מכללת הדסה, החוג למדעי המחשב
תכנות מונחה עצמים ופיתוח משחקים
סמסטר ב', תשפ"ב

תרגיל 3

תאריך אחרון להגשה:

הנביאים - יום א', 10/04/22, בשעה 23:59

שטראוס-גברים - מוצאי שבת, 09/04/22, בשעה 23:59

שטראוס-נשים - מוצאי שבת, 09/04/22, בשעה 23:59

מטרת התרגיל:

תרגול חריגות (exceptions).

תיאור כללי:

תרגיל זה יתבסס על הפתרון של תרגיל 1 (שתקבלו מאיתנו), כאשר אתם נדרשים לשנות ו"לשדרג" את הפתרון הזה באמצעות טיפול בחריגות, ולהפוך אותו לעמיד בפני טעויות של משתמשים לא זהירים. השינוי יכלול שיפור של מימוש הפקודות הקיימות כדי לזרוק שגיאות במקרה של קלט שגוי. כמו כן נוסיף אפשרות לקריאת פקודות מקובץ נתון וביצוע שלהם. נשתמש בתרגיל זה רק בחריגות של השפה (std::exception והמחלקות היורשות ממנה) או, במקרה הצורך, ניצור מחלקות שיורשות מהן.

גם בתרגיל זה נשתמש במסוף (terminal) בלבד.

פירוט הדרישות:

1. כל השגיאות שנזכיר כאן ונטפל בהן, **חייבות** להשתמש בחריגות (exceptions) כדי להודיע על הבעיה. בדרך כלל נעדיף שהחריגה תיזרק מתוך הפונקציה או המחלקה המממשת את הפעולה ושהיא תיתפס ותטופל באחת הרמות מעליה, ברמה הכי גבוהה האפשרית.
2. כשהמשתמש מכניס פקודה, יש לוודא שהמשתמש לא הכניס פקודה שגויה או לא קיימת. אם אין פקודה כזו, התכנית תתריע עם הודעת שגיאה מתאימה ותציג את רשימת הפעולות מחדש.
3. עבור פקודה תקינה (הפקודות הקיימות מתרגיל 1 והפקודות הנוספות בהמשך), יש לוודא שהמשתמש לא הזין נתון לא תקין (כמו אותיות או מספרים שליליים כשמצפים למספר חיובי, מספר פעולה לא תקין

וכו'). כך גם אם הקלט לא מכיל מספיק ארגומנטים לפקודה או שיש יותר מידי ארגומנטים לפקודה הנוכחית (רלוונטי בקריאה מקובץ כפי שמוסבר בסעיף הבא, אבל אולי גם בקריאה מהמקלדת, ראו בהערות בהמשך). בכל המקרים הללו התכנית תתריע עם הודעת שגיאה מתאימה ותציג את רשימת הפעולות מחדש.

