הנדסת תוכנה 094129

תרגיל בית 2

תכנות מודולרי, שימוש במחלקות וניהול זיכרון דינאמי

תאריך אחרון להגשה : 27-05-2018 עד השעה 23:55

נושא התרגיל: מימוש ממשק iRobot מתקדם

<u>תיאור המערכת – דומה למערכת בתרגיל בית 1, כולל השינויים שיפורטו בהמשך.</u>

iRobot החליטו להוסיף מספר אפשרויות נוספות למערכת המתקדמת שלהם. בכדי להגדיר לרובוט מקומות אשר הוא לא יוכל להגיע אליהם הוחלט לממש קירות וירטואלים (virtual wall).

<u>מפה:</u>

כפי שצוין לעיל, לרובוט חיישנים רבים אשר עוזרים לו למפות את הסביבה בצורה חכמה ולזהות היכן בחדר מסוים יש לכלוך והיכן לא, ובכך לייעל את הניווט שלו בסביבה. המיפוי מיוצג באמצעות מפה דו מימדית, כאשר תא עם הערך '0' משמעו נקי, תא עם הערך '1' משמעו קיר (אין מעבר) ותא עם הערך '2' משמעו תא מלוכלך.
את יכולות התנועה של boomba נמדל בעזרת שמונה כיוונים: למעלה (U), למטה (D), שמאלה (L), ימינה (R); ולארבעת האלכסונים: למעלה ימינה (UR), למעלה שמאלה (UL), למטה ימינה (DR), למטה שמאלה (DL).

לדוגמא נתונה מפה 7X7 הבאה:

	0	*1*	*2*	*3*	*4*	*5*	*6*
0	0	1	1	1	1	1	1
1	1	2	2	0	1	0	1
2	0 	0	1	1	1	2	1
3	1	0	0	0	0	0	1
4	1	1	1	2	1	0	1
5	1	0	0	0	0	2	1
6	1	0	1	1	1	1	1

נניח כפי שתואר לעיל רובוט יחיד בשם "boom" נמצא בתא (2,0). אם יקבל פקודה ימינה (R) ינוע לתא (2,1). אם בהימצאו בתא (2,1) יקבל את הפקודה למעלה ימינה (UR), "boom" ינוע לתא (1,2).
כעת אם בהימצאו בתא (1,2) יקבל את הפקודה למטה (D) הוא לא יזוז כיוון שיש קיר וירטואלי.
בנוסף, במידה ו- "boom" יקבל את הפקודה "clean" הוא ינקה את התא בו הנמצא. כלומר ערך התא יהפוך ל-0 (לא

משנה ערכו הקודם- '0' או '2').
התוכנית תתחיל עם מפה כפי שהוגדרה בדוגמא לעיל.
אפשר להגדיל את המפה בזמן ריצה. כלומר ברגע שמנסים להוסיף קיר או מעבר במקום חיובי, לא מוגדר (פקודות 3 או 4 המוגדרות בהמשך, קואורדינטות חיובית בלבד), המפה תתרחב לגודל הרצוי, כאשר כל התאים שלא הוגדרו יאותחלו ל"קיר" '1'.

מבנה הנתונים לתכנה:

במימוש המחלקות עליכם להשתמש בטיפוסי הנתונים הבאים (שימו לב שעליכם למקם את המחלקות בקבצי header השונים בהתאם למבנה הקוד):

```
// A vector of robots
typedef std::vector<Robot*> RobotVec;
typedef std::vector<Robot*>::iterator RobotVec_it;
typedef std::vector<Robot*>::const_iterator RobotVec_cit;

// Map types:
typedef int size_type;
typedef int** grid_type;

// Robot connection status
typedef enum { COMMUNICABLE = 0, NON_COMMUNICABLE = 1} connection_e;
```

כחלק מהפתרון עליכם לממש את המחלקות הבאות, בהתאם לממשק המתואר להלן:

