Università della Calabria

Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti

A.A. 2006/2007

Introduzione al corso

Paolo Trunfio *

* DEIS, Università della Calabria – http://si.deis.unical.it/~trunfio

Obiettivo del corso

Obiettivo del corso è acquisire le conoscenze necessarie per realizzare applicazioni informatiche secondo il paradigma della **programmazione orientata agli oggetti**, utilizzando il linguaggio **C++**.

Contenuti delle lezioni:

- Studio delle caratteristiche di base del C++ partendo dalle conoscenze su Java acquisite nel corso di Fondamenti di Informatica.
- Studio della programmazione orientata agli oggetti e sviluppo di applicazioni basate su tale paradigma, facendo uso dei meccanismi e delle librerie standard di C++.

Contenuti delle esercitazioni:

- Progettazione ed implementazione di applicazioni informatiche basate sui concetti presentati a lezione.

Informazioni sul corso

Docente:

Paolo Trunfio
DEIS - Cubo 41C, terzo piano

Materiale didattico:

- Lucidi delle lezioni: http://si.deis.unical.it/trunfio/teaching/poo0607/
- Lucidi ed altro materiale: http://icampus.ingegneria.unical.it/

Libro di testo:

S. B. Lippman, J. Lajoie. “C++: Corso di programmazione”.
Terza Edizione. Addison-Wesley, 2000.
1209 pagine.

oppure:

H. Schildt. “Guida al C++”.
Terza Edizione. Mc Graw-Hill, 2003.
585 pagine.

Informazioni sull’esame

Propedeuticità:

- Calcolo I.
- Fondamenti di Informatica.

Modalità di svolgimento dell’esame:

- Prova scritta obbligatoria.
- Prova orale facoltativa (obbligatoria nel caso in cui non sia stato conseguito un voto minimo nella prova scritta).

La programmazione orientata agli oggetti

- Nella **programmazione orientata agli oggetti** (**POO**) un programma è composto da un insieme di entità software, denominate **oggetti**, che interagiscono secondo determinati schemi. Esempi di linguaggi che supportano la programmazione orientata agli oggetti sono **Java** e **C++**.
- La programmazione orientata agli oggetti differisce dalla cosiddetta **programmazione imperativa**, basata sul concetto di **istruzione**: un programma consiste in una sequenza di istruzioni che specificano in modo dettagliato le operazioni che devono essere eseguite dall'elaboratore per risolvere il problema dato. Esempi di linguaggi imperativi sono **Fortran**, **Pascal** e **C**.
- E' possibile utilizzare i linguaggi orientati agli oggetti per realizzare programmi prevalentemente imperativi (*come nel corso di Fondamenti di Informatica*), ma in tal modo non si sfruttano i vantaggi derivanti dall'uso dei meccanismi propri della programmazione orientata agli oggetti.

Il linguaggio C++

- **C++** è un linguaggio di programmazione orientato agli oggetti sviluppato da Bjarne Stroustrup a partire dai primi anni 80. La versione standard di C++ è stata rilasciata nel 1998.
- C++ è basato sul linguaggio **C**, dal quale eredita in gran parte la sintassi. La maggior parte del codice C è anche codice C++ valido, ma C non è esattamente un sottoinsieme di C++.
- Oltre al supporto alla programmazione orientata agli oggetti, C++ fornisce numerose caratteristiche aggiuntive rispetto a C, incluso il meccanismo dei *template* ed un'ampia libreria standard (la codiddetta *Standard Template Library* o *STL*).
- C++ presenta numerose similitudini con il linguaggio **Java**, del quale può essere considerato un progenitore. Chi conosce Java è in grado di scrivere semplici programmi C++ senza particolari difficoltà.

