תרגיל 4

Code Race

הגשה בזוגות או יחידים בלבד.

תאריך אחרון להגשה : 7.1.2016

.Can use string & stl libraries

תיאור התרגיל :

בתרגיל זה נתרגל את עקרונות ההורשה והפולימורפיזם עם תיבלון של אלגוריתמיקה.

רקע סיפורי לתרגיל :

ערב אחד ביער, הצב, הארנבת והצפרדע החליטו לערוך בינהם מירוץ מקצה אחד של היער ועד לקצהו השני, כאשר המנצח יזכה בתואר "נסיך הניווטים המקומי".

רגע קצר לפני שהחל המירוץ, הגיע הקרנף עם להקתו והוכיח מי מלך הג'ונגל המקומי, במקום.

הינשופים אשר היו עדים למקרה המצער אך נמנעו מלהתערב, התווכחו בינם לבין עצמם מי היה עלול לנצח לולא הופעתו הלא צפויה של הקרנף.

לאחר דיון הוחלט לנסות ולשחזר את תנאי התחרות בתוכנת מחשב ולהריץ עליו סימולציות עד אשר יקבלו תשובה מכרעת.

בתוכנית זו תממשו תוכנית אשר תוכל לשחזר את המירוץ.

מחלקות :

מחלקת בסיס 1 :

- Tile - מחלקה זו תהיה מחלקת הבסיס של מסלול המירוץ והיא תייצג משבצת.
  יורשים מ Tile :
  - EasyTile
  - NormalTile
  - CropsTile
  - WaterTile

מתודות עיקריות :

onEnter (Competitor&c) :
מתודה זו תדאג להפעיל את הפעולה המתאימה אצל כל מתחרה כאשר הוא מודיע למשבצת שהוא נכנס אליה.

getType() :
מתודה זו תדאג להחזיר את סוג המשבצת. יהיה מותר להשתמש במתודה זו **_רק_** במתודת lookAhead של מחלקת competitor

whoAmI() :
מתודה זו תדפיס למסך את סוג המשבצת.

מחלקת בסיס 2 :

- Competitor - מחלקה זו תייצג מתחרה במירוץ.

  יורשים מCompetitor :
  - Turtle
  - Rabbit
  - Frog

מתודות עיקריות :

moveTo(tile) :
מתודה זו תזיז את השחקן למשבצת ותדאג להודיע למשבצת אליהם הם עוברים שהמתחרה כרגע נמצא במשבצת.

lookAhead() :
מתודה זו תבדוק לאיזו משבצת כדאי למשתתף (עדיפות לזולה יותר) ותשלח למתודת moveTo את המשבצת הנבחרת. אין לזוז באלכסון. יש לצאת מנקדות הנחה כי לכל משתתף עדיפות אחרת.

run(), walk(), eat(), swim() :

מתודות אלו ייקראו מהמשבצות המתאימות (מים - שחיה, קל - ריצה וכו') ויידאגו לעדכן את סך העלויות בכמות הנכונה.

מהלך המשחק :

מסלול המירוץ יהיה מורכב ממשבצות (Tiles). את הרכב המשבצות תקבלו בקובץ (ראו קובץ מצורף. שימו לב, קובץ זה הוא לא קובץ הבדיקה, על מסלול המירוץ להיות בנוי בצורה דינמית עם קבלת הקובץ).

לכל משתתף תהיה חוקיות תנועה משלו.

כל המשתתפים מתחילים מאותה נקודת התחלה.

כל המשתתפים מסיימים באותו קו סיום.

התנועה תתרחש בתורת, בכל תור יזוזו כלל המשתתפים ואז יחל תור חדש.

חישובי עלות התנועה יהיו מבוססים על הטבלה הבאה :

משבצת \ משתתף	EasyTile	NormalTile	CropsTile	WaterTile
Frog	2	1	1	0.5
Rabbit	0.5	1	2	1
Turtle	0.75	1	1	1

- כל עלות של משבצת תיספר אצל המשתתף, בסוף המירוץ, המשתתף עם העלות הנמוכה ביותר ייבחר כ"היה יכול להיות נסיך הנווטים".
- המירוץ מסתיים כאשר המשתתף האחרון הגיע לקו הסיום (נניח, אם המשתתף הראשון שלכם הגיע לסיום, יש לחכות להגעת האחרים גם עד להכרזת המנצח).
- כל משתתף ישמור את רשימת המשבצות בהן עבר. בסוף המירוץ יודפס למסך המסלול(טקסטואלית) של המשתתפים בתוספת מקומו במירוץ. (טיפ : >>)
- אין בעיה אם משתתפים יהיו באותה המשבצת בזמן המירוץ.
- כל משתתף יכול לנוע באיזה כיוון (מלבד אלכסונים) שיבחר כל זמן שהנקודה אינה מחוץ למפה.

- ניתן את משבצות ההתחלה והסוף לסמן גם כערך enum כלשהו ע"מ שיהיה נוח יותר לעבודה.
- בכלל יהיה מומלץ אם את משבצת הסיום גם הייתם מורישים/ות וממשים/ות גם אצלו את מתודת onEnter שתדאג לכך שהמשתתף יידע שהוא סיים איכשהו.
- כאשר יגיע למשבצת חדשה, המשבצת תפעיל על המתחרה את המתודה המתאימה להיותו עליה.
- **עליכם לממש מתודת lookup אשר תבדוק את סביבת המתחרה ותדאג לקרב אותו אל עבר היציאה.**

<u>בונוס :</u>

- שימוש ב : A*, BFS, DFS or Dijikstra על מנת למצא את דרך המתחרה.
- תצוגה ויזואלית של התחרות. (כל תור יחכה ללחיצה של המתשמש וידפיס למסך את המפה עם המשתמשים, במקרה של 2 מתחרים או יותר על אותה המשבצת, יש להדפיס את מספר הדיירים במשבצת.

<u>דגשים :</u>

- להשתמש בעקרונות הפולימורפיזם.
- שימושים בDeep & Shallow Copy.
- אין להשתמש במתודת getType() ממחלקת המשבצת בשום מתודה מלבד LookUp. (כאשר מחפשים את המשבצת הבאה).
- הבודק אלרגי להעתקות.
- לזכור סדר הריסה של אובייקטים כאשר משחררים מצביעים.

<u>קישורים מומלצים :</u>

https://en.wikipedia.org/wiki/Dijkstra's_algorithm

https://en.wikipedia.org/wiki/A*_search_algorithm

https://en.wikipedia.org/wiki/Depth-first_search

שיהיה בהצלחה ולמידה פוריה!

- אריאל.