הספריה התקנית - איטרטורים ואלגוריתמים

איטרטורים

הספריה התקנית מגדירה כמה סוגים של איטרטורים.

- איטרטור טריביאלי - משמש לקריאה בלבד, אי אפשר להזיז אותו.
- איטרטור קלט - משמש לקריאה בלבד, אפשר להזיז אותו קדימה (++).
- איטרטור פלט - משמש לכתיבה בלבד, אפשר להזיז אותו קדימה (++).
- איטרטור קדימה (forward iterator) - שילוב של איטרטור קלט ופלט, יכול לשמש לקריאה ולכתיבה.
- איטרטור דו-כיווני (bidirectional iterator) - כמו הקודם, רק שהוא יכול גם לזוז אחורה (--).
- איטרטור גישה אקראית (random access iterator) - כמו הקודם, רק שאפשר גם לבצע עליו אריתמטיקה כמו עם פוינטרים של סי.

המיכלים השונים מציעים איטרטורים ברמות שונות, למשל:

- **זרמים** מציעים איטרטורי קלט, פלט ואיטרטור "קדימה".
- **קבוצה, מפה ורשימה** מציעות איטרטור דו-כיווני.
- **וקטור** מציע איטרטור גישה אקראית.

לכל מיכל יש שיטה begin המחזירה איטרטור לתחילת המיכל ושיטה end המחזירה איטרטור ל**אחרי** סוף המיכל.

למיכלים המאפשרים הליכה אחורה (כמעט כולם, חוץ מזרמים ו-forward_list) יש גם שיטה rbegin המחזירה איטרטור לסוף המיכל ושיטה rend המחזירה איטרטור ל**לפני** תחילת המיכל (עבור איטרציה בסדר הפוך).

לכל איטרטור יש גם גירסה שהיא const - גירסה המאפשרת לקרוא את הפריטים במיכל אבל לא לשנות אותם. אפשר לגשת אליה ע"י cbegin, cend, crbegin, crend.

כשרוצים להכניס פרט באמצע מיכל כלשהו (לא בסוף או בהתחלה), משתמשים בדרך-כלל באיטרטור שאומר איפה בדיוק להכניס.

- כדי להכניס פריט לפני המקום שהאיטרטור i מצביע עליו - **c.insert(i,x)**
- כדי להכניס פריטים בקטע החצי-פתוח מ-first ל-- last
  c.insert(i,first,last)
- אותו הדבר עם מחיקה - **erase**.

- אותו הדבר עם הכנסה במקום - **emplace**.

הפונקציות **insert, erase, emplace** עובדות בצורה דומה בכל המיכלים התומכים בהם (ראו טבלה ב **http://www.cplusplus.com/reference/stl**), אלא שהן שומרות על השמורה של המיכל. כך למשל, אם מנסים להכניס איבר למיכל מסודר, והאיטרטור שנותנים ל-**insert** מצביע למקום הלא נכון מבחינת הסדר - האיטרטור הזה ישמש רק כרמז (**hint**), החיפוש יתחיל משם אבל בסופו של דבר הפריט יוכנס למקום הנכון לפי הסדר.

תקינות איטרטורים

כשעובדים עם איטרטורים, חשוב לוודא שהם **תקינים**. מתי איטרטור עלול להיות לא-תקין? למשל, כשתוך כדי לולאה, אנחנו מוחקים את האיבר שהאיטרטור מצביע עליו:

- ברשימה, מפה וקבוצה - האיטרטור מכיל פוינטר המצביע למקום לא מאותחל בזיכרון - שגיאה חמורה.
- בוקטור - האיטרטור מצביע לאיבר הבא אחרי האיבר שמחקנו - לא שגיאה כל-כך חמורה, אבל עדיין לא מה שרצינו.

אז מה עושים? החל מ- C++11, השיטה erase מחזירה איטרטור מעודכן ותקין לאחרי המחיקה. צריך פשוט לשים את האיטרטור הזה באיטרטור שלנו. ראו דוגמת קוד במצגת ו כאן:
https://stackoverflow.com/q/2874441/827927

איטרטורים על מפה

כשמשתמשים באיטרטור i על מפה (map), הסוג של *i הוא זוג (pair) של מפתח+ערך.