Java vs. C++: un esempio

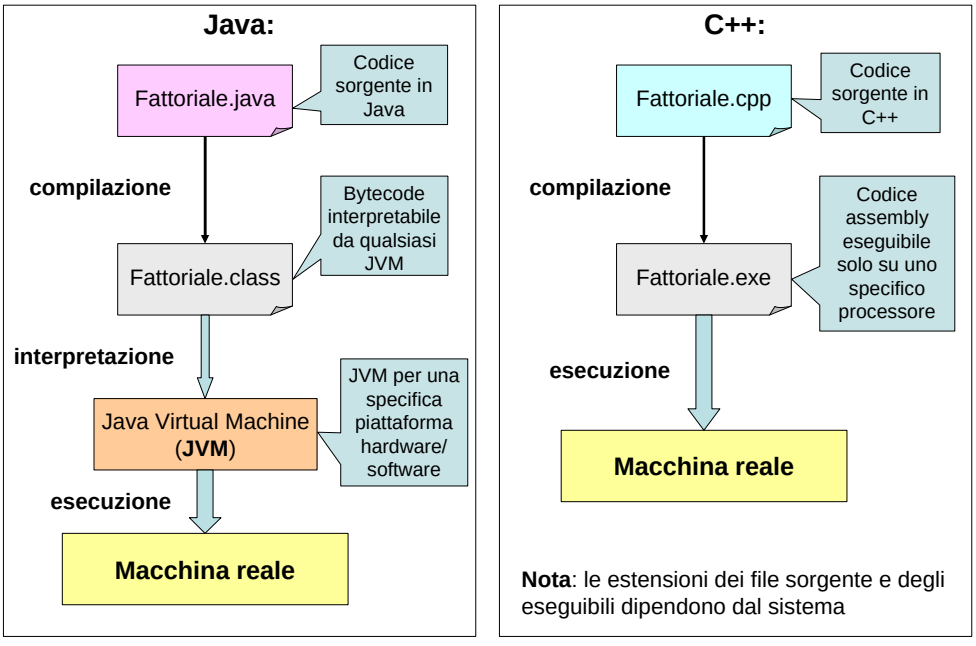
Fattoriale.java

```
import java.util.Scanner;

public class Fattoriale {

  public static int leggiPositivo() {
    Scanner sc = new Scanner(System.in);
    int n;
    do {
      System.out.print("Inserisci un numero positivo: ");
      n = sc.nextInt();
    } while (n < 0);
    return n;
  }

  public static int fattoriale(int n) {
    int fatt = 1;
    for (int k = n; k > 1; k--)
      fatt *= k;
    return fatt;
  }

  public static void main(String[] args) {
    int a = leggiPositivo();
    int f = fattoriale(a);
    System.out.print("Fattoriale di "+a+" = "+f+"\n");
  }
}
```

Fattoriale.cpp

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


int leggiPositivo() {
  int n;
  do {
    cout << "Inserisci un numero positivo: ";
    cin >> n;
  } while (n < 0);
  return n;
}


int fattoriale(int n) {
  int fatt = 1;
  for (int k = n; k > 1; k--)
    fatt *= k;
  return fatt;
}

int main() {
  int a = leggiPositivo();
  int f = fattoriale(a);
  cout << "Fattoriale di " << a << " = " << f << "\n";
  return 0;
}
```


Java vs. C++: compilazione ed esecuzione



Nota: le estensioni dei file sorgente e degli eseguibili dipendono dal sistema

Java vs. C++: portabilità ed efficienza

Portabilità:

- Il programma Java (bytecode) può essere interpretato ed eseguito su qualsiasi macchina che disponga di una Java Virtual Machine.
- Il programma C++ (eseguibile) può essere eseguito solo sulle macchine con la piattaforma per la quale il programma è stato generato.
- Per utilizzare il programma C++ su una piattaforma diversa è necessario ricompilarlo per quella piattaforma in modo da generare l'eseguibile specifico.

Efficienza:

- I programmi C++ sono tipicamente più veloci di quelli Java, principalmente perchè l'eseguibile C++ contiene istruzioni di basso livello (assembly) per uno specifico processore.
- Il bytecode Java contiene istruzioni di livello intermedio che devono essere interpretate da uno strato software aggiuntivo (la JVM) per poter essere eseguite.

Ambiente di programmazione

- L'ambiente di programmazione utilizzato durante il corso è Bloodshed **Dev-C++**.
- La versione 4.9.9.2 di Dev-C++ è installata sui computer del Laboratorio di Informatica.
- Il file di installazione **devcpp-4.9.9.2_setup.exe** (circa 9 MB) può essere copiato presso il Laboratorio di Informatica, scaricato dal sito del corso su icampus, oppure scaricato direttamente da Internet.

Dev-C++: un esempio

Supponiamo di voler scrivere ed eseguire il seguente programma utilizzando **Dev-C++:**

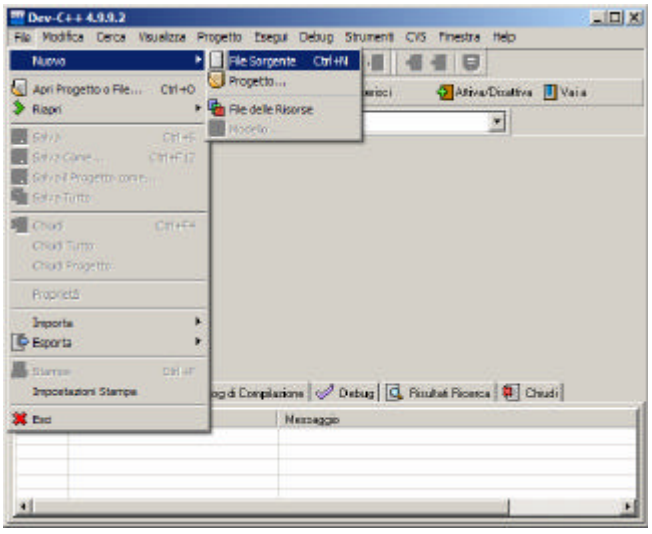
hello.cpp