Class **name**	Class members (**private**)	**Public** member methods
Coordinate	int x; int y;	Coordinate(int new_x, int new_y); Coordinate(const Coordinate& new_coordinate); ~Coordinate(); void print() ...
Robot	Coordinate coordinate; const std::string name;	Robot(const Coordinate& new_coordinate, const std::string& new_name); ~Robot(); void print() ...
RobotDB	RobotVec robots; Map *map;	RobotDB(Map *new_map); ~RobotDB(); void DeleteRobot(const std::string& rName);
Map	grid_type ppGrid; size_type size_h; size_type size_w;	Map(); ~Map();
Interface	RobotDB* pRobots;	Interface(RobotDB* pNew_Robots); ~Interface(){}; void StartControl(Map* map);

- עליכם לממש **לפחות** את המתודות שצוינו לעיל.
- כל שדות המחלקות פרטיות (all class members are private) גם הקיימים וגם החדשים, אם יהיו כאלה.
- למחלקה Robot שדה נוסף בשם dust_bin מסוג int אשר מתאר את מצב מיכל האבק.
- אין לשכפל מידע קיים ולשמור אותו תחת שם אחר.
- אין לשנות את מימוש המתודות print, printLoc, printClean של המחלקות Coordinate ושל Robot (מימוש של המתודות נמצא בקבצים המצורפים לתרגיל הבית).

הקלט לתכנה:

התכנה קולטת את נתוני הפקודות מערוץ הקלט הסטנדרטי (cin). הקלט מחולק לפקודות, וכל פקודת תופסת שורה אחת בדיוק בקלט. סדר ביצוע הפקודות חייב להיות סדר הופעתן בקובץ. המילה הראשונה בפקודה הינה תמיד השם שלה, ולאחר מכן פרמטרים לפקודה, מופרדים ברווחים. הפקודות האפשריות הינם (יש חשיבות ל-uppercase):

מס'	תיאור פקודה	מבנה שורת הפקודה	פרמטרים
1	הזז רובוט	Move NNN W	NNN- שם הרובוט בעל שלושה תווים בדיוק. W- כיוון התנועה מתוך הכיוונים: L, R, U, D, UR, UL, DR, DL
2	הזזה מרובה	MoveMulti NNN W1, W2 …end	NNN- שם הרובוט. W1, W2 … - כל כיוון כמו פקודה מס' 1. מס' הכיוונים אינו ידוע מראש. **שימו לב** שסוף הפקודה מסומן ע"י המחרוזת הקבועה end
3	הוסף קיר	AddWall x y	הוסף קיר למפה בשורה ה-x ובעמודה ה-y. (x,y מספרים שלמים)
4	הוסף מעבר	AddPath x y	הוסף מעבר נקי למפה בשורה ה-x ובעמודה ה-y. (x,y מספרים שלמים)
5	הצג רובוט חדש	Place NNN X Y	NNN- שם הרובוט. צור רובוט חדש בשם NNN והצב אותו במיקום ה-(x,y) במפה.
6	מחק רובוט	Delete NNN	NNN- שם הרובוט. מחק רובוט NNN מהמפה.
7	הזזה ובנייה	MoveBuild NNN W	NNN- שם הרובוט. W- כיוון התנועה מתוך הכיוונים: L, R, U, D, UR, UL, DR, DL. אם בכיוון הפקודה ישנו קיר, הרובוט יהרוס את הקיר (אחד) ויזוז תא אחד.
8	לכלך	AddDirt X Y	לכלך את המפה בשורה ה-x ובעמודה ה-y (x,y מספרים שלמים אי שליליים)
9	נקה	Clean NNN	NNN- שם הרובוט בעל שלושה תווים בדיוק. נקה את המפה במיקום הנוכחי של הרובוט.

