העמסת אופרטורים

העמסה (overloading) היא מצב של כמה פונקציות עם אותו שם וארגומנטים מסוגים שונים.

בשפת ++C, גם אופרטור הוא פונקציה, ולכן אפשר **להעמיס אופרטורים**: להגדיר אופרטורים המבצעים פעולות שונות לפי סוג הארגומנטים המועברים אליהם. לדוגמה:

- **אופרטורים חשבוניים** (חיבור, חיסור, כפל, השוואה, וכו') מוגדרים על מספרים שלמים וממשיים; אנחנו יכולים להעמיס אותם גם במחלקות שאנחנו בונים, המייצגים עצמים מתמטיים מורכבים יותר. למשל: מספרים מרוכבים (ראו תיקיה 1), מטריצות, פולינומים וכו'. בספריה התקנית הם הורחבו גם למחרוזות.
- **אופרטורי זרימה** (<< >>) - מוגדרים במקור (בשפת סי) על מספרים שלמים, אבל בשפת ++C העמיסו אותם לזרמי קלט ופלט כפי שכבר ראינו. גם אנחנו יכולים להרחיב אותם כדי לכתוב ולקרוא מחלקות שאנחנו בונים (ראו תיקיה 1).
- **אופרטור סוגריים מרובעים []** - מוגדר לגבי מערכים בסיסיים; אפשר להרחיב אותו גם למחלקות שאנחנו בונים, כשאנחנו רוצים לגשת לדברים לפי אינדקס (ראו תיקיה 2). בספריה התקנית הם הורחבו למבנים כגון vector וגם map (ראו תיקיה 3).
- **אופרטור סוגריים עגולים ()** - יכול לשמש להגדרת אובייקטים המתפקדים כמו פונקציות - "פונקטורים" (functors). ראו תיקיה 4.

כשמעמיסים אופרטורים, חשוב לשים לב שהערך המוחזר תואם למשמעות של האופרטור. למשל:

- אופרטור + מחזיר את הסכום; אופרטור += מגדיל את הארגומנט השמאלי שלו אבל **גם** מחזיר את הסכום (שהוא הארגומנט השמאלי אחרי ההשמה).
- אופרטור = (השמה) מחזיר *this - זה מאפשר לבצע השמות בשרשרת.
- האופרטורים >> << מחזירים את זרם הקלט/פלט שהם מקבלים - שוב כדי לאפשר שרשרת.
- לאופרטור הגדלה באחד (++) והקטנה באחד (--) יש שתי גירסאות: כששמים אותו לפני המספר (prefix), הוא מחזיר את המספר אחרי ההגדלה; כששמים אותו אחרי המספר (postfix), הוא מחזיר את המספר לפני ההגדלה. כדי להבדיל בין האפשרויות, יש להגדיר את הגירסה השניה עם פרמטר מדומה מסוג int, למשל:
  - `T& T::operator++();  // prefix operator`
  - `T& T::operator++(int);  // postfix operator`

מקורות

- מצגת של אופיר פלא.

סיכם: אראל סגל-הלוי.