מכללת הדסה, החוג למדעי המחשב

תכנות מונחה עצמים ופיתוח משחקים

סמסטר ב', תשע"ט

תרגיל 3

תאריך אחרון להגשה:

קמפוס הנביאים: יום א', ל' ניסן, 05/05/2019 בשעה 23:59
קמפוס שטראוס גברים: יום ז', כ"ט ניסן, 04/05/2019 בשעה 23:59
קמפוס שטראוס נשים: יום ב', א' אייר, 06/05/2019 בשעה 23:59

מטרות התרגיל:

תרגול חריגות (exceptions).

תיאור כללי:

תרגיל זה יתבסס על הפתרון של תרגיל 1 (שתקבלו מאיתנו), כאשר אתם נדרשים "לשדרג" את הפתרון הזה באמצעות טיפול בחריגות, ולהפוך אותו לעמיד לטעויות של משתמשים לא זהירים. זה יכלול מעבר על כל הפקודות וכן על המימוש שאולי עלול לזרוק שגיאות עם קלטים לא נכונים. נשתמש בתרגיל זה רק בחריגות של השפה (std::exception והמחלקות היורשות ממנה) או ניצור מחלקות שיורשות מהן במקרה הצורך.
בתרגיל זה נשתמש במסוף (terminal) בלבד.

פירוט הדרישות:

1. כל השגיאות שנזכיר כאן, ושתטפלו בהן, **חייבות** להשתמש בחריגות (exceptions) כדי להודיע על הבעיה. בדרך כלל נעדיף שהחריגה תיזרק מתוך הפונקציה או המחלקה הממשת את הפעולה ושהיא תיתפס ותטופל באחת הרמות מעליה, ברמה הכי גבוהה האפשרית.

2. עבור הפקודות הקיימות כבר מתרגיל 1, עליכם לעבור עליהם ולוודא שהמשתמש לא הכניס פקודה שגויה או לא קיימת. עבור פקודה תקינה, עליכם לוודא שהמשתמש לא הזין נתון לא תקין (כמו אותיות או מספרים שליליים או לדוגמא: כשיש צורה אחת בלבד הפקודה `area 0` תקינה, אך הפקודות `area 1` וגם `del 2` אינן תקינות). עליכם לוודא שהמשתמש לא הזין נתון לא קיים (למשל `cre t` הוא חוקי אבל `cre x` לא חוקי). או אם הקלט אינו מספק את כל הנתונים (רלוונטי בעיקר בקריאה מקובץ כפי שנבאר בסעיף הבא).

בכל המקרים הללו התכנית תתריע עם הודעת שגיאה מתאימה ותציג את רשימת הפקודות מחדש.

3. נוסיף לרשימת הפקודות פקודת read שתקבל כפרמטר נתיב לקובץ עם פקודות, ושתדע לקרוא את הקובץ. הקובץ יכול להכיל את כל סוגי הפקודות, כאשר אין חובה לתמוך בפקודה read בעצמה. נוודא שהתוכנית שלנו יודעת להתמודד גם במקרים שהקובץ אינו קיים או אינו ניתן לקריאה, ובמקרים כאלה נציג הודעת שגיאה, כמו בפקודות אחרות שנכשלות, ונחזור להצגת רשימת הצורות כרגיל.
4. אם פקודת read לא הצליחה לפתוח את הקובץ, היא תציג הודעת שגיאה. אם היא הצליחה לפתוח את הקובץ הנתון, היא תקרא פקודה אחרי פקודה ותבצע אותן, כל שורה היא פקודה נפרדת (ניתן להעזר ב stringstream). אם ביצוע אחת השורות נכשל (מאחת הסיבות המוזכרות בסעיפים האחרים), נפסיק את פעולת הקריאה, נציג את השורה השגויה בתוספת הודעת שגיאה מתאימה, ונציע למשתמש לבחור האם להמשיך לקרוא ולבצע את המשך הקובץ, או להפסיק לקרוא ולחזור למצב הקודם.
5. בפתיחת התוכנית, לפני הצגת הפקודות, נבקש מהמשתמש לקבוע מה מספר הצורות (כלומר מספר האיברים ברשימה) המקסימלי שניתן יהיה לאחסן בתכנית זו . עליכם לוודא שמספר הצורות המקסימלי לא גדול מ-100. גם פה נוודא שהקלט תקין ואם הוא לא תקין (כלומר אם הקלט גדול מ-100, או הקלט קטן או שווה 0, או הקלט כלל לא מספר, אלא אותיות וכדומה) נציג שגיאה ונשאל שוב. במהלך התכנית, נוודא שלא נוצרות צורות יותר ממה שנקבע מלכתחילה ובהצגת רשימת הצורות נדפיס גם את מספר הצורות המקסימלי. נשים לב שהמשתמש יכול למחוק חלק מהצורות מהרשימה וכך לפנות מקום ליצור צורות נוספות בלי להגיע למגבלה שנקבעה. גם במהלך קריאת הקובץ צריך להתחשב במגבלת המקום ולהימנע מביצוע פקודה שתגרום להוספת צורה כשהרשימה מלאה (ואז נפעל כמתואר בסעיף 3, נשאל את המשתמש אם להמשיך או להפסיק את ביצוע הקובץ).
6. פקודה נוספת שתתווסף לרשימת הפקודות היא resize, בה יוכל המשתמש לשנות את מכסת הצורות כרצונו. אם הגודל החדש שבחר המשתמש יהיה קטן ממספר הצורות הקיימות כרגע, נציג למשתמש הודעה מתאימה שתציע לו לבחור גודל חדש או למחוק את הצורות שבסוף הרשימה עד למכסה המותרת.

