# נושאים מתקדמים בתכנות

## תרגיל 2

בתרגיל זה עליכם לבנות אלגוריתם אוטומטי עבור שחקנים ויריץ משחק אוטומטי באמצעות:

1. קובץ עבור לוח הקרב של שני השחקנים שיכלול את פיזור הספינות על הלוח של שני השחקנים.
2. שני קבצי dll, אחד עבור כל אחד מהשחקנים, שיכילו אלגוריתם משחק.

**אלגוריתם משחק:**

עליכם לממש 3 מחלקות עבור אלגוריתם משחק. כל אחד מהאלגוריתמים שתממשו יידרש לרשת מהמחלקה האבסטרקטית IBattleshipGameAlgo (שמוצגת באופן מלא ומדוייק בקובץ במקביל לקובץ התרגיל). אלגוריתם המשחק שתכתבו יידרש לממש את המתודות האבסטרקטיות הבאות:

```
void setBoard(int player, const char** board, int rows, int cols);

bool init(const std::string& path); // returns whether the init succeeded or failed

std::pair<int, int> attack(); // ask player for his move

void notifyOnAttackResult(int player, int row, int col, AttackResult result); // notify on last move result
```

שימו לב שהוספנו לפונ' setBoard פרמטר נוסף: מספר השחקן כפי שמנהל המשחק נתן לו.
כמו כן, התווספה הפונ' init שמחזירה האם היא הצליחה או לא.

הקריאה ל-init על אלגוריתם תתבצע לאחר הקריאה ל-setBoard. כמו כן, טעינת ה-dll-ים תתבצע סדרתית – יש לטעון את האלגוריתם הבא רק לאחר שהאלגוריתם הקודם נטען ואותחל בהצלחה.

המחלקה האבסטרקטית עודכנה ועליכם לעדכן את המחלקות הנורשות בהתאם!

האלגוריתמים שעליכם לממש הינם:

1. אלגוריתם נאיבי
   א. האלגוריתם יתקוף באופן סדרתי החל מהמשבצת השמאלית ביותר בשורה העליונה, ימשיך בתקיפות לאורך השורה ימינה עד לסיום השורה, בסיום השורה יעבור למשבצת השמאלית ביותר בשורה שמתחתיה וימשיך בה משמאל לימין וכך הלאה לשורות הבאות.
   ב. למרות שהוא אלגוריתם נאיבי, הוא לא טיפש. האלגוריתם לא יתקוף את הספינות של עצמו ולא יתקוף במשבצות שע"פ כללי המשחק לא יכולות להימצא בהן ספינות יריב.
2. אלגוריתם צעדים ידועים מראש מקובץ
   א. זהו האלגוריתם שמימשתם בתרגיל הראשון, ללא שינוי
   שמות קבצי המהלכים יהיו attack.* כאשר הקובץ שייבחר יהיה לפי הכלל הבא: אם מספר השחקן שקיבלתי מהמשחק הוא 0 – הקובץ הראשון במיון לקסיקוגרפי, אחרת – הקובץ השני במיון לקסיקוגרפי. במידה וישנם יותר משני קבצים, יש להתעלם מהקובץ השלישי ואילך במיון הלקסיקוגרפי. במידה ויש קובץ אחד ושני שחקנים - שניהם ישחקו לפי אותו קובץ. במידה ואין קובץ מתאים בספריה ה- init ייכשל והמשחק יצא בהודעת שגיאה שמפורטת בהמשך.
3. אלגוריתם חכם
   א. זהו האלגוריתם המנצח שלכם.
   ב. החלטות המימוש לגביו הן במגרש שלכם אולם אם המימוש שלו יהיה לא יעיל או מטופש הוא יפסיד נקודות גם אם הוא אלגוריתם מדהים (לא יעיל או מטופש – בהשוואה למימוש שמשיג את אותה מטרה באופן יותר יעיל / אלגנטי).

**<u>תיאור קבצי הקלט לתוכנית:</u>**

1. <u>לוח הקרב</u>:
   א. מבנה לוח הקרב, חוקיותו ומדיניות דיווח השגיאות נשאר זהה לתרגיל 1.

2. <u>קבצי מהלכי תקיפה</u>:
   א. רלבנטי עבור אלגוריתם צעדים ידועים מראש מקובץ
   ב. מבנה הקובץ וצורת השימוש בו זהים לתרגיל הראשון

3. <u>2 קבצי אלגוריתמים (dll)</u>:
   א. אתם אמנם מגישים 3 קבצי אלגוריתמים, אך קלט המשחק הוא 2 בלבד.
   ב. המשחק צריך לדעת לתמוך **בכל** dll המממש את הממשק שהוגדר, אפילו אם ה-dll לא נכתב על-ידכם, כל עוד הוא עומד בכללים שמוגדרים בתרגיל.

