Laboratorio Tecnologie dell' Informazione

PROVA SCRITTA – 26 Giugno 2009
Parte I – TEORIA

1 (10 punti)

Si descrivano i meccanismi di passaggio dei parametri a metodi e funzioni del C++, indicando le maggiori differenze rispetto al C.

2 (10 punti)

Si descrivano gli specificatori di accesso ai membri di una classe (es. `public`,…) e come questi varino secondo il tipo di ereditarietà in una classe derivata.

3 (10 punti)

Gli attributi ed i metodi membri di una classe possono essere dichiarati `static`. Indicare cosa questo significhi per ogni tipo di membro. Fornire un esempio di semplice classe in cui viene definito un attributo di tipo intero ed un metodo getter per tale attributo, mostrando anche come inizializzare l'attributo e come invocare il metodo.

4 (6 punti)

Completare la seguente classe scrivendo una dichiarazione di metodo "void draw()" in modo tale che risulti astratta. Estendere la classe creando due sottoclassi polimorfiche Rectangle e Triangle che specializzano tale metodo.

```
class Shape {
public:
      Shape();
      virtual ~Shape();
      ................;
};
```

Le sottoclassi devono poi essere utilizzabili nel seguente programma, che deve essere completato nella riga indicata da FIXME:

```
#include <iostream>
#include <vector>
#include "Shape.h"
#include "Triangle.h"
#include "Rectangle.h"
using namespace std;

int main() {

      vector<Shape*> shapes;
      Triangle* aT = new Triangle;
      Rectangle* aR = new Rectangle;

      shapes.push_back(aT);
      shapes.push_back(aR);
      vector<Shape*>::const_iterator itr;
      for (itr = shapes.begin(); itr != shapes.end(); itr++ )
            (*itr)...draw(); // FIXME: come invoco questo draw ?

      return 0;
}
```

5 (5 punti)

Scrivere un costruttore di copia ed un distruttore per la seguente classe.

```
class Buffer {
public:
	Buffer(int size) {
		_size = size;
		_buffer = new char[_size];
	};
	virtual ~Buffer() {
		// FIXME scrivere il distruttore !
	};
	void setCharAt(int pos, char value) {
		if( pos>=0 && pos < _size )
			_buffer[pos] = value;
	};
	char getCharAt(int pos){
		if( pos>=0 && pos < _size )
			return _buffer[pos];
		else
			return '\0';
	};
	int getSize() { return _size; };

private:
	char* _buffer;
	int _size;
};
```

6 (9 punti)

Si definiscano due classi: una che rappresenta un pixel nello spazio di colore RGB (3 interi, uno per ogni canale di colore R, G e B) ed una che rappresenta un pixel nello spazio di colore a livelli di grigio (1 unsigned char). Si definisca una classe template che rappresenta un'immagine bidimensionale. Le dimensioni dell'immagine (X ed Y) sono passate nel costruttore. La classe ha i metodi generici getPixel e setPixel che rendono e impostano il valore di un pixel ad una data coordinata.

7 (10 punti)

Si scriva un Class Adapter per la seguente classe RGBPixel per fare in modo che possa essere usata con un client che si aspetta l'interfaccia definita nella classe IGrayPixel. Se ne disegni il diagramma UML di classe.
Per convertire in livello di grigio una tripletta RGB si usi la formula: Per calcolare la luminosità di un pixel, a partire dalla tripletta RGB corrispondente, si usi la formula:
grigio = 0.299 * R + 0.587 * G + 0.114 * B

```
class RGBPixel {
public:
     RGBPixel() { _R = _G = _B = 100; };
     int getR() { return _R; };
     int getG() { return _G; };
     int getB() { return _B; };
     void setR(int R) { _R = R; };
     void setG(int G) { _G = G; };
     void setB(int B) { _B = B; };

private:
     int _R;
     int _G;
     int _B;
};

class IGrayPixel {
public:
     virtual unsigned char getPixel() = 0; // return gray level
};
```