מבוא לתכנות מערכות – 234124

תרגיל בית 3

סמסטר חורף 2024-2025

תאריך פרסום: 4.6.25
תאריך אחרון להגשה: 18.6.25 עד השעה 23:55
מתרגל אחראי לתרגיל: רון רפאלי

1 הערות כלליות

- תרגיל זה מהווה 6% מהציון הסופי בקורס.
- התרגיל להגשה בזוגות בלבד.
- כל ההודעות והעדכונים הנוגעים לתרגיל זה יפורסמו באתר הקורס ב-GR.
- מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל בפיאצה (קישור באתר הקורס) או בסדנאות. אין לשלוח דוא"ל לגבי התרגיל.
- שימו לב – לא תינתנה דחיות למועד הגשת התרגיל.
- על כל הקוד שאתם כותבים לעמוד בקונבנציות המפורטות בקובץ "קונבנציות לכתיבת קוד" הנמצא באתר הקורס. אי עמידה בקונבנציות עלולה לגרור הורדת ניקוד.
- קבצי התרגיל מסופקים לכם ב- GitHub.
- החומר הנדרש כדי לפתור את התרגיל הינו החומר הנלמד עד הרצאה ותרגול 7 (כולל). אין להשתמש ב-STL, כמו כן אין צורך להשתמש בירושה או בכל חומר שעוד לא למדנו.
- תיקונים למסמך התרגיל, יסומנו בצהוב.

2 הקדמה

המטרות של תרגיל בית זה הן כתיבת מחלקות גנריות בשפת ++C, שימוש בהן, ושימוש בקוד מסופק על ידי קריאת הממשק שלו.
מומלץ לפתור תחילה את החלק הרטוב ורק אז את החלק היבש. לכל שאלה או בקשה לעזרה בתרגיל ניתן לכתוב לנו בפורום הקורס בפיאצה או להגיע לסדנאות לעזרה בשיעורי הבית (זמנים ומיקומים מפורסמים באתר הקורס).
לפני פרסום שאלה בפורום אנא בדקו אם כבר נענתה – מומלץ להיעזר בכלי החיפוש שהוצגו במצגת האדמיניסטרציה בתרגול הראשון.

תוכלו למצוא את הקבצים המסופקים לתרגיל זה ב-GitHub. בדומה לתרגילי הבית הקודמים, עליכם ליצור Repo פרטי בחשבון ה-GitHub שלכם, המכיל את קבצי התרגיל כפי שסופקו.

3 חלק יבש

את התשובות לחלק זה יש לצרף בקבצים dry.pdf ו-log.txt, פירוט על כל אחד מהם בהמשך.

3.1 שאלות על SortedList

השאלות בסעיף זה של המסמך מתייחסות למבנה הנתונים שתממשו בחלק הרטוב של התרגיל.
עליכם לענות על השאלות הבאות ולהגישן בקובץ pdf בשם dry.pdf שיצורף להגשה הסופית.

1. במבנה SortedList הגנרי, מה הדרישות ההכרחיות שעל הטיפוס T לקיים? הסבירו במשפט לכל דרישה.

2. נניח כי היינו רוצים לממש איטרטור non-const עבור ה-SortedList. כלומר, עבור איטרטור זה, האופרטור* היה מחזיר T&. איזו בעיה עלולה להיווצר במימוש?

3. סטודנטית בקורס מבוא לתכנות מערכות סיימה לפתור את תרגיל בית 3 והחליטה להשתמש ברשימה הממויינת מהתרגיל לפרוייקט צד שהיא מפתחת בשעות הפנאי. במימוש פרוייקט הצד הסטודנטית נדרשה לסנן רשימה של מספרים שלמים, כך שישארו בתור רק מספרים המתחלקים במספר כלשהו <u>שאינו ידוע בזמן קומפילציה אלא רק בזמן ריצה</u>. הסבירו כיצד ניתן לממש את הפונקציונליות הדרושה בעזרת הפונקציה filter.

3.2 Git

בתרגיל זה, כמו בכל שאר התרגילים בקורס, עליכם להשתמש ב-Git במהלך העבודה על התרגיל. הקפידו על עבודה נכונה עם Commits ו-Branches כפי שנלמדה בהרצאות ובתרגולים.

<u>בסיום העבודה על התרגיל</u>, עליכם להדפיס את ה-log של ה-Repo לקובץ log.txt באמצעות הפקודה הבא:

```
git log --graph --all --stat > log.txt
```

צרפו את הקובץ log.txt ל-Repo (על ידי הוספת Commit נוסף) על מנת להגיש אותו.