התנהגות הרובוטים:

- הוספה של קיר/מעבר במקום לא מוגדר ושלילי (מספיק x או y שלילי) אינם משנים את מצב התכנה.
- הוספה של קיר/מעבר במקום לא מוגדר אי שלילי (x ו-y של קואורדינטה חיוביים או 0) יגדילו את המפה באופן מינימאלי כך שיכילו את הקואורדינטה החדשה, כך שכל תא שלא הוגדר יאותחל לקיר (1). לדוגמא לאחר הפקודה AddPath 0 7 על המפה מהדוגמא למעלה תיראה כך:

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	1	1	**0**
1	1	2	2	0	1	0	1	**1**
2	0	0	1	1	1	2	1	**1**
3	1	0	0	0	0	0	1	**1**
4	1	1	1	2	1	0	1	**1**
5	1	0	0	0	0	2	1	**1**
6	1	0	1	1	1	1	1	**1**

- לכלוך תא או הזזה/הוספה של רובוט במקום לא מוגדר (או במקום בו יש קיר) אינם משנים את מצב התכנה. מחיקת רובוט לא קיים אינה משנה את מצב התכנה.
- בעקבות תקלה בחיישני הרובוט, במידה ורובוט יצא מגבולות המפה הקשר עם הרובוט מופסק לצמיתות. לכן, מאותו הרגע והלאה מיקום הרובוט יהיה בנקודה (-1,-1) והרובוט יפסיק להגיב לפקודות מס' 1, 2, 7 ו-9 (הזזות וניקיון).
- ניתן להניח כי הכיוונים יהיו אך ורק אחד מהשמונה הנתונים.
- אם הרובוט(הנמצא במפה) קיבל הוראה לזוז לתא בו נמצא קיר, מיקום הרובוט לא ישתנה ותודפס הודעה על מיקומו.
- ניסיון ליצור רובוט חדש בתא לא חוקי (מחוץ למפה) יסתיים ללא יצירת הרובוט, ולכן גם לא תהיה שום הדפסה.
- ניסיון ללכלך או להוסיף קיר בתא בו ישנו רובוט (או רובוטים) יסתיים ללא הצלחה. ניסיון ללכלכך תא ובו יש קיר גם כן יסתיים ללא הצלחה.
- ניסיון חוקי (במקום חוקי - לפי המוזכר לעיל) ליצור רובוט קיים, יזיז את הרובוט למקום החדש, גם אם הרובוט היה מחוץ לגבולות המפה.
- מותר ליצור יותר מרובוט אחד בכל תא ומס' הרובוטים שיבנו אינו ידוע מראש.
- לכל רובוט יש מיכל אבק. מצב מיכל האבק מוגדר להיות בסולם מ-0 עד 5. כאשר 0 משמעו שהמיכל ריק (נקי) ו-5 המיכל מלא (מלוכלך) והרובוט לא יכול יותר לנקות. כל רובוט חדש נוצר עם מיכל ריק.
- בעקבות תלונות בקרב המשתמשים על ניקיון סרק, הוכנס שדרוג למערכת וכעת **כאשר אך ורק כאשר הרובוט מנקה תא מלוכלך מיכל האבק יעלה באחד ותודפס הודעת ניקיון מתאימה.**
- כאשר המיכל מתמלא, בהינתן פקודת ניקיון הרובוט לא ינסה לנקות אלא יגיע אל לנקודת הניקיון (0,0) וירוקן את המיכל(וזה יהיה אף מיקומו החדש).
- שלב הקלט יסתיים כאשר תתקבל הפקודה ctrl+z או סוף הקובץ, במקרה והקלט הופנה מקובץ.
- אין ליצור תפריט בחירה למשתמש- יש להקיש ישירות את הפקודות.
- אין להוסיף הדפסות משלכם ויש להשתמש רק בפעולות ההדפסה המפורטות בהמשך.

