Дублиращ се код в програма обозначава лош дизайн/структура на кода и прави

поддръжката му трудна. Когато е нужна промяна във кода, то тази промяна трябва да се

направи на всяко копие на този код. А самият код става дълъг и неоптимизиран. За

оптимизиране може да се ползват изброен тип или като се разбие на по-малки методи,

правейки кода по гъвкав, сбит и прегледен.

Пример:

*//keys е масив който съдържа всички клавиши дали са натиснати или не.*

**void** KeysPressed(**bool** keys[]) {

**cout** << “Pressed keys are: \n“;

**if** (keys[*VK\_ENTER*]) **cout** << “Pressed Enter \n”;

**if** (keys[*VK\_LEFT*]) **cout** << “Pressed Left \n”;

**if** (keys[*VK\_UP*]) **cout** << “Pressed UP \n”;

… //Още клавиши

**if** (keys[*'A'*]) **cout** << “Pressed A \n”;

**if** (keys[*'B'*]) **cout** << “Pressed B \n”;

… //Още букви

}

Ако в този пример се опитаме да направим превод на български, трябваше на всеки ред

да променим “Pressed “ със “Натиснат ”.

Пример- оправен:

**void** KeysPressed(**bool** keys[]) {

**cout** << “Pressed keys are: “;

**for**(**int** i; i<= KEYSMAX; i++)

**if** (keys[i]) *//Ако i-тият клавиш е натиснат*

**cout** << „Pressed “ << getKeyName(i) << „ \n”;

}

За да се избегне повтарянето на цели методи в сходни класове се използва

наследяването на класове, където от даден базов клас се наследяват всичките функции

и полета, и няма нужда да се имплементират наново в новият ни клас, което ни

позволява да опишеме само нещата по които се различава новия клас от базовия.