אקסלנטים באקדמיה – הדסה

סמסטר ב', תשפ"ב

משימות לשבוע 3-4

תאריך הגשה

עד יום א', ב' ניסן תשפ"ב, 04/04/2021, בשעה 23:59

הנחיות כלליות

בכל המשימות, התבססו בבקשה על הקבצים לדוגמה שבאתר. שימו לב לקובצי ההנחיות על שימוש ב־CMake כדי להבין איך להשתמש ב־CMake ואיך להתאים את קובצי הדוגמה למשימות שתגישו. כל משימה תהיה בתיקייה נפרדת, עם קובץ CMakeLists ראשי משלה. שימו לב ששם התיקייה יתאים לשם המשימה. כל התיקיות האלה תהיינה יחד בתוך תיקייה ראשית בשם tasks_week3_student_name (כש־student_name מוחלף בשם המגיש/ה, כמובן).

להגשה, נמחק מתוך כל תיקיית משימה את התיקיות vs. ו־out, נכווץ את התיקייה הראשית הנ"ל לקובץ ZIP יחיד בשם tasks_week3_student_name.zip (כלומר, אותו שם כמו התיקייה הראשית) ונעלה למודל למשימה המתאימה. שימו לב שאם קובץ ה־ZIP שוקל יותר מכמה מאות KB, כנראה שלא מחקנו את כל התיקיות שצריך למחוק לפני ההגשה.

גן חיות (Zoo) – חובה

במשימה הזו נממש תוכנית המציגה את החיות שבגן חיות מסוים וייחודי. התוכנית תיצור אובייקטים המייצגים את החיות ותציג מידע אודותיהם.

מטרות

תרגול של מחלקות, ירושה ופולימורפיזם.

תיאור כללי

התוכנית תציג מפה של גן החיות ולאחריה רשימה ממוספרת של החיות שבגן החיות ותמתין לפקודה. המפה והרשימה יכילו בהתחלה את רשימת החיות המופיעה בהמשך. הרשימה תכיל את שם החיה, הסוג והמיקום (קואורדינטות, מספר השורה ואז מספר העמודה). הרשימה לא בהכרח יציבה. כלומר, כתלות במימוש שתבחרו לבצע, ייתכן שהרשימה מדפיסה את החיות לפי סדר שבו הן מופיעות במפה (סדר שמשתנה כשהחיות זזות). אחרי כל פקודה, כל החיות זזות (אלא אם נאמר אחרת, ראו להלן ברשימת הפקודות). התזוזה עצמה תהיה כמפורט להלן ותלויה בסוג החיה. שימו לב שהגדרת התזוזה היא עבור כל "תור". לא נעבוד כאן לפי שעון. כלומר, אם לא התקבל שום קלט, החיות לא זזות.

תצוגת המפה

מכיוון שהתרגיל מוצג בטרמינל, ללא גרפיקה, מיקום של חיה יסומן בעזרת האות הראשונה של שם המחלקה. אם יותר מחיה אחת נמצאת באותו המיקום, לא משנה איזו חיה תוצג במפה (זה תלוי בצורת המימוש). גודל הלוח הוא 20*40 משבצות (כלומר, 20 שורות ו־40 עמודות).

הפקודות

הפקודות האפשריות הן:

1. stop – לגרום לחיה לעצור. הפקודה תקבל כפרמטר את מספר החיה (כפי שמופיע ברשימה). החיה הזו לא תזוז מכאן והלאה, אלא אם תופעל עליה שוב פקודת move.
2. move – לגרום לחיה לזוז. הפקודה תקבל כפרמטר את מספר החיה (כפי שמופיע ברשימה). לפקודה יש משמעות רק אם החיה עצרה לפני כן. כשהחיה מתחילה לזוז, היא מגרילה מחדש את הכיוון שלה, ושוב, כפי שמתואר בהמשך.
3. create – יצירת חיה נוספת הפקודה תקבל כפרמטרים את סוג החיה ושם עבורה. המיקום של החיה יוגרל אקראית.
4. del – מחיקת חיה מהרשימה. מקבלת כפרמטר את מספר החיה (כפי שמופיע ברשימה).
5. delAll – מחיקת כל החיות מסוג מסוים (כנראה הן לא מתאקלמות היטב בגן והוחלט להעביר אותן לגן אחר). הפקודה תקבל כפרמטר את סוג החיה ותמחק מהרשימה את כל החיות מהסוג הזה (אם קיימות).
6. help – הדפסת טקסט עזרה המסביר מה הפקודות האפשריות, מה הפרמטרים שלהן ומה הן עושות.
7. exit – יציאה מהתוכנית.
8. . (נקודה) – הפקודה הזו תשמש כדי לגרום לתוכנית להתקדם לתור הבא, בלי לשנות שום דבר. כלומר, כל החיות שבתנועה ממשיכות לזוז לפי הכללים.