4 חלק רטוב

לאור כמות הסטודנטים הגדולה בקורס מת"מ, ולאור העבודה הרבה שיש לסגל הקורס, החליטו המתרגלים לייצא חלק מהמשימות שלהם לגורמים חיצוניים (Outsourcing).לצורך כך, החליטו לממש מערכת שתעזור לנהל את המשימות ואת העובדים החדשים.

בחלק זה של התרגיל, תעזרו למתרגלים בקורס בכך שתממשו מודול לעבודה עם רשימות ממוינות (SortedList)ומודול לניהול משימות ועובדים (TaskManager).

4.1 הערות מימוש כלליות

- תוכלו למצוא דוגמאות שימוש לכלל הדברים אותם אתם נדרשים לממש בסעיפים אלו בקובץ main.cpp המסופק לכם.
- אין להוסיף דבר על הממשק אותו נתבקשתם להגדיר. בכל מקרה של שגיאה, על המשתמש לקבל חריגה מתאימה.
- על המודולים שאתם מממשים לתמוך באובייקטים קבועים ולא קבועים.

SortedList 4.2

ראינו בתרגולים ובהרצאות מספר מבני נתונים שמתאימים לפעולות שונות. בסעיף זה, נממש מבנה נתונים שהוא רשימה ממוינת. המשתמש במבנה הנתונים יכול להניח שכל פעם שהוא עובר את הרשימה, האיבר הגדול ביותר יהיה בהתחלה, והרשימה תהיה ממוינת עד האיבר האחרון שהינו הקטן ביותר.

מבנה הנתונים צריך לקיים את הדרישות הבאות:

1. כמות האיברים ברשימה אינה חסומה.
2. איברי הרשימה ממוינים בכל רגע נתון. הסדר בין האיברים מוגדר על ידי אופרטור < (גדול מ-) של טיפוס איברי הרשימה.
3. הרשימה יכולה להכיל כפילויות, ובמקרה כזה האיבר הישן יותר יהיה למעלה (כאילו הוא יותר גדול). (לדוגמה, אם נכניס לרשימה 1 ואז 1 ולבסוף 2 אז בראש הרשימה יהיה 2, לאחר מכן 1 ולבסוף 1)

4.2.1 ממשק המחלקה

על המחלקה לתמוך בפעולות הבאות:

1. בניה של רשימה ריקה באמצעות בנאי חסר פרמטרים.
2. העתקה והשמה של רשימות.
3. Insert - מתודה המקבלת אלמנט חדש כפרמטר ומכניסה אותו לרשימה.
4. remove - מתודה המקבלת איטרטור (יפורט בהמשך) כפרמטר, ומסירה את האלמנט אליו הוא מפנה מהרשימה.
5. length - מתודה המחזירה את מספר האלמנטים ברשימה.
6. filter - מתודה המקבלת פרדיקט (אופרציה המקבלת אלמנט מהרשימה ומחזריה ערך בוליאני) ומחזירה רשימה חדשה שמכילה רק את האיברים מהרשימה המקורית שהפעלת הפרדיקט עליהם מחזירה true.
7. apply - מתודה המקבלת אופרציה מטיפוס איברי הרשימה לטיפוס איברי הרשימה, ומחזירה רשימה חדשה המכילה את תוצאות הפעלת הפונקציה על כל איברי הרשימה.
8. begin - מתודה שמחזירה איטרטור לתחילת הרשימה.
9. end - מתודה שמחזירה איטרטור לסוף הרשימה ("אחרי" האיבר האחרון).

4.2.2 איטרטורים

לצורך איטרציה על איברי הרשימה נממש מחלקת איטרטור קבוע בשם ConstIterator, אשר יוגדר בתוך מחלקת הרשימה (כלומר שמו המלא SortedList::ConstIterator).

על האיטרטור לספק למשתמש את הממשק הבא:

1. (prefix) ++operator - מקדם את האיטרטור לאיבר הבא ברשימה. אם האיטרטור מצביע לסוף הרשימה, יש לזרוק חריגה מטיפוס std::out_of_range.
2. =!operator - מקבל איטרטור נוסף ומחזיר true אם ורק אם שניהם אינם שווים.
3. *operator - מאפשר גישה לאיבר אליו האיטרטור מפנה, לקריאה בלבד.