**בבדיקה נריץ קבצים שלנו על התכנית שלכם ולכן חשוב שהתכנית תדע להתמודד עם קבצים שונים בפורמט שהוגדר ולהוציא פלט מדוייק כפי שמוגדר.**

**<u>הרצת התכנית:</u>**

יורץ ה-exe של התכנית והוא יחפש 2 קבצי dll ב-path שבו הוא מחפש את הקובץ sboard. לפי הכללים של תרגיל 1. חובה למצוא 2 קבצי dll – <u>לא פחות</u>, אחרת יש להוציא הודעת שגיאה ולצאת (אבל לא עם exit).

קבצי ה-dll יורצו לפי מיון לקסיקוגרפי של שמם: שחקן A יקבל את הנמוך יותר במיון הלקסיקוגרפי ושחקן B את הגבוה. במידה וישנם יותר מ-2 קבצי dll התכנית תתעלם מאלו שאחרי השניים הראשונים במיון.

ה-exe יקבל ב-command-line ארגומנט ראשון עבור ה-path בו יש לחפש את הקבצים שתוארו. במידה ולא נמסר פרמטר עבור ה-path התכנית תחפש את הקבצים ב-working directory. במידה וחסרים קבצים, יש להוציא הודעת שגיאה ולסיים את ריצת התכנית:

Wrong path: <path>
Missing board file (*.sboard) looking in path: <path>
Missing an algorithm (dll) file looking in path: <path>
Cannot load dll: <file name including full path>
Algorithm initialization failed for dll: <file name including full path>

א. מדיניות קבלת הארגומנט והצגת השגיאות זהה לזו של התרגיל הראשון.
ב. הודעות השגיאה על תקינות קובץ הלוח זהות למה שהוגדר בתרגיל הראשון.

**הסבר לגבי יצירת dll, טעינת ה-dll ושימוש בו כולל הדגמה תקבלו בתירגול.**

**קבצי ה-dll:**

א. קובץ ה-dll יכיל את מחלקת האלגוריתם שלכם וכן פונקציה גלובלית בשם GetAlgorithm שתחזיר הקצאה דינמית של האלגוריתם שמומש בקובץ זה (ראו IBattleshipGameAlgo.h). הסברים נוספים לגבי ה-dll תקבלו בתירגול.

ב. לצרכי הפיתוח:
1. בשלב ראשון ניתן לממש את האלגוריתמים לא כ-dll אלא כמחלקות בפרוייקט הראשי.
2. בשלב שני מומלץ ליצור פרוייקט אחד עבור התוכנית המריצה (Console Application) ועוד 3 פרוייקטים מסוג (dll). כמו כן, מומלץ ליצור את כולם תחת אותו ה-Solution.
3. כדאי כנראה ליצור פרוייקט נוסף בשם Common שיכיל מימושים משותפים לפרוייקטים האחרים.

ג. לצרכי הגשה:
פירוט הקבצים יימסר בקובץ CMakeLists.txt אשר מגדיר את תהליך הבנייה והקימפול של ארבעת התוצרים שאותם נריץ בבדיקה. כל הקבצים יוגשו בתיקייה אחת בלי קשר לתוצר שאליו הם שייכים. קבצים שמעורבים ביותר מפרוייקט אחד (למשל גם ב-exe וגם בכל ה-dll-ים וכד') – יש לדאוג שיופיעו בפירוט של קובץ ה-CMakeLists עבור כל תוצר שצריך אותם אבל יש לצרף את הקובץ רק פעם אחת.

**תצוגת המשחק למסך (חובה בתרגיל זה!)**

יש לתמוך בהדפסת ריצת המשחק במסך קונסול:

- יש למצוא מימוש של הפונ' gotoxy ברשת, כנ"ל של setTextColor וגם של hideCursor
- לקבל ב-command-line פרמטר quiet– שקובע שלא תהיה הדפסה של הריצה למסך במידה ולא מתקבל פרמטר זה אז כברירת מחדל תהיה הדפסה למסך של מהלך הריצה!
- יש להציג את הספינות של שני השחקנים באופן צבעוני, אנימציה של "הפצצה" באמצעות סימן @ או סימן אחר שתבחרו, סימון של פגיעה באמצעות * או סימן אחר שתבחרו
- מותר לסמן את כלי השיט של השחקנים השונים באופן שתבחרו ובלבד שיהיה ברור למי שייך כל כלי
- יש לקבוע זמן המתנה סביר כברירת מחדל בין מהלך למהלך, על מנת לאפשר צפייה בריצת המשחק ולאפשר לקבוע זמן שונה באמצעות פרמטר command-line: -delay <delay in ms>
- יש לאפשר סדר כלשהו של הפרמטרים ב-command-line

