**Програмни практики: наследяемост и поломорфизъм в програма.**

Основна характеристика на C++ е наследяването. То позволява да се създават класове,

получени от други класове, така че те автомачино включват някои от свойте

„родителски” членове, както и свойте собствени.

Когато имаме много общи полета за различни обекти се използва наследяване. Пример:

триъгълник и правоъгълник, общите неща са страна и височина, защото са подклас на

полигон. Когато един клас наследява друг се използва следната форма.

**class** <име-производен-клас>: <тип-достъп> <име-базов-клас> {

//член-променливи специфични за този клас

//член-функции специфични за този клас

};

Тук тип-достъп е една от трите ключови думи: public, private или protected. Ако

базовият клас е деклариран като public в производния клас, всички public, private и

protected компоненти на базовия клас се наследяват съответно като public, private и

protected компоненти на производния клас. При private е по-разично. Когато клас

наследява друг клас като private то той наследява всички public членове като private.

Не се наследяват конструктори, деструктори и приятелски функции.

Тъй като конст. не се наследяват автоматично и трябва да се предефинират наново, като

аргументите на конструкторът от наследеният клас трябва съвпадат или разсширяват

аргументите на конст. от базовия клас. Един конструктор може да наследи базов

конструктор така:

<производен-конструктор>(<списък аргументи>):<базов-клас>(<списък аргументи>) {

//тяло на конструктора на производния клас

}

Полиморфизъм-способността на обекти, принадлежащи към различни класове(типове)

да изпълняват метод извикан с еднакво име, всеки според подходящия начин.

Операторът overloading на цифрови оператори =,-,\*,/ позволява полиморфична

обработка на различни числени типове: int,double т.н. всеки, от които има различни

диапазони и представяне.

Полиморфизмът се използва при наследяването на класове. При тях е много подходящо

общите методи да бъдат с еднакви имена. Полиморфизмът не е overloading или

overriding. Полиморфизмът касае само до приложение със специфично

изпълнение(представяне) на интерфейс или по-общо базов клас. Методът overloading се

отнася до методи с еднакво име, но различни сигнатути. Методът overriding(динамичен

полиморфизъм) означава да се предостави нова реализация на дадения метод, различна

от наследената от суперкласа реализация. Новата реализация в подкласа има същото

име, същия брой и тип на параметрите и връща същия резултат като реализацията на

метода в подкласа.

Пример: Методът Speak() за куче ще върне лае, а за прасе същият ще върне грухти.

Кучето и прасето наследяват метода Speak от животно, но методите на подкласа

override методи на базовия клас известно като overring полиморфизъм.