הספריה התקנית – אלגוריתמים

בספריה התקנית של C++ יש 105 אלגוריתמים, נכון לשנת 2017. אנחנו צריכים להכיר את כולם.

למה? - כי האלגוריתמים פותרים בעיות מאד נפוצות בתיכנות. אם לא נכיר את כולם, אנחנו עלולים לנסות לממש אותם בעצמנו תוך כדי פרוייקט אחר. ואז, אנחנו כנראה נממש אותם בחיפזון, בלי בדיקות, בצורה לא יעילה ועם באגים. בנוסף, האלגוריתמים נכנסו לתקן של השפה, ולכן מתכנתים בכל העולם משתמשים בהם ומבינים אותם, ומצפים שגם מתכנתים חדשים יכירו אותם וישתמשו באותה שפה.

איך אפשר לזכור 105 אלגוריתמים? - זה לא קל, אבל אנחנו נחלק אותם לקבוצות הגיוניות (ע"פ המצגת המעולה של יונתן בוקארה – ראו קישור למטה).

לפני שנתחיל, נדגיש שהאלגוריתמים בספריה התקנית מקבלים כקלט **זוג איטרטורים** ולא מיכל. זאת בניגוד לשפות אחרות כגון ג'אבה. בג'אבה יש אלגוריתם סידור נפרד עבור List, עבור מערך של תוים, מערך של מספרים וכו'...; בספריה התקנית יש אלגוריתם **אחד** לסידור טווח, והטווח נתון ע"י שני איטרטורים – התחלה (begin) ואחרי-הסוף (end). אותו אלגוריתם יכול לסדר גם וקטור, גם מערך על המחסנית, גם deque, וגם כל מבנה אחר שיש לו איטרטור-התחלה ואיטרטור-סוף.

שאילתות – queries

בקבוצת השאילתות נמצאים אלגוריתמים רבים המקבלים כקלט טווח (איטרטור התחלה ואיטרטור סוף), ומחזירים ערך כלשהו על הפרטים הנמצאים בטווח. למשל, האלגוריתם count מחזיר את מספר המופעים של פרט מסויים בטווח, האלגוריתם accumulate מחזיר את סכום הפרטים בטווח, וכו'. ראו תיקיה 10.

אלגוריתמים על קבוצות – set algorithms

בקבוצה זו נמצאים אלגוריתמים המקבלים כקלט שתי קבוצות ומחזירים קבוצה שלישית. כל אחת מקבוצות-הקלט מועברת ע"י שני איטרטורים – התחלה וסוף. תנאי מקדים לאלגוריתמים האלה הוא, שהפרטים בכל אחת מקבוצות-הקלט מסודרים לפי אופרטור "קטן מ-" (<) או לפי פונקטור אחר המועבר כקלט. תנאי זה מתקיים עבור std::set, אבל גם עבור וקטור שסידרנו אותו בעזרת sort (ראו למטה). הפלט מיוצר בעזרת איטרטור נוסף – איטרטור-פלט. למשל, האלגוריתם set_union מקבל חמישה איטרטורים – התחלה וסוף של קבוצה א, התחלה וסוף של קבוצה ב, ואיטרטור-פלט עבור התוצאה. האלגוריתם מחשב את האיחוד של קבוצה א וקבוצה ב (אוסף הפרטים הנמצאים לפחות באחת משתי הקבוצות, ללא כפילויות), ומכניס את התוצאה לאיטרטור-הפלט. ראו תיקיה 11.

פרמוטציות – permutations

בקבוצת הפרמוטציות נמצאים אלגוריתמים המקבלים כקלט טווח (איטרטור התחלה ואיטרטור סוף), ומשנים את סדר הפרטים בסדרה הנמצאת בין האיטרטורים, אך אינם משנים את הפרטים עצמם. למשל, האלגוריתם sort מסדר את הסדרה לפי האופרטור "קטן מ-"', או לפי פונקטור אחר כלשהו המועבר כפרמטר לאלגוריתם. ראו תיקיה 12.

מעבירים - movers

בקבוצה זו נמצאים אלגוריתמים המעתיקים או מעבירים פרטים בין טווח אחד לטווח אחר. הפשוט ביותר ביניהם הוא copy: הוא מקבל טווח קלט (ע"י שני איטרטורים), ואיטרטור פלט, ומעתיק את כל הפרטים מטווח הקלט לאיטרטור הפלט. ראו תיקיה 13.

משני-ערך – value modifiers

בקבוצה זו נמצאים אלגוריתמים המשנים את הערכים בטווח נתון. הפשוט ביותר הוא fill – הוא מקבל טווח (= איטרטור התחלה ואיטרטור סוף), וממלא אותו בערך נתון. ראו תיקיה 14.

מיליות – "Runes"

לרוב האלגוריתמים שראינו עד כה יש כמה גירסאות, שאפשר ליצור ע"י הוספות "מיליות" לשם האלגוריתם. לדוגמה, המילית is מציינת שאילתה: is_sorted = האם הטווח מסודר, is_sorted_until = עד איפה בדיוק הטווח מסודר, וכו'. ראו תיקיה 15.

משני-מבנה – structure changers

בקבוצה זו נמצאים שני אלגוריתמים שמטרתם למחוק פרטים מתוך טווחים. למשל remove מקבל טווח (= שני איטרטורים) וערך, ו"מוחק" את כל הפרטים בטווח השווים לערך הנתון. למה "מוחק" בגרשיים? כי האלגוריתם לא באמת יכול למחוק אותם – הוא הרי לא מקבל רפרנס לאוסף, אלא רק שני איטרטורים! אז מה הוא עושה? - הוא מעביר כל הפרטים שאמורים להישאר בטווח (= שאינם שווים לערך הנמחק) לתחילת הטווח, ואז מחזיר איטרטור (נניח iter) לאחרי-הסוף של הפרטים הנשארים. לאחר מכן, אפשר להשתמש בשיטה אחרת של האוסף (למשל vector::erase) כדי לבצע את המחיקה עצמה. ראו תיקיה 16.

אלגוריתמים נוספים

שני האלגוריתמים האחרונים הם transform – ביצוע טרנספורמציה כלשהי על פרטים בטווח (נתון ע"י שני איטרטורים) ושפיכת התוצאה לטווח אחר (הנתון ע"י איטרטור פלט), וfor_each – ביצוע פעולה כלשהי על כל פרט בטווח (נתון ע"י שני איטרטורים). ראו תיקיה 17.

סוגי איטרטורים

כל אלגוריתם דורש איטרטור ברמה מסויימת (ראו טקסט קודם על איטרטורים).

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.

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

- Jonathan Boccara, "105 algorithms in less than an hour", CPPCON 2018, https://youtu.be/2olsGf6JIkU
- Sean Parent, "C++ Seasoning", GoingNative 2013, https://www.youtube.com/watch?v=qH6sSOr-yk8,
- Jonathan Boccara, "Is for_each obsolete? (no)" https://www.fluentcpp.com/2018/03/30/is-stdfor_each-obsolete/
- מצגות של אופיר פלא.
- Peter Gottschling, "Discovering Modern C++", chapter 4.
- Marius Bancila, "Modern C++ Programming Cookbook", chapter 5
- תיעוד הספריה התקנית: http://www.cplusplus.com/reference/stl.

סיכם: אראל סגל-הלוי.