קווים מנחים: עקרונות תכנות מונחה עצמים ב C++

יחיאל קמחי

C מול C++: מנקודת מבט כרונולוגית, C++ התפתחה כהרחבה של C, אך הסיבה הישירה לכך היתה כדי להקל על מימוש (כתיבת מהדר עבור) C++. מאז התפתחה C++ ללא הכר ומעובדה היסטורית זו נותרו רק שני שרידים: יש מהדרים מאוחדים ל C++ ול C, והסטנדרד של C++ מכיל בתוכו חלק נכבד של שפת C (אך לא את השפה במלואה). בנוסף, נותר גם בלבול אצל מתכנתים רבים, הרואים, עדיין, את C++ כהרחבה של C בכוון OOP. הבה נומר במפורש: C++ איננה הרחבה של C ואיננה דומה ל- C, פרט לדמיון חיצוני בפעולות הבסיסיות. כמו-כן, המחוייבות להיות "תואמת C ככל-האפשר" מהווה נטל על C++ ובשום פנים אינה יתרון. למעשה, יודעי C צריכים "לשכוח" C כדי להצליח להבין את "הראש" של C++: עובדה – כל תוכנית C, המומרת ל C++ חייבת להשתנות.
דוגמה: ב C++ הטיפוסיות חזקה. השימוש בהמרות (casting) בדרך כלל מוטעה ומיותר. למקרים המועטים בהם ההמרה המפורשת הכרחית – יש פעולות המרה בטוחות.

C++ כשפה לתכנות-מונחה-עצמים: C++ איננה שפה טהורה לתכנות-מונחה-עצמים, אך היא מאפשרת שימוש יעיל בכל הטכניקות של גישה זו. הבסיס לתכנות-מונחה-עצמים הוא הגדרה מדויקת של סוגי-העצמים הנדרשים בתוכנה, כולל זיהוי תכונותיהם, התנהגויותיהם והיחסים ביניהם (בתוך אותו הסוג ובין סוגים שונים). המחלקה (class) היא המנגנון המתאר (קבוצה של) עצמים בעלי תכונות זהות. מנגנון ההורשה הגלויה (public inheritance) מתאר קשר בין מחלקות, שבו העצמים במחלקה היורשת מהווים מקרה-מיוחד של העצמים במחלקה הנורשת (כיוון שיש להם תכונות והתנהגויות נוספות על המקוריות), והם מהווים תת-קבוצה בין העצמים במחלקת-הבסיס.
מנגנון התבניות (templates) מתאר קשר אחר בין מחלקות, כאשר המחלקות השונות מגדירות התנהגויות זהות – אך ביחס לעצמים מסוגים שונים. מנגנון התבניות מאפשר למנוע שכפול קוד ע"י המתכנת, מבלי לפגוע בטיפוסיות החזקה של השפה.

C++ כשפה לתכנות בטוח: בעת התכנות, C++ מאפשרת הפרדה מפורשת בין התנהגות (חיצונית) לבין המימוש (הפנימי) המאפשר התנהגות זו. כל ויתור קטן על הפרדה זו, פוגם בתוכנה ומחירו גבוה.
C++ מספקת מנגנונים רבים שתפקידם לאפשר למתכנת ליצור קוד בטוח (יחסית) לשימוש.

כתיבת קוד ב-"רוח C++" תאפשר גילוי שגיאות רבות בזמן ההידור:

1. הטיפוסיות החזקה שומרת מפני שימוש בפעולות עם עצמים שאינם מתאימים עבורן.
2. הקפדה על קבועים (const-correctness) מגינה על עצמים מפני שינוי בלתי-צפוי.
3. יוצרים (constructors) מגדירים במדויק כיצד, אם בכלל, אפשר ליצור עצמים, וההורס (destructor) חוסך למתכנת השקעת מאמצי-נקיון, שהמהדר יכול לבצע בעצמו.

סיכום: C++ אינה שפה קלה, אך היא מאפשרת למתכנת גמישות רבה, שאיננה קיימת ברוב השפות האחרות. בין השאר, המתכנת יכול לבחור במימוש יעיל של אלגוריתם נתון בלי לשלם הרבה בוויתור על קריאות התוכנית (התשלום מתבטא ביצירתיות המימוש).
קל מאד לכתוב תכנית C++ בזבזנית במשאבים, אך באותה המידה קל לכתוב תכנית יעילה למדי (לרוב, יעילה מתוכנית מקבילה ב C) – כל הנדרש הוא הכרה של המנגנונים הבסיסיים בשפה ומשמעויותיהם. אין צורך להכיר בעל-פה את כל האופנים השונים לשימוש במנגנוני-השפה – מספיק להכיר את המנגנונים העיקריים בשפה ואת משמעויותיהם, ולחפש בספרים טובים את הפרטים הטכניים בשעת הצורך.