הנחות שמותר לכם להניח לגבי הקלט:

- פרט לשגיאות שתוארו המחייבות טיפול, אפשר להניח שיתר פקודות הקלט מכילות רק פקודות בפורמט הנ"ל.
- ניתן להניח כי הקלט מכיל רק פקודות בפורמט הנ"ל.
- ניתן להניח שלא מנסים להסיר רובוט שלא קיים בבסיס הנתונים.
- ניתן להניח שכל המחרוזות המוזכרות לעיל מכילות רק אותיות באנגלית (גדולות או קטנות) ומספרים. ובפרט, שאינן מכילות רווחים.
- ניתן להניח כי לא ייווצר קיר בנקודה 0,0 שכן זוהי נקודת ריקון המיכל של הרובוט.
- בסיום כל פקודה שהצליחה (הזזה/הוספה) יש לקרוא לפקודת הדפסה השייכת לאובייקטים מסוג Robot:
  PrintLoc();
  פקודת הוספה של רובוט (גם רובוט הנמצא ב 1-,1-) שאחריה הרובוט נמצא על המפה, תמיד מוגדרת כהצלחה.
- בסיום כל פקודת ניקיון מוצלחת יש לקרוא לפקודת ההדפסה: PrintClean()

הפלט של המערכת:

ההדפסות של המערכת יתבצעו בעזרת קריאה למתודה Print() המוגדרות בקובץ Robot.h, המצורף לתרגיל זה.
בסופו של דבר, בהינתן המפה למעלה, על המערכת עבור הקלט:

```
Place rb 2 0
Move rb R
Move rb D
Move rb R
Move rb R
Move rb R
Move rb R
Move rb U
Clean rb
Move rb UR
MoveBuild rb UR
```

וגם עבור הקלט:

```
Place rb 2 0
MoveMulti rb R D R R R R end
Move rb U
Clean rb
Move rb UR
MoveBuild rb UR
```

התכנית תדפיס:

```
Robot: rb at: 2,0
Robot: rb at: 2,1
Robot: rb at: 3,1
Robot: rb at: 3,2
Robot: rb at: 3,3
Robot: rb at: 3,4
Robot: rb at: 3,5
Robot: rb at: 2,5
Robot: rb is cleaning at: 2,5
dust bin: 1
Robot: rb at: 2,5
Robot: rb at: 1,6
```

(אין רווחים לאחר התו האחרון)

דרישות המימוש:

