3. lekce

Kopírování objektů

Při kopírování takových objektů, které mají **všechny atributy staticky definované**, můžeme používat kopírující konstruktor, který vytvoří kompilátor. Druhou možností je vytvoření kopírujícího konstruktoru.

Pokud ale používáme objekty, která mají **některé atributy definované dynamicky** (např. pole), pak musíme rozhodnout, jakým způsobem budeme objekty kopírovat:

a) mělká kopie objektu (propojení atributů adresami v paměti, vzájemná závislost objektů)
b) hluboká kopie objektu (nezávislost objektů)
c) copy on write (metoda počítaných referencí)

Volbu způsobu kopírování volíme podle potřeby zpracování dat...

Mělká kopie objektu

Při mělké kopii se jednoduše atributy jednoho objektu přiřadí k atributům druhého objektu. To znamená, že v případě, kdy máme dynamicky alokované proměnné, tak oba dva objekty pracují se stejnou adresou v paměti – to znamená, že pokud změníme hodnotu atributu jednoho objektu, pak se zároveň změní hodnota odpovídajícího atributu druhého objektu.

Výhody – rychlost provedení (krátký kód a velká rychlost zpracování dat)

Nevýhody – takto vytvořené objekty jsou na sobě závislé (mají společná data, pokud je změníme u jednoho objektu, pak se změna projeví u všech ostatních objektů – ve všech kopiích

Hluboká kopie objektu

Při hlubokém kopírování nejprve alokujeme nové místo v paměti a do něj zkopírujeme hodnoty atributů zdrojového objektu.

Např.: V objektu A bude pole. V objektu B alokujeme paměť odpovídající velikosti pole a následně pole z objektu A překopírujeme do objektu B prvek po prvku.

Výhody - nezávislost dat atributů jednotlivých objektů, při změně hodnot atributu jednoho objektu nejsou ovlivněny hodnoty atributů ostatních objektů

Nevýhody - složitější kód programu
- pomalejší zpracování

Copy on write

Při této metodě nejprve objekty zkopírujeme pomocí mělké kopie a ve chvíli, kdy chceme objekt změnit, provedeme hlubokou kopii objektu. To je výhodné v případě, že neměníme všechny objekty, které jsme zkopírovali.

Pro „copy on write“ se používá metoda počítaných referencí, při které má každý objekt, kromě svých dat, ještě informaci o tom, kolikrát byl zkopírován (protože se jedná o mělkou kopii, údaj je přenesen do originálu i všech jeho kopií). Z tohoto důvodu má každý objekt atribut pro ukládání počtů kopií. Číslo představuje počet závislých (stejných) objektů (se stejnou adresou v RAM).

Když objekt kopírujeme, tak tuto hodnotu vždy o jedničku zvětšíme.

V případě, že objekt měníme (některý z jeho atributů), nejprve zjistíme, zda objekt byl již někdy kopírován. V případě, že ne, můžeme ho rovnou změnit. V případě, že ano, pak hodnotu počtu zkopírování snížíme o jedničku, provedeme hlubokou kopii tohoto objektu a poté nový objekt změníme (nově vytvořený objekt již nemá žádnou vazbu na zdrojový objekt).

Výhody - relativně rychlý algoritmus, při změně atributů objektu přestanou být zdrojový a cílový objekt na sobě závislé

Nevýhody - složitý a dlouhý kód