# Corso Programmazione ad Oggetti
# Corso di Laurea in Informatica
# Esercitazione di Laboratorio del 06/05/2016

**Esercizio 1.** Realizzare una classe Coppia, che rappresenti una coppia di interi.

Dotare la classe di:

- un costruttore senza parametri;
- un costruttore con due parametri;
- i metodi *get* e *set* per il primo ed il secondo intero;
- un metodo *stampa* per la stampa su standard output;
- un metodo *calcolaSomma*, che restituisca la somma data dai due interi che compongono la coppia.

Realizzare, quindi, una classe ***Tripla***, che erediti da ***Coppia***, e che rappresenti una tripla di interi. Quale tipo di ereditarietà è più opportuno?

Implementare il costruttore senza parametri e un costruttore con tre parametri rappresentanti i tre interi che compongono la tripla. Dotare la classe dei metodi *get* e *set* per il terzo intero, e ridefinire i metodi *stampa* e *calcolaSomma*.

Realizzare un main in cui si creano degli oggetti di tipo ***Coppia*** e di tipo ***Tripla*** e verificare il funzionamento dei metodi ridefiniti. Verificare che sia possibile richiamare i metodi *get* e *set* sui primi due interi su un oggetto di tipo ***Tripla***. Provare a cambiare il tipo di ereditarietà ed analizzare i cambiamenti si verificano.

**Esercizio 2.** Applicare l'ereditarietà per implementare una classe rappresentante una coda di interi dal prodotto limitato. Una coda di interi dal prodotto limitato è una coda di interi in cui il prodotto di tutti i numeri in essa contenuti è inferiore ad un dato valore specificato in fase di costruzione. La classe dovrà comportarsi come una coda. Tuttavia, quando si richiede di inserire un nuovo numero in una coda dal prodotto limitato, l'inserimento ha successo a patto che l'aggiunta del nuovo elemento non faccia eccedere il limite massimo del prodotto.

**Esercizio 3.** Al pronto soccorso, i pazienti vengono registrati e gli viene fornito un codice di gravità. La classe ***Paziente*** è definita nel seguente modo:

```
#ifndef PAZIENTE_H_
#define PAZIENTE_H_

#include <iostream>
#include <string>
using namespace std;

enum CodiceUrgenza {
	BIANCO = 0, VERDE, GIALLO, ROSSO
};

class Paziente {

private:
	string nome;
	CodiceUrgenza codice;

public:
	Paziente() : nome(""), codice(BIANCO) {}
	Paziente(string n, CodiceUrgenza c) : nome(n), codice(c) {}

	CodiceUrgenza getCodice() const { return codice; }
	void setCodice(CodiceUrgenza codice) { this->codice = codice; }

	const string& getNome() const {  return nome; }
	void setNome(const string& nome) { this->nome = nome; }

	bool operator==(const Paziente& p) const { return codice == p.codice; }
```

# Corso Programmazione ad Oggetti
# Corso di Laurea in Informatica
# Esercitazione di Laboratorio del 06/05/2016

```
        bool operator<(const Paziente& p) const { return codice < p.codice; }
        bool operator<=(const Paziente& p) const { return codice <= p.codice; }
        bool operator>(const Paziente& p) const { return codice > p.codice; }
        bool operator>=(const Paziente& p) const { return codice >= p.codice; }

};

#endif
```

Utilizzare l'ereditarietà per realizzare la classe ***CodaPazienti*** che implementi una coda con priorità di pazienti, in cui i pazienti che arrivano sono serviti in base al codice di urgenza (da rosso a bianco) e, a parità di urgenza, nel consueto ordine FIFO (First In First Out).

**Alcuni suggerimenti per lo svolgimento:**

- Non utilizzare caratteri accentati per nomi di campi, variabili o metodi.
- Per il carattere tilde (~) occorre digitare: ALT + 126 su sistemi Windows, ALT gr + ì su sistemi Linux, ALT + 5 su sistemi Mac OS X.
- Per ogni esercizio realizzare una cartella contenente tutti i file creati.
- Si consiglia di utilizzare un semplice editor di testo (ad esempio, gedit), e compilare da linea di comando. Per compilare occorre passare tutti i file sorgente (.cpp) al compilatore.
- Minimizzare le inclusioni.