אחרי ביצוע הפקודה (כולל עדכון המפה והרשימה, אם יש צורך), נדפיס מחדש את המפה והרשימה המעודכנות ושוב נמתין לפקודה (אלא אם הפקודה הקודמת הייתה פקודת יציאה, כמובן).

רשימת החיות ההתחלתית

הרשימה הבאה מציגה את השמות והסוגים של החיות שניצור בתחילת התוכנית, עוד לפני שנקבל קלט כלשהו, כך שהן יהוו את הרשימה ההתחלתית. המיקום שלהן יוגרל אקראית.

- Simba – Lion
- Mufasa – Lion
- Rafiki – Monkey
- Akka – Goose
- Morten – Goose
- Hedwig – Owl
- Toto – Dog
- Jaws – Shark
- Nemo – Clownfish

תזוזה

- אריה צועד תמיד שתי משבצות ותמיד ימינה או שמאלה (כארי בסוגר). הוא שומר על אותו הכיוון עד שהוא מגיע לקצה הלוח ואז מסתובב בחזרה וכן הלאה, או עד שהוא נעצר (בעקבות stop) ואז בפקודת move הבאה נגריל מחדש את הכיוון.
- קוף קופץ באופן אקראי משבצת או שתיים לכיוון כלשהו בקווים ישרים (שמוגרל גם הוא). בשונה מהאחרים, אצלו ההגרלה של גודל הצעד ושל הכיוון מתרחשת מחדש בכל חמישה תורים, בלי קשר ל־move.
- אווזים צועדים בצעד מתנדנד, משבצת אחת "קדימה" ומשבצת אחת ימינה או שמאלה, לחילופין בכל תור; כלומר אם האווז בנקודה (5,6) (כלומר, שורה 5 ועמודה 6) בכיוון לימין הלוח והוא התנדנד קודם שמאלה,

הצעדים הבאים שלו יהיו (6,7) ואז (5,8) ואחריו (6,9). הם שומרים על אותו הכיוון עד שמגיעים לקצה הלוח ואז מסתובבים בחזרה. הכיוונים מוגרלים מחדש בפקודת move.

- ינשוף עף בקווים אלכסוניים, 3 משבצות בכל פעם, וחוזר בחזרה כשמגיע לקצה הלוח. גם פה, הכיוון מוגרל מחדש בפקודת move.
- כלב רץ שלושה צעדים קדימה ואז חוזר אחד בתור הבא. הכיוון, כצפוי, מוגרל מחדש בפקודת move.
- כריש שוחה במהירות של 5 משבצות בכל פעם. הכיוון נבחר אקראית בין כל הכיוונים האפשריים, כולל האפשרויות של ישר או אלכסוני. גם פה ההגרלה של הכיוון היא רק בפקודת move.
- "דג ליצן" שוחה הרבה יותר לאט, רק משבצת אחת בכל פעם אבל הכיוון מתנהג כמו אצל הכריש.

