור שינוי קנה מידה.

בנאי המחלקה מקבל כפרמטר את השיעורים של הפינה השמאלית העליונה, את הגובה, הרוחב וצבעו של המלבן.

שימו לב! במחלקה זו נוסף משתנה סטטי לשם הבדיקות. אין לשנות אותו או את ערכו!

**המחלקה Grid:**

מייצגת רשת של מלבנים. המחלקה מכילה 2 משתנים. משתנה השומר את כמות המלבנים ברשת, ומשתנה אשר מהווה מערך של מצביעים למלבנים מהם מורכבת הרשת.

הנקודה השמאלית העליונה של הגריד הינה 0,0 (כמו בציור למטה).

המחלקה מכילה מתודת getRectAtPoint() שמקבלת כפרמטר נקודה במישור ומחזירה מצביע למלבן המכיל אותה מתוך מלבני הרשת. במימוש המתודה ניתן להניח (אין צורך לבדוק) שקיים בדיוק מלבן אחד ברשת שמכיל את הנקודה הנתונה.

המחלקה מכילה מתודת getRectAtIndex() המקבלת אינדקס של מלבן ומחזירה את מצביע למלבן שנמצא באינדקס הזה (שימו לב כי האינדקסים מתחילים מ-0).

כמו כן ישנה מתודה moveGrid() המקבלת ערכי delta להזזה. הפונקציה תזיז את כל הגריד מהיכן שהיה למקום החדש לפי הערכים שקיבלה (כמובן שיש צורך להזיז את כל המלבנים).

מתודה נוספת היא scaleGrid() המקבלת ערכי גובה ורוחב חדשים של מלבן ומתקנת את קנה המידה של כל המלבנים בהתאם.

בנאי המחלקה מקבל כפרמטר את הרוחב והגובה של כל אחד ממלבני הרשת, את כמות המלבנים בכל שורה ובכל עמודה ואת צבעם ההתחלתי של כל המלבנים ומאתחל את מערך המלבנים כך שהמלבנים ייווצרו לפי העמודות משמאל לימין. להלן דוגמא לסידור של 12 מלבנים, המספרים מייצגים את סדר המלבנים במערך:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 3 | 6 | 9 |
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |

**רמז לפתרון:**

בהינתן מחלקה A בעלת בנאי שמוגדר כך: A(int x) ניתן להקצות מערך של אובייקטים מסוג A באופן הבא:

A\*\* arr = new A\*[5];

arr[0] = new A(42);

arr[1] = new A(17);

...

**רמז נוסף:**

בתרגיל זה אנו משתמשים במערכת קורדינטות בה ציר ה-Y מופנה כלפי מטה (ראו איור). כלומר, ערכי Y גדלים ככל שמתקדמים כלפי מטה.



טוב משהו שהבנתי אחרי שהסבירו לי ואחרי שקיללתי את אביב:

בתכנות מערכת צירים עובדת הפוך

הסיבה שציר y גדל ככל שהוא יורד למטה במערכת צירים זה בגלל שאם משתמשים בלולאה

(מטריצה לדוגמא) הi שמייצג את השורות יורד למטה (פשוט אי אפשר לעלות למעלה)

מייל מחיים המתרגל

כמו כן, מטלה ראשונה הועלתה כפי שידוע לכם. הבהרה בנוגע לפרמטר ה-scaling (קנה מידה):

מדובר על ערך double שמרחיב/מכווץ את הגרף. למשל: אם הנקודה היא (1,2) והקנה מידה הינו 3 הרי ערכי ה-X וה-Y מוכפלים פי 3, כלומר הנקודה ה"מתוחה" הינה: (3,6).

כשבניתם את המלבן, נתנו לכם גובה ורוחב ולפיהם חישבתם את הקורדינטות של הפינות. מה שהפונקציה עושה, זה לקבל גובה ורוחב חדשים של המלבן.

הקושי בניסוח: יש אינסוף אפשרויות להשיג את השינוי הזה. כדי לקצר את האורך של המלבן ב2 יחידות, אפשר לקחת יחידה 1 מהX של כל אחת מהפינות. אבל אפשר באותה מידה לקחת 2 מהX של אחת מהפינות. למעשה יש אינסוף קומבינציות אפשריות של שינויים שאפשר לעשות למלבן, העיקר שההקטנה של הX של שתי הנקודות תשלים ל2 (לדוגמה, 1.5 ו0.5).

מה שהם באמת התכוונו, זה שתחשבו מחדש את האורך והחורב של המלבן (ע"י חיסור של X וY של הנקודות), תחשבו יחס בין האורך והרוחב החדשים לבין הישנים, ותשנו את הסקאלה של הנקודות בהתאם ליחס שיצא.

לדוגמה כדי להגדיל פי 2 מלבן שאורכו ורוחבו 1, צריך לחלק 2 ב1, ואת היחס (נסמן בr) לשים בsetScale של כל אחת מהנקודות