קווים מנחים לתכנות נכון

לפניך אוסף של קווים מנחים אשר יעזרו לך לכתוב קוד נכון, קל להבנה וקל לתחזוקה.

שים לב, כשמם כן הם – קווים מנחים, אך לא חוקים. תמיד חשוב על הקוד שאתה כותב והבן מה אתה עושה. כך תבטיח שימוש נכון בכללים וגם תדע מתי לא להשתמש בהם.

העקרונות שלפניכם נלקחו מתוך הספר Thinking in C++. ניתן למצוא אותו מופץ בחינם באינטרנט. הוא מעט מיושן (נכתב בתקופה שמערכות הפעלה תמכו ב- 16ביט) אולם מומלץ לקריאה.

1. כתבו קוד שעובד. אחר כך הפכו אותו למהיר יותר
2. אלגנטיות תמיד משתלמת
3. הפרד ומשול
4. הפרד בין יוצר המחלקה ובין משתמש המחלקה
5. בחרו שמות משמעותיים וברורים עד כדי כך שלא יהיה צורך בהערות
6. Access control – (private, protected, public) מאפשר לשנות כמה שרצה בעתיד בלי לשנות את המחלקה עצמה
7. ניתוח ועיצוב (design) צריך לייצר לפחות את המחלקות, את הממשק הציבורי שלהן ואת היחסים בין המחלקות
8. כתבו את קוד הבדיקות קודם
   כתובו את קוד הבדיקות קודם כדי לאשר (או להפריך) את התכנון שלכם
   אם אתם לא יכולים לכתוב בדיקות, אתם לא יודעים איך המחלקה צריכה להראות
9. את כל הבעיות אפשר לפתור ע"י רמת הפשטה נוספת
   מחלקה צריכות להיות אטומיות – Single responsibility Principle
10. שימו לב לגודל של פונקציות (ומחלקות). אם פונקציה גדולה מידי, כנראה שהיא עושה יותר ממה שהיא צריכה וצריך לפצל אותה
11. שימו לב לרשימת ארגומנטים גדולה מידי
12. על תחזור על עצמך. Don’t repeat yourself (DRY)
13. שימו לב ל- switch או הרבה if-else. לפעמים זה סימן לבדיקה של טיפוס (type-check coding). אפשר להחליף בפולימורפיזם
14. מבחינת עיצוב (design) הפרד בין הדברים שמשתנים והדברים שלא משתנים
15. שימו לב ליצירת הגבלות בירושה (עקרון הפרדת הממשקים ועקרון ההחלפה של ליסקוב)
16. אל תוסיפו פונקציונליות חשובה דרך ירושה. עשו זאת במחלקת האב
17. פחות זה יותר – Less is more
18. קראו בקול את שמות המחלקה. שימו לב ליחסים Is-a ו- has-a
19. תמיד תעדיפו הכלה על פני ירושה. תמיד שאלו אם יש צורך ל- upcast למחלקת האב.
20. על תבצעו אופטימיזציה בשלב מוקדם מידי (premature optimization)
21. לעולם על תשתמשו בגלובליים, שימו הכל במחלקות
22. התייחסו לאזהרות של הקומפיילר כשגיאות
23. בחרו קונבנציה ודבקו בה