תיכון

1. עבור כל סוג חיה שבחרנו לממש (ראו הערה 6), ניצור מחלקה בשם הזה.
2. כל המחלקות יורשות ממחלקת בסיס Animal.
3. לכל המחלקות יהיה בנאי שמקבל &const std::string, שיכיל את שם החיה, ו־&const Location שמכיל את המיקום ההתחלתי שהוגרל עבורה.
4. נגדיר את הפונקציות הבאות במחלקת Animal. ההחלטה על כל פונקציה האם היא ממומשת במחלקת Animal או שהיא pure virtual וממומשת במחלקות היורשות תלויה במקום שבו נחליט לשמור את ה־data members הרלוונטיים.
   a. void printDetails() const – מדפיסה את שם החיה, הסוג שלה והמיקום (לשימוש בהדפסת הרשימה)
   b. char getInitial() const – מחזירה את האות הראשונה בשם המחלקה, לשימוש להדפסה במפה.
   c. Location getLocation() const – מחזירה את המיקום של החיה.
   d. void step() – גורם לחיה להתקדם צעד (לפי הכללים של אותה החיה), בתנאי שהחיה כרגע בתזוזה (לא עצרנו אותה קודם לכן)
   e. void stop() – גורם לחיה לעצור, כך שקריאות נוספות ל־step() לא יגרמו לה לזוז.
   f. void move() – גורם לחיה לעבור למצב תנועה, כלומר אם כעת נקרא ל־step() היא אכן תזוז.
   g. void turnVertically() – גורם לחיה לשנות כיוון אנכי (כלומר, אם הייתה בתנועה כלפי מעלה, להתחיל לנוע מטה, ולהיפך), בלי לשנות מהירות.
   h. void turnHorizontally() – גורם לחיה לשנות כיוון אופקי (כלומר, אם הייתה בתנועה ימינה, להתחיל לנוע שמאלה, ולהיפך), בלי לשנות מהירות.
5. נגדיר struct בשם Location, שיכיל מספרי שורה ועמודה. המספרים יכולים להיות שליליים (שימושי לצורך הזזה בכיוון שמאל/למעלה, לשימוש עם אופרטור החיבור). הוא צריך לתמוך בהדפסה בעזרת אופרטור הדפסה (אופרטור >>) ולתמוך באופרטור חיבור ו־+= וכן באופרטורים == ו־=!.
6. נגדיר מחלקת Zoo שתנהל את המשחק. היא צריכה להכיל את רשימת החיות או את המפה (לא נחזיק מידע כפול!). נגדיר בה את הפונקציות הבאות:
   a. אפשר להגדיר בנאי אם יש דברים שנרצה לאתחל בו.
   b. void run() – הפונקציה הראשית שמופעלת מה־main. זו הפונקציה היחידה (מלבד הבנאי, אם נבחר לממש אותו) שתהיה public. כל שאר הפונקציות שלהלן הן private.
   c. עבור כל פקודה מרשימת הפקודות, ניצור פונקציה בשם זהה. הפונקציות האלה תהיינה private, כאמור. לבחירתנו על כל פונקציה האם היא מחזירה משהו או לא (לא בהכרח כל הפונקציות זהות בזה). נשים לב האם הפונקציה צריכה להיות const או לא. עבור פקודת . (נקודה) נקרא לפונקציה בשם step. עבור פקודת exit אין חובה ליצור פונקציה נפרדת.
7. ההחלטה איזה מידע יהיה שמור ואיפה נשארת פתוחה. זה חלק חשוב מהאתגר לתרגיל הזה.
8. אם יש צורך בזה במחלקות שלנו, נוסיף הורס ומימושים נכונים עבור פעולות העתקה (או הגדרה שלהם כ־delete). אם אין צורך – מה טוב.

9. מותר להוסיף פונקציות עזר חיצוניות (שלא חברות באחת המחלקות הנ"ל) או פונקציות עזר שהן private במחלקה. אסור להוסיף פונקציה ציבורית (public) נוספת לאף אחת מהמחלקות (למעט Location שגם ככה הכול שם public, כי זה struct) מעבר למה שהוזכר לעיל.

הערות

1. יש לשקול היטב את התיכון (design) של התוכנית, במיוחד אלו data members ואלו member functions מוגדרות בכל מחלקה ואיפה הפונקציות ממומשות.
2. נוודא שהקוד שלנו בדוק היטב ועובד כראוי, מבחינת ההתנהגות הרצויה ומבחינת ניהול הזיכרון (שכעת לא צריך להיות ידני, ולמעשה מומלץ שלא יהיה כזה).
3. בהמשך לסעיף הקודם, המחלקות std::unique_ptr, std::vector ו־std::string יכולות להיות מאוד שימושיות. השתמשו בהן בתבונה.
4. יש לוודא שהקלט תקין מהבחינה שלא קיבלנו פקודה שאינה קיימת, שם מחלקה שאינה קיימת או מספר חיה שלא קיים ברשימה. אין צורך להתמודד עם קלט לא צפוי במובן של הכנסת מילה כשציפינו למספר (למספר של חיה).
5. להגרלה של מספר רנדומלי אפשר להשתמש בפונקציית ()rand המוכרת. שימו לב להשתמש בפונקציית ()srand פעם אחת **בדיוק**, עדיף בתחילת ריצת התוכנית, כדי לאתחל את ה־seed של ()rand. דרך אחרת היא להשתמש בכלים של C++11 (ראו תיעוד כאן, בעיקר הדוגמה שבסוף הדף).
6. **עדכון: מספיק לממש רק 3 סוגים של חיות מתוך הרשימה לעיל. בהתאמה, קחו מרשימת החיות ההתחלתית רק את אלו המתאימות לסוגים שתחליטו לממש.**