**שיטת העבודה המקובלת למימוש מחלקה**

**המסורת היא:**

* **לממש הכרזה של המחלקה בקובץ header בדרך כלל עם סיומת .h או .hpp**
* **את קוד המימוש בקובץ .cpp**
* **את קוד הלקוח בקובץ .cpp נפרד.**
* **מופעים של מחלקות יכולים (ובדרך כלל) להיות מוקצים על המחסנית ומשתחררים אוטומטית שם. אבל שטחים דינמיים לא משתחררים אוטומטית.**
* **למתכנת קיימת אפשרות להגדיר destructor הנקרא אוטומטית עם שחרור מופע מחלקה. תפקידו העיקרי לשחרר שטחים דינימיים.**
* **אחד הבעיות הכרוניות היא שכישלון של שחרור שטח דינמי אינו מדווח.**

**מלכודת מספר 1: אם מקצים מערך ע"י**

**Ptr = new type[n];**

**אז צריך לשחרר ע"י**

**delete [] ptr;**

**­**

**מתודות עזר (מקביל למתודות סטטיות ב-Java, C#)**

* **אפשר גם פה לממש מתודות סטטיות**
* **היסטורית בדרך כלל עשו זאת עם מה שנקרא "פונקציות גלובליות".**
* **בניגוד למתודות סטטיות, לפונקציות גלובליות אין זכאות לגשת לשדות המוגנים אלא אם כן הן מוכרזות כ-friend במחלקה.**
* **Friend יכולה להיות מתודה של מחלקה ומחלקה אחרת כולה.**
* **השלכות של הכרזת friend היא הצרה ביותר האפשרית.**

**התנגשויות שמות:**

**אפשר לפתור ע"י this-> או אופרטור ה-::.**

**ב-C++ מילת המפתח this הוא פוינטר.**

**העמסת אופרטורים ב-C++**

* **כל העמסת אופרטורים באה תחת הכותרת הזו.**
* **רוב האופרטורים ניתנים להעמסה הן כמתודות והן כפונקציות גלובליות.**
* **אם האופרטור מועמס כמתודה, זה קובע את אחד אופרנדים (השמאלי).**
* **אופרטור ההשמה ב-C++ פירושו "להעתיק בית בית". ב-C++ ניתן (ולפעמים אף חייבים) להעמיס את המקרה של השמת מופע של המחלקה. ב-C# זה בלתי אפשרי.**

**מלכודת מספר 2:**

**כאשר למחלקה יש מצביעים לשטחים דינמיים, המתכנת חייב לממש (משיקולים של הנדסת תוכנה):**

* **פונקציה הורסת**
* **בנאי העתקה**
* **העמסת אופרטור ההצבה "=".**

**אם זה לא נעשה קיימת אפשרות ש-destructor ישחרר שטח של מופע (גלובלי) שלא אמור להשתחרר, בגלל שחרור אוטומטי של משתנה זמני:**

**התסריט הזה יכול להתרחש ב-3 הקשרים:**

* **העברת פרמטר by value**
* **השמה**
* **החזרת תוצאת פונקציה.**

**הבעתיות נובעת מכך שבשפת C++ השמה של מופע אחד מחלקה בשנייה ממומשת ע"י העתקה בית בית (shallow copy).**

**בנאי העתקה**

**בנאי העתקה מוגדר כבנאי המקבל כפרמטר מופע של אותה מחלקה by reference ו-const.**

**class A**

**{**

**A(const A& a);**

**}**

**במידה והמתכנת אינו דורס את בנאי ההעתקה, הקומפיילר מעמיד לרשותו בנאי העתקה ברירת מחדל, המעתיק בית בית (בסדר גמור אם אין מצביעים לשטחים דינמיים).**

**בנאי העתקה נקרא:**

* **ישירות ע"י המתכנת (נדיר)**
* **נקרא אוטומטית בהעברת פרמטר by value**
* **נקרא אוטומטית בהחזרת ערך פונקציה.**

**אופרטור ההשמה**

* **השפה מעמידה לרשותנו אופרטור השמה ברירת מחדל המעתיקה בית בית.**
* **אופרטור ההשמה מתנהג בחלקו השני כמו בנאי העתקה, אבל בחלקו הראשון צריך להביא בחשבון שהמשתנה היעד יתכן שהצביע על משהו דינמי (שעלינו לשחרר במסגרת ההשמה).**

**הורשה ב-C++**

**הבדל לעומת השפות הקודמות:**

**מושג בשם "סוג הורשה".**

**סוג ההורשה קובע את מעמד השדות של מחלקה האב בקוד הלקוח של מחלקת הבן.**

**הכרוזות הורשה נראות ככה:**

**class b: a**

**class b: public a**

**class b: protected a**

**class b: private a**

**מציין רמת הנגישות קובעת את סוג ההורשה, שהיא רמת הנגישות המירבית של שדות האב בקוד הלקוח של הבן.**

**במושגים של C++, ההורשה ב-Java וב-C# תמיד public.**

**כל סוג הורשה שאיננו public מגביל מאד את הפולימורפיזם ולכן כמעט כל הורשה היא public.**