הערות לתיכון התוכנית:

- ברור שעליכם לחשוב על תכנון ראוי שיתאים לקוד הקיים כדי לאפשר לו להתקמפל ולרוץ באופן תקין, אולם חשוב שלא תצמצמו ואף תשנו מבנים במקרה הצורך. אולי כדאי לחשוב על מחלקות חדשות שיעטפו אזורים בקוד ובכך קליטת השגיאות תהיה קלה בהרבה.
- נראה שכדאי להיעזר בחריגות המובנות של ה-streams למיניהם.
- כדאי לשנות את הקוד בשלבים כך שבכל שלב אפשר לוודא שהקוד עדיין מתקפל והשגיאות מתקבלות בהצלחה כמצופה.

קובץ ה-README:

יש לכלול קובץ README שיקרא README.doc, README.docx או README.txt (ולא בשם אחר).

קובץ זה יכיל לכל הפחות:

1. כותרת.
2. פרטי הסטודנט: שם מלא כפי שהוא מופיע ברשימות המכללה, ת"ז.
3. הסבר כללי של התרגיל.
4. **תיכון (design): הסבר קצר מהם האובייקטים השונים בתוכנית, מה התפקיד של כל אחד מהם וחלוקת האחריות ביניהם ואיך מתבצעת האינטראקציה בין האובייקטים השונים.**
5. רשימה של הקבצים שנוצרו ע"י הסטודנט, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.
6. מבני נתונים עיקריים ותפקידיהם.
7. אלגוריתמים הראויים לציון.
8. באגים ידועים.
9. הערות אחרות.

יש לתמצת ככל שניתן אך לא לוותר על אף חלק. אם אין מה להגיד בנושא מסוים יש להשאיר את הכותרת ומתחתיו פסקה ריקה. תכתבו ב-README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.

אופן ההגשה:

הקובץ להגשה: יש לדחוס כל קובץ הקשור לתרגיל, למעט מה שיצוין להלן, לקובץ ששמו exN_name.zip, כאשר N הוא מספר התרגיל ו־name הוא השם המלא. במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית

exN_name1_name2.zip, עם שמות המגישים בהתאמה (ללא רווחים; גם בשמות עצמם יש להחליף רווחים בקו תחתי או להצמיד את שני חלקי השם).

לפני דחיסת תיקיית ה־Solution שלכם יש למחוק את הפריטים הבאים:

- תיקיות בשם debug, release או x64 (לרוב יש יותר מתיקייה אחת בשם זה, אחת בתיקיית ה־Solution ואחת בתיקיית כל פרויקט).
- תיקייה (לפעמים מוסתרת!) בשם vs. שאמורה להיות בתיקייה עם ה־Solution.

ככלל אצבע, אם קובץ ה־zip שוקל יותר ממ"ב אחד או שניים, כנראה שלא מחקתם חלק מהקבצים הבינאריים המוזכרים.

את הקובץ יש להעלות ל־Moodle של הקורס למשימה המתאימה.

הגשה חוזרת: אם מסיבה כלשהי סטודנט מחליט להגיש הגשה חוזרת יש לוודא ששם הקובץ זהה לחלוטין לשם הקובץ המקורי. אחרת, אין הבודק אחראי לבדוק את הקובץ האחרון שיוגש.

כל שינוי ממה שמוגדר פה לגבי צורת ההגשה ומבנה ה־README עלול לגרור הורדת נקודות בציון.

מספר הערות:

1. שימו לב לשם הקובץ שאכן יכלול את שמות המגישים.
2. שימו לב שעליכם לשלוח את תיקיית ה־solution כולה, לא רק את קובצי הקוד שיצרתם. עקבו אחרי ההסבר המפורט באתר, במקרה שאתם לא בטוחים איך למצוא את התיקייה. תרגיל שלא יכלול את ה־solution, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

המלצה כללית: אחרי שהכנסתם את הקובץ להגשה, העתיקו אותו לתיקייה חדשה, חלצו את הקבצים שבתוכו ובדקו אם אתם מצליחים לפתוח את ה־solution שבתוכו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!