

## חלק יבש

א. לא נוכל להגדיר את פעולות ה-`iterator` כ-`const` מאחר וחלק מהפעולות על ה-`iterator` משנות אותו וכאשר מגדירים פעולה כ-`const`, הכוונה היא לכך שאובייקט ה-`this`, במקרה שלנו ה-`iterator` לא יוכל להשתנות. לדוגמה, האופרטור ++ מקדם את ה-`iterator` להצביע על האובייקט הבא וכך משנה אותו.

ב. 1) `copy constructor` – המתודה משתמשת במתודה `pushBack`, בה יש שימוש ב-`copy constructor` של המחלקה `node`, ובה מניחים שלטיפוס הטמפלייטי יש `copy constructor`.

2) `operator=` - באופרטור ההשמה, יש שימוש במתודה `pushBack` בה משתמשים בבנאי העתקה של `node`, ובה מניחים שלטיפוס הטמפלייטי יש בנאי העתקה.

3) `pushBack` – במתודה יש שימוש בבנאי העתקה של `node`, ובה מניחים שלטיפוס הטמפלייטי יש בנאי העתקה.

4) `filter` – בפונקציה יש שימוש במתודה `pushBack`, בה כמו בסעיפים הקודמים יש שימוש בבנאי העתקה ולכן מניחים שלטיפוס הטמפלייטי יש בנאי העתקה. בנוסף, הפונקציה מקבלת טיפוס טמפלייטי נוסף בו אנחנו מניחים שיש אופרטור סוגריים.

5) `transform` – הפונקציה מקבלת טיפוס טמפלייטי נוסף עליו מניחים, כמו בפונקציה `filter`, שיש לו אופרטור סוגריים. בנוסף, אנו מניחים שלטיפוס הטמפלייטי יש הורס.

ג. בזמן הקומפילציה, הקומפיילר יוצר מחלקה חדשה, אותה הוא מעתיק מקובץ ה-`h`, ומשנה את הטיפוס הטמפלייטי לפי הטיפוס המשומש כרגע. למשל, בעת שימוש בתור עם `<int>`, במקום `<T>`, תהיה מחלקה חדשה תור עם הטיפוס `<int>`. אם נממש את הפונקציות / מתודות בקובץ `cpp` נפרד, מאחר והקומפיילר יוצר מחלקה חדשה, לא תהיה גישה לאימפלמנטציה של המתודות / הפונקציות בקובץ `cpp` וזה ייצור בעיות קומפילציה. הבעיה נוצרת בשלב הלינקינג, כאשר המהדר מחפש את האימפלמנטציות של המתודות.

ד. ניתן ליצור functor בעל member שהוא `int`. בעת יצירת האובייקט, מתקבל בבנאי מספר (שלא בהכרח ידוע בעת הקומפילציה, למשל כקלט מהמשתמש), ואז ה-functor נשלח כארגומנט לפונקציה `filter`. בתוך ה-functor, יש העמסת אופרטור סוגריים, שמחזיר ערך בוליאני, אמת אם המספר המתקבל כארגומנט באופרטור סוגריים מתחלק ב-`int` שהוא member של ה-functor, ושקר, אחרת.