העמסה בשפת C++

העמסת פונקציות

העמסה (overloading) היא מצב שבו יש כמה פונקציות עם אותו שם וארגומנטים שונים. הבחירה לאיזה פונקציה לקרוא מתבצעת ע"י הקומפיילר. למה זה חשוב? כי זה מאפשר לנו לקרוא לפונקציות בשמות קצרים וברורים, ולסמוך על הקומפיילר שיבחר את הפונקציה הנכונה לפי הצורך. הרבה מהתכונות המתקדמות של שפת C++ מסתמכות על העמסה, ולכן כדאי להכיר היטב את המנגנון.

איך הקומפיילר יודע איזה פונקציה לבחור? הוא עובד בכמה שלבים:
1. מציאת כל הפונקציות עם השם המתאים.
2. מתוך קבוצה 1, מציאת כל הפונקציות עם מספר הפרמטרים המתאים.
3. מתוך קבוצה 2, מציאת הפונקציות עם סוג הפרמטרים המתאים ביותר.

שלב 3 הוא הכי מסובך, כי הקריטריון להתאמה בסוג הפרמטרים לא תמיד ברור. לדוגמה ראו בתיקיה 5. הגדרנו שם פונקציה בשם power המחשבת חזקה של שני מספרים. כשהחזקה היא מספר שלם חיובי אפשר להשתמש ברקורסיה:

```
int power(int a, unsigned int b) {
      cout << " power of ints" << endl;
      return b==0? 1: a*power(a,b-1);
}
```

אבל כשהחזקה היא מספר ממשי צריך להשתמש בלוג:

```
double power(double a, double b) {
      cout << " power of reals" << endl;
      return exp(b*log(a));
}
```

הקריאה power(2,3) מגיעה לפונקציה הראשונה והקריאה power(2.0,3.5) מגיעה לפונקציה השניה. לאן תגיע הקריאה power(2,3.5)? היינו מצפים שהיא תגיע לפונקציה השניה, אבל אצלי זו שגיאת קומפילציה - הקומפיילר לא בטוח לאיזו פונקציה התכוונו, רמת ההתאמה היא זהה כי בשני המקרים צריך לבצע המרה (להמיר מספר שלם לממשי או להיפך).

לאן תגיע הקריאה power(2,-3)? היינו מצפים שהיא תגיע לפונקציה השניה כי החזקה שלילית, אבל אצלי היא מגיעה לפונקציה הראשונה ונכנסת לרקורסיה ארוכה מאד - המספר מינוס3 מומר למספר חיובי גדול - ממש לא מה שהתכוונו לעשות.

המסקנה: צריך מאד להיזהר בהעמסת פונקציות, במיוחד כשהן עם אותו מספר פרמטרים ועם פרמטרים מספריים.
ראו דוגמה בתיקיה 7.

מקורות

- מצגות של אופיר פלא ומירי בן-ניסן.

סיכם: אראל סגל-הלוי.