4. נוסיף לרשימת הפקודות פקודת read שתקבל כפרמטר נתיב לקובץ עם פקודות, ושתדע לקרוא את הקובץ. הקובץ יכול להכיל את כל סוגי הפקודות, למעט הפקודה read עצמה שאין חובה לתמוך בה (ציינו ב־Readme האם היא נתמכת בקריאה מקובץ או לא).
5. 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 (כך שייתכן שיש יותר מידי או פחות מידי ארגומנטים עבור הפקודה המסוימת). 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.
6. כשנסיים את המעבר על הקובץ, בין אם בהצלחה ובין אם בגלל שגיאה (וכמוסבר בסעיף הקודם), נחזור לשלב הקודם. כלומר, אם לפני כן היינו במצב של קריאה מהמקלדת, נחזור להדפיס את רשימת הפעולות ולקרוא פקודות מהמקלדת. אם היינו לפני כן בקריאה של קובץ אחר, ומימשנו תמיכה בפקודת read מתוך קובץ, נחזור לבצע את הקובץ הקודם.
7. בפתיחת התוכנית, לפני הצגת הפעולות, נבקש מהמשתמש לקבוע מה מספר הפעולות המקסימלי שניתן יהיה לאחסן בתכנית זו. נוודא שמספר הפעולות המקסימלי לא גדול מ־100 ולא קטן מ־3. גם פה נוודא שהקלט תקין ואם הוא לא תקין (כלומר אם הקלט גדול מ־100, קטן מ־3, או שהוא כלל לא מספר, אלא אותיות וכדומה) נציג שגיאה ונשאל שוב. במהלך התכנית, נוודא שלא נוצרות יותר פעולות ממה שנקבע מלכתחילה, כלומר אם המשתמש מנסה ליצור פעולה חדשה (ישירות או כפקודה המגיעה מקריאת הקובץ) והמכסה התמלאה כבר, הפקודה צריכה להיכשל. כמו כן, בהדפסת רשימת הפעולות, יש להציג את המידע על מספר הפעולות המותר. שימו לב שהמגבלה איננה על מספר הפעולות שנוצרות במהלך כל ריצת התוכנית, אלא על מספר הפעולות הנוכחי, כלומר אם המשתמש מוחק פעולות, הן כבר לא נספרות במכסת הפעולות המותרת. כאמור, גם במהלך קריאת הקובץ צריך להתחשב במגבלת המקום ולהימנע מביצוע פקודה שתגרום להוספת פעולה כשהרשימה מלאה (ואז נפעל כמתואר בסעיף 5, נשאל את המשתמש אם להמשיך או להפסיק את ביצוע הקובץ).
8. פקודה נוספת שתתווסף לרשימת הפקודות היא resize, בה יוכל המשתמש לשנות את מכסת הפעולות כרצונו. אותן בדיקות מסעיף 7 נדרשות גם פה. בנוסף, אם יקטין המשתמש את כמות הפעולות מהקיימות - זו איננה שגיאה (ולכן לא נדרוש שימוש בחריגה), והוא יקבל הודעה מתאימה שתציע לו ביטול של הפעולה או מחיקת הפעולות 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 תצטמצמו רק לשילוב הדרישות החדשות בקוד הקיים אלא גם תחשבו מחדש על מבנה הקוד, איחוד או פיצול של פונקציות מסוימות, וכדומה. אולי כדאי לחשוב על מחלקות חדשות שיעטפו אזורים בקוד ויעשו את הטיפול בשגיאות קל יותר.
- שקלו את האפשרות להשתמש בחריגות המובנות של ה־streams למיניהם.
- כדאי לשנות את הקוד בשלבים (קטנים ככל האפשר) כך שבכל שלב אפשר לוודא שהקוד עדיין מתקפל והשגיאות מתקבלות בהצלחה כמצופה.
- כדי לעבוד על כל שורה (מהקובץ) בנפרד, זכרו את הכלים std::getline ו־stringstream.
- בידכם להחליט האם הקריאה מהמקלדת גם היא תיעשה על כל שורה בנפרד. אם כן, צריך לטפל גם פה במקרים של יותר מידי או פחות מידי ארגומנטים לפקודה. אם לא – אין כזה דבר יותר מידי או פחות מידי ארגומנטים, נקרא מהמקלדת ארגומנטים עד שיש לנו מספיק עבור הפקודה הנוכחית וארגומנטים עודפים יישארו ב־buffer עד הפעם הבאה שננסה לקרוא מהמקלדת. בכל מקרה, ציינו את החלטתכם בקובץ ה־Readme.
- שימו לב שכשמתקבל קלט לא תקין מהמקלדת, ייתכן שהמימוש שלנו גורם לקבלת הקלט להפסיק ושאר הקלט יחכה לפעם הבאה וייתכן שקודם נקרא את המידע ורק אז נוציא את השגיאה וממילא כל הקלט עבור הפקודה הנוכחית "נפסל". למשל, אם הפקודה מצפה לקבל שני מספרי פעולות, והראשון מביניהם הוא מספר לא חוקי, ייתכן שנעצור מיד ולא נקרא בכלל את מספר הפעולה השניה, וייתכן שנקרא את מספר הפעולה השניה ורק אחר כך נבדוק ונגלה שהמספר הראשון איננו חוקי. שני המימושים האלה לגיטימיים, אין צורך להתאמץ במיוחד לממש כך או כך, ועדיף קוד קצר, מסודר וקריא מאשר קוד מורכב ומסובך רק כדי להשיג התנהגות מסוימת. בכל מקרה, ציינו את ההתנהגות שבחרתם בקובץ ה־Readme. נעיר כי בקריאה מהקובץ, אנחנו מחוייבים להתייחס לכל שורה בנפרד, ולכן בכל מקרה כל השורה "נפסלת" כשחלק מהמידע בה שגוי.
- חלק מהקלטים השגויים כבר מטופלים בקוד שתקבלו, כי אלה היו חלק מהדרישות המקוריות של תרגיל 1. למרות זאת, כעת תצטרכו להמיר את הזיהוי והטיפול הזה לשימוש בחריגות, כאמור. שקלו אם יש חלקי קוד שניתן לשפר בעקבות המעבר לשימוש בחריגות.

קובץ ה־README:

יש לכלול קובץ README שיקרא README.doc, README.docx או README.txt (ולא בשם אחר). הקובץ יכול להיכתב בעברית ובלבד שיכיל את הסעיפים הנדרשים.

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 ומתחתיה 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_firstname_lastname.zip, כאשר N הוא מספר התרגיל ו־firstname_lastname הוא השם המלא (לדוגמא אלברט איינשטיין יגיש כך את התרגיל הראשון: ex1_albert_einstein.zip). במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית exN_firstname1_lastname1_firstname2_lastname2.zip, עם שמות המגישים בהתאמה (ללא רווחים; כלומר, גם בשמות עצמם יש להחליף רווחים בקו תחתי, כפי המודגם לעיל). כמו כן, במקרה של הגשה בזוג, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

לפני דחיסת תיקיית הפרויקט שלכם יש למחוק את הפריטים הבאים:

- תיקייה בשם out, אם קיימת
- תיקייה בשם vs.

שתי התיקיות האלה נמצאות בתיקייה הראשית (זו שאנחנו פותחים בעזרת VS). התיקייה vs. לפעמים מוסתרת, אבל אם תפתחו את קובץ ה־zip שיצרתם, בוודאי תוכלו למצוא אותה ולמחוק אותה.
ככלל אצבע, אם קובץ ה־zip שוקל יותר ממ"ב אחד או שניים, כנראה שלא מחקתם חלק מהקבצים הבינאריים המוזכרים.

וודאו כי קובץ ה־zip מכיל תיקייה ראשית אחת, ורק בתוכה יהיו כל הקבצים ותתי התיקיות של הפרויקט.

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 הפרוייקט כולה, לא רק את קובצי הקוד שיצרתם. תרגיל שלא יכלול את כל הקבצים הנדרשים, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

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

בהצלחה!