איטרטורים על מיכלים אסוציאטיביים

למיכלים אסוציאטיביים, כמו מפה או קבוצה, יש שיטות מיוחדות שמחזירות איטרטורים:

- find - מקבל מפתח, מחזיר איטרטור לערך המתאים או end() אם הערך לא נמצא.
- lower_bound - מקבל מפתח, מחזיר איטרטור לערך הכי קטן שהוא שווה או גדול מהמפתח.
- upper_bound - מקבל מפתח, מחזיר איטרטור לערך הכי קטן שהוא גדול ממש מהמפתח.

ראו תיקיה 5.

מתאמים

מתאם (adaptor) הוא דגם-עיצוב שנועד להתאים מחלקה נתונה לממשק רצוי. בספריה התקנית יש כמה מתאמים, למשל:

- stack - מתאם ההופך כל מיכל סדרתי למחסנית ע"י הוספת שיטות push, pop (השיטה pop שולפת את האיבר הכי חדש בתור).
- queue - מתאם ההופך כל מיכל סדרתי לתור חד-כיווני ע"י הוספת שיטות push, pop (השיטה pop שולפת את האיבר הכי ישן בתור).

אפשר לבנות מחסנית/תור על-בסיס וקטור, deque, רשימה מקושרת, או כל מיכל סדרתי אחר.

אלגוריתמים

האלגוריתמים בספריה התקנית מקבלים כקלט **זוג איטרטורים** ולא מיכל (זאת בניגוד לג'אבה. בג'אבה יש אלגוריתם סידור נפרד עבור List, עבור מערך של תוים, מערך של מספרים וכו'..). דוגמאות לאלגוריתמים (ראו בתיעוד הספריה):

- סידור - sort
- מיזוג מערכים מסודרים - merge
- העתקה - copy

האלגוריתמים האלה עובדים על איטרטורים, ולכן אפשר להשתמש בהם על כל מיכל התומך באיטרטורים מהסוג המתאים.

לדוגמה, האלגוריתם sort עובד על איטרטורים מסוג RandomAccess. לכן אפשר להשתמש בו לסידור וקטור וגם מערך פרימיטיבי.

אבל, אי אפשר להשתמש באלגוריתם sort לסידור list, כי האיטרטור שלה הוא מסוג Bidirectional (לא תומך למשל בפעולת חיסור).

מה עושים? משתמשים בשיטת sort המיוחדת של list; ראו תיקיה 6.

הודעות שגיאה

אחד הקשיים העיקריים בעבודה עם STL הוא הודעות השגיאה. למשל, אם ננסה להריץ את אלגוריתם sort על list, לא נקבל הודעה פשוטה שאומרת "אי אפשר להריץ sort על list", אלא הודעה ארוכה ומסובכת הנכנסת לפרטי התבניות בספריה התקנית (ראו דוגמה בתיקיה 6).

כדי לפענח את הודעת השגיאה, צריך לחפש את ה-note המפנה לשורה בקוד שלנו, ומשם לנסות להבין מה הבעיה.

ישנן ספריות המנסות לתת הודעות שגיאה משמעותיות יותר, למשל boost, STLFilt - לא ניכנס לזה בקורס הנוכחי.

מחרוזות

מחרוזת ממומשת כמיכל של תוים (char) בתוספת פונקציות שימושיות למחרוזות, כמו חיפוש תת-מחרוזת, שירשור ועוד.

כדי להפוך ערך כלשהו למחרוזת, אפשר להשתמש בשיטה הגלובלית to_string, או ב-ostringstream, או בסיומת s, למשל "abc" עם סיומת s מציין אובייקט string המכיל "abc".

ספריות נוספות

בנוסף לספריה התקנית, יש ספריות נוספות. המקובלת ביותר היא boost היא "כמעט" תקנית - הרבה מהדברים בספריה התקנית התחילו את דרכם ב-boost, השתכללו והשתפרו, עד שבסוף נכנסו לספריה התקנית. לכן, אם חסר לכם משהו בספריה התקנית - נסו לחפש ב-boost.

מקורות

- מצגות של אופיר פלא.
- Peter Gottschling, "Discovering Modern C++", chapter 4.
- תיעוד הספריה התקנית: http://www.cplusplus.com/reference/stl.
- השוואת ביצועים בין deque לבין וקטור: https://www.codeproject.com/Articles/5425/An-In-Depth-Study-of-the-STL-Deque-Container
- מחיקה תוך כדי ריצה https://stackoverflow.com/q/2874441/827927

סיכם: אראל סגל-הלוי.