1. **המימוש חייב להכיל לפחות את הקבצים הבאים:**
   א. main.cpp - כפי שניתן על ידי צוות הקורס בקובץ המצורף.
   אין לשנות קובץ זה ,והמימוש שלכם חייב להשתמש בפונקציית main המוגדרת בו!
   ב. Interface.h – ממשק השליטה ברובוטים. מותר להוסיף מתודות ו/או תכונות במקרה הצורך, **אין למחוק תכונות/מתודות קיימות!**
   ג. Coordinate.h – ייצוג קואורדינטה במישור. מותר להוסיף מתודות ו/או תכונות במקרה הצורך, **אין למחוק תכונות/מתודות קיימות!**
   ד. Robot.h – מחלקה המייצגת רובוט בודד. כל ההדפסות יבוצעו אך ורק בעזרת המתודה print של מחלקה זו. מותר להוסיף מתודות ו/או תכונות במקרה הצורך, **אין למחוק תכונות/מתודות קיימות!**
   ה. RobotDB.h – מכיל ממשק של מודול שמממש את **אוסף הרובוטים**, ז"א שומר את הנתונים במבנה נתונים פנימי וממשמש פעולות עליו.
   ו. Map.h - מכיל ממשק של מודול שמממש את ממשק **המפה**, ז"א שומר את נתוני ופעולות המפה.
   יש לאתחל את המפה לפי הדוגמא בדף הראשון.
   ז. קבצי מימוש של המודולים הנ"ל (קבצי ה-cpp).
   ח. קבצי קוד אחרים, במידה ותמצאו זאת לנכון.
2. חובה לתעד את הקוד באנגלית (אין צורך לדאוג לרמת האנגלית) והתיעוד חייב להופיע בתוך הקוד, לפני הפונקציה\טיפוס\משתנה שהתיעוד מתייחס אליהם. נא תעדו ע"פ ההסברים באתר הבא:
   http://www.edparrish.net/common/cppdoc.html
3. **התיעוד יכיל:**
   א. בראש כל מחלקה תיאור קצר (כמה שורות) של ייעוד המחלקה והקשר שלו למחלקות האחרות.
   ב. לפני כל פונקציה תיאור בן שתי שורות של ייעוד הפונקציה, וכן תיאור של כ"א מהפרמטרים שלה (שורה אחת לכל פרמטר) וערך ההחזר שלה. תיעוד הפרמטרים\ערך החזר צריך להכיל הסבר לגבי תוכן המשתנה ,לא הטיפוס שלו.
   ג. לפני כל משתנה גלובלי תיאור בשורה של הייעוד שלו .
4. **יש לתת שמות בעלי משמעות לכל משתנה!** למען הסר ספק:
   א. השמות i,j,k,l,m עבור משתני לולאה שהם אינדקסים במערך הם כן בעלי משמעות.
   ב. השם it עבור iterator והשם cit עבור const_iterator הם כן בעלי משמעות ,לצורך מילוי הוראה זו.
   ג. פרט למצוין לעיל ,שם שהוא ראשי תיבות <u>לא</u> נחשב לבעל משמעות.
5. אין בתרגיל זה דרישה של יעילות זמן. בפרט, מבנה הנתונים vector בפירוש אינו היעיל ביותר מבחינה זו. עם זאת, אנו שומרים לעצמנו את הזכות להוריד נקודות על מימוש שמבצע פעולות מיותרות בעליל (פעולות שגם אם לא יבוצעו הקוד נשאר תקין).
6. עבודה עם (בפרט מעבר על) מיכלים סדרתיים תיעשה אך ורק בעזרת איטרטור המתאים ביותר.
7. הקדישו תשומת לב ומחשבה לאופן בו הפונקציות שלכם מקבלות פרמטרים .השתמשו בהעברה ע"י references במקום העברה by value במקרים בהם הדבר אפשרי וחוסך העתקות. חובה להשתמש ב - const לפי הנלמד בקורס!
8. **יש בתרגיל זה דרישה של יעילות זיכרון חופשי (heap).** עליכם ליצור (בעזרת new) אך ורק את האובייקטים הנדרשים באותו רגע, ולשחרר זיכרון (בעזרת delete או []delete) ברגע שאין בו יותר שימוש. **בפרט, כשהתוכנית מסתיימת, כל הזיכרון שהקצית בעזרת new חייב להיות משוחרר!** בפרט, אין ליצור העתקים של המפה.
9. **למען בהירות:** קבצי התרגיל, כפי שנתנו לכם, לא אמורים להתקמפל עד אשר תוסיפו את המחלקות המתבקשות.

<u>הוראות הגשה:</u>

1. התרגיל להגשה בזוגות **בלבד**.
2. הקוד חייב להיכתב על פי מוסכמות כתיבת הקוד (Coding Conventions) בקורס. קוד שלא עומד במוסכמות לא יזכה במלוא הניקוד.
3. **חובה** לקמפל את התרגיל בעזרת סביבת eclipse. תוכנית אשר תצליח להתקמפל בסביבה אחרת ולא בסביבה זו **<u>תחשב כקוד שלא עובר קומפילציה</u>**.
4. ההגשה חייבת להכיל **קובץ ZIP יחיד בלבד** (ולא קובץ RAR וכדומה) המכיל:
   - תיקיה בשם code ובה כל קבצי **קוד המקור (h/cpp)**, ללא קבצי הרצה.
5. שם הקובץ חייב להיות hw2_xxxxxxxxxx_yyyyyyyyyy.zip, כאשר xxxxxxxxxx ו-yyyyyyyyyy הם מספרים תעודות הזהות של המגישים, כולל ספרת ביקורת
6. ההגשה היא אלקטרונית **בלבד**, דרך אתר ה-moodle של הקורס. תרגילים שיוגשו בכל דרך אחרת **לא ייבדקו**.
7. אין להגיש את אותו הקובץ פעמיים. התרגיל יוגש על ידי <u>אחד</u> מבני הזוג.
8. שימו לב שההגשה תיחסם בדיוק בשעה 23:55. מומלץ להגיש לפחות שעה לפני המועד האחרון.
9. תרגיל בית שלא יוגש על פי הוראות ההגשה - **לא ייבדק**.

בהצלחה!