4.2.3 מימוש המודול

דגשים למימוש:

1. את המימוש עליכם לכתוב בקובץ SortedList.h.
2. יש לממש את המבנה כך שיהיה לא תלוי במשתמש מבחינת זיכרון. כלומר, עליו ליוצר עותקים משלו לכל האובייקטים שהוא מקבל מהמשתמש.
3. אתם רשאים להוסיף מתודות פרטיות ומחלקות כרצונכם, תוך שמירה על עקרונות תכנות נכון. אין לשנות את ממשק הרשימה או האיטרטור.
4. הרשימה יכולה להכיל ערכים שווים מבחינת האופרטור <. ובמקרה כזה האיבר הישן יותר יהיה למעלה (כאילו הוא יותר גדול). (לדוגמה, אם נכניס לרשימה 1 ואז 1 ולבסוף 2 אז בראש הרשימה יהיה 2, לאחר מכן 1 ולבסוף 1).
4.

בנוסף, אנו ממליצים להתחיל ממימוש של הרשימה כרשימה לא גנרית, ללא apply, filter. לאחר מכן לממש גנריות, ולבסוף לממש את filter ואת apply.

4.3 TaskManager

במודול זה עליכם להגדיר מחלקה בשם TaskManager. מחלקה זו תשמש לצורך ניהול משימות ועובדים. מסופק לכם ממשק ומימוש מלא (אין צורך לממש בעצמכם) של המחלקות Task ו-Person שבשימוש המחלקה, קראו את הממשק והתיעוד שלהן כדי להבין כיצד צריך להשתמש במחלקות אלו.

מסופק לכם הקובץ TaskManager.h עם ממשק המחלקה. ניתן להוסיף מתודות ושדות פרטיים כרצונכם. מנהל המשימות מחלק משימות לאנשים, לכל משימה יש מזהה (id) ייעודי עבורה, שמחולק למשימות בסדר כרונולוגי עולה המתחיל ב0 למשימה הראשונה, 1 לשנייה וכו'. מספר העובדים המקסימלי הוא 10.

את מימוש המודול יש לכתוב בקובץ בשם TaskManager.cpp, ניתן להוסיף שדות ומתודות פרטיות בלבד לקובץ TaskManager.h.

4.3.1 ממשק המחלקה

על המחלקה לתמוך בפעולות הבאות:

1. בניה של מנהל משימות ריק באמצעות בנאי חסר פרמטרים.
2. assignTask - מתודה המקבלת שם עובד ומשימה, ומכניסה את המשימה לרשימת המשימות של העובד. המשימה תקבל את ה-id הבא במערכת. אם שם העובד לא קיים במנהל המשימות, הוא יתווסף למערכת. אם מנסים להוסיף משימה לעובד שעוד לא קיים, והמערכת כבר מלאה באנשים, עליכם לזרוק חריגה מסוג std::runtime_error.
3. completeTask - מתודה המקבלת שם עובד ומסמנת שהעובד השלים את המשימה הנוכחית שלו. כלומר, המשימה בעדיפות העליונה עבור אותו עובד מוסרת מהרשימה. אם העובד לא בחברה, המתודה לא תעשה דבר (בפרט לא תזרוק חריגה).
4. bumpPriorityByType - מתודה המקבלת סוג של משימה (מוגדר בקובץ Task.h) ומספר. המתודה מעלה לכל המשימות מהסוג שהתקבל את התיעדוף שלהן בכמות שהתקבלה. אם המספר שלילי, המתודה לא תעשה דבר.

4.3.2 מתודות הדפסה

עליכם לממש מתודות המדפיסות את המשימות בכמה דרכים:

1. printAllEmployees - מדפיסה את כל אחד מהעובדים (ואת המשימות שמוקצות לכל אחד) לפי סדר יצירתם, באמצעות אופרטור הדפסה של Person. לאחר כל עובד תודפס ירידת שורה.
2. printAllTasks - מדפיסה את כל המשימות במערכת (של כל העובדים) לפי סדר החשיבות שלהן, אמצעות אופרטור ההדפסה של Task. לאחר כל משימה תודפס ירידת שורה. אם יש כמה משימות עם אותה חשיבות, המשימה הישנה יותר תודפס קודם.
3. printTasksByType - מקבלת סוג משימה כארגומנט. מדפיסה את כל המשימות במערכת (של כל העובדים) מהסוג שהתקבל לפי סדר החשיבות שלהן, באותו האופן כמו במתודה printAllTasks.