**הדפסות בסיום המשחק**

באופן זהה לתרגיל 1. כמו כן, בתרגיל הזה - כמו בתרגיל 1 - ההרצה היא של משחק בודד.

**דגשים:**

א. כל מה שנכתב בדף "שינויים תיקונים והבהרות לתרגיל 1" ב-moodle תקף גם לתרגיל זה (אלא אם נכתב במפורש אחרת בתרגיל זה).
ב. יש לעקוב אחר הדף "שינויים תיקונים והבהרות לתרגיל 2" והפורום ב-moodle שכן הם חלק בלתי נפרד מהנחיות ודרישות התרגיל.
ג. ניתן להשתמש רק בספריות סטנדרטיות ובאלו שאושרו בתרגיל 1 (במידת הצורך אפשר לברר בפורום – moodle אם עולה צורך בספריות נוספות).
ד. טרם ההגשה, יש לפתוח solution חדש נקי, ולוודא שהפרוייקט מתקמפל לכם ללא שום הגדרות חדשות ושינויים בו, שכן בדיוק כך אנחנו נבדוק את ההגשות שלכם.
ה. עליכם להימנע מלהגיש כל קובץ שאינו קשור לתוכנית שלכם, ובפרט stdafx.h אשר נוצר אוטומטית לכל מי שלא פתח פרוייקט ריק.
ו. התוכניות תיבדקנה מקומפלות ב-x64 Release. עקרונית כל עוד לא ביצעתם בקוד משהו ספציפי שמניח אחרת (למשל מניח גודל של פוינטר) לא אמורה להיות השפעה כלשהי, עם זאת מומלץ שתקמפלו גם אתם במהלך הבדיקה שלכם באותה תצורה.

**מה מגישים (ומה לא!):**

א. קובץ ZIP בשם:
**ex1_<id1>_<id2>_<id3>.zip**
לדוגמא: ex2_123456789_987654321.zip (כאן יש רק 2 סטודנטים).
**ב. רק סטודנט אחד מכל קבוצה צריך להגיש את התרגיל!!!**
**(יינתן קנס על הגשות כפולות בגין העבודה המיותרת שהדבר מייצר)**
ג. הקובץ יכיל:
a. **כל** קבצי ה- cpp. וה-h. של התכנית כולל הקובץ IBattleshipGameAlgo.h ושל שלושת הספריות הדינאמיות. **ללא קבצים מיותרים ולא נחוצים כגון stdafx.h - הקפידו להתחיל מפרוייקט ריק!**
b. קובץ בשם: "students.txt" ובתוכו היוזר האוניברסיטאי שלכם ותעודת הזהות, לדוגמא:
itzikkobra 123456789
heizenberg 987654321
c. קובץ CMakeLists.txt מה-moodle – עליכם לעדכן בראש הקובץ את כל המשתנים המוגדרים באיזור לעריכה, במקומות המתאימים, ובפרט:
   i. יוזר אוניברסיטאי
   ii. ת.ז.
   iii. רשימת **כל** כל קבצי ה- cpp. וה-h. שנדרשים ל-exe.
   iv. 3 רשימות נפרדות של **כל** קבצי ה- cpp. וה-h. שנדרשים – בנפרד - לכל אחד משלושת ה-dll-ים.

   CMake היא מערכת בניית פרוייקטים (מזכיר את makefile בלינוקס) שנועדה להקל על תהליך הבדיקה האוטומטי. מלבד לעדכן את הקובץ, אין לכם שימוש בו.
d. קבצי קלט שאיתם בדקתם את ההגשה שלכם: קובץ sboard. אחד ושני קבצי attack. – כעיקרון לא יעשה שימוש בקבצים אלו, אבל – במידה ומסיבה כלשהי לא נצליח להריץ את התרגיל שלכם נרצה לבדוק שהוא עובד לפחות עם קבצי קלט סבירים שלכם.
e. **נא לא לספק קבצי dll מוכנים – אנחנו נייצר אותם מתוך הקוד שלכם!**