4.3.3 בדיקות מסופקות

בקובץ main.cpp תוכלו למצוא תכנית פשוטה הבודקת את המודולים SortedList ו-TaskManager. יש לקמפל את הבדיקות עם הקבצים שכתבתם באמצעות הפקודה:

```
g++ -DNDEBUG -std=c++17 -Wall -pedantic-errors -Werror -o TaskManager *.cpp
```

ניתן להריץ את כל הבדיקות על ידי הפקודה :

```
./TaskManager
```

ניתן להריץ בדיקה בודדת על ידי הוספת האינדקס שלה, למשל:

```
./TaskManager 1 > test1.out
```

בתיקיה tests, תוכלו למצוא דוגמאות פלט לכל הבדיקות המסופקות (6 בדיקות).
תוכלו לבדוק את פלט התכנית אל מול הפלט המצופה למשל באמצעות הפקודה הבאה:

```
diff --strip-trailing-cr -B -Z test1.out tests/test1.expected
```

5 הבהרות

- אין להשתמש ב STL או ב smart pointers.
- אין להוסיף קוד למחלקות Person ו Task.

6 הגשה ובדיקה

6.1 הגשה

עליכם להגיש את הקבצים הבאים:

- dry.pdf
- log.txt
- SortedList.h
- TaskManager.cpp
- TaskManager.h

את ההגשה יש לבצע במערכת ה-Gradescope.

שימו לב – על אחד השותפים (בלבד) להגיש את פתרון התרגיל במערכת. לאחר ההגשה יש להוסיף את השותף השני להגשה באמצעות לחיצה על כפתור "Group Members" ב-Gradescope.

הערות:

- ניתן להגיש מספר פעמים את התרגיל, ההגשה האחרונה (בלבד) היא זו שתיבדק ותקבל ציון.
- בכל הגשה נוספת שאתם מבצעים, יש להוסיף מחדש את השותף להגשה ב-Gradescope.
- וודאו שהקבצים אותם אתם מגישים (בפרט קבצי PDF) ניתנים לפתיחה ולצפייה.

6.1.1 הגשה באמצעות GitHub

בתרגיל זה, יש להגיש את הפתרון שלכם במערכת ה-Gradescope באמצעות GitHub. עליכם להגיש את ה-Repo שיצרתם בחלק היבש ואליו הוספתם את הפתרון לחלק הרטוב. לשם כך בדף הגשת התרגיל בחרו באפשרות ההגשה באמצעות GitHub, תנו ל-Gradescope הרשאות גישה לחשבון ה-GitHub שלכם ובחרו את ה-Repo אותו תרצו להגיש.

הקפידו על עבודה נכונה עם Git כפי שלמדנו בהרצאות ובתרגולים.

6.2 בדיקה אוטומטית

מיד עם ההגשה שלכם במערכת ה-Gradescope, יתבצעו מספר בדיקות אוטומטיות על קובץ ההגשה:

1. בדיקות שפיות – בודקות שכל הקבצים הדרושים נמצאים ב-zip המוגש ושהקוד שלכם מתקמפל כראוי. בדיקות אלו נועדו לוודא שפורמט ההגשה שלכם תקין.
2. טסטים אוטומטיים – הבדיקה האוטומטית מריצה את התוכניות שהגשתם עם הקלטים שסופקו לכם בקבצי התרגיל. הטסטים האוטומטיים בודקים שריצת התכנית הסתיימה בהצלחה ללא שגיאות זיכרון ונתנה פלט נכון.

בדיקות אלו נועדו עבורכם לוודא שההגשה שלכם תקינה והקוד עובד כראוי, קראו היטב את הפלט והמשוב מהבדיקות!

על מנת להשוות תכנים של קבצים בבדיקה האוטומטית, נשתמש בפקודת ה-diff הבאה (כאשר file1 ו-file2 הם הקבצים אותם נרצה להשוות:

```
diff --strip-trailing-cr -B -Z file1 file2
```

<u>שימו לב</u> – בנוסף לבדיקות שסופקו לכם, ההגשה הולכת לעבור בדיקות אוטומטיות נוספות לפיהן יקבע הציון על התרגיל. אנו מצפים שתבדקו בעצמכם את התרגיל שלכם מעבר לבדיקות המסופקות, האחריות על נכונות ההגשה היא שלכם בלבד.

6.3 תחרות מימז

בתרגיל זה מתקיימת תחרות מימז (Memes).
כדי להשתתף בתחרות, עליכם להוסיף מימ לתחילת הקובץ dry.pdf. הזוג שיזכה בתחרות יקבל בונוס של 5 נקודות לציון התרגיל (לכל אחד מהשותפים).

חוקי התחרות:

1. מותר לכם להגיש לתחרות מימ אחד <u>בלבד</u>.
2. נושא המימ חייב להיות קשור לחומר הקורס.
3. המימ הזוכה יפורסם בתרגול לאחר לפרסום הציונים.

בהצלחה !
بالنجاح !
GOOD LUCK!