```
#include <iostream>

using namespace std;

int main () {
  cout << "Ciao a tutti\n";
  system("pause");
  return 0;
}
```

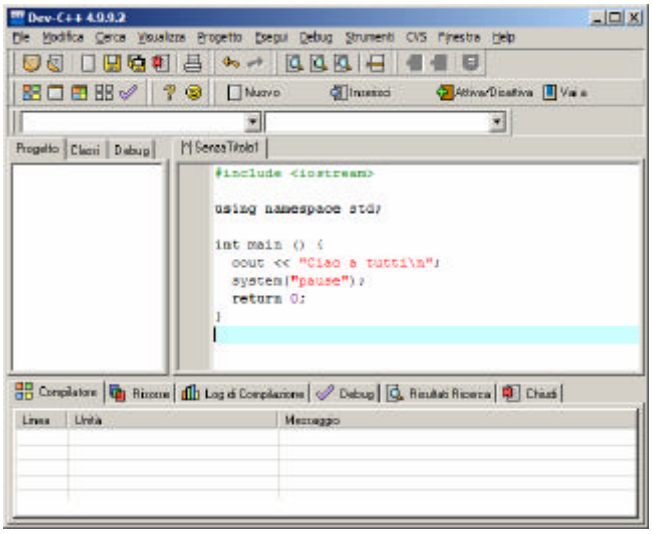
Con questa istruzione si blocca l'esecuzione fino a quando non si preme un tasto

Istruzione facoltativa

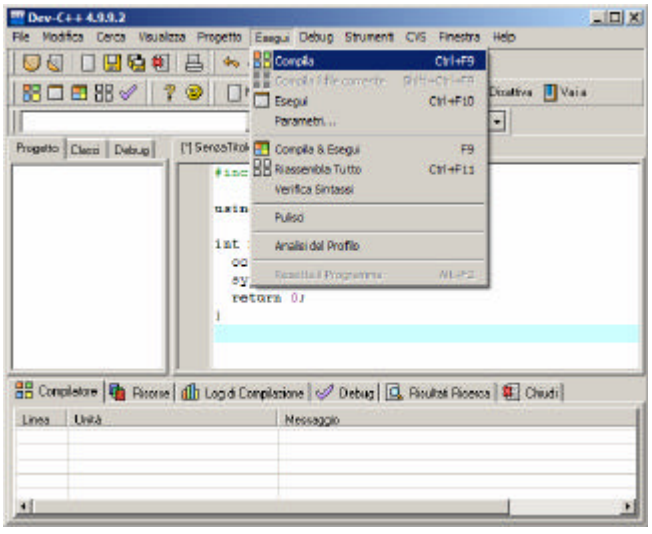
Dev-C++: un esempio



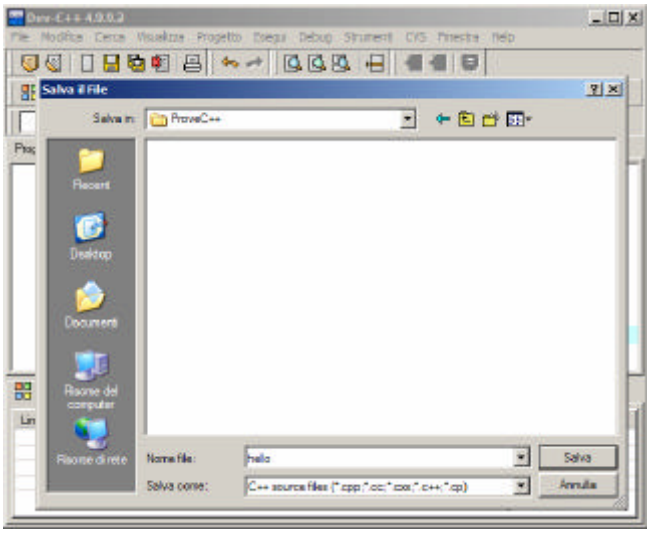
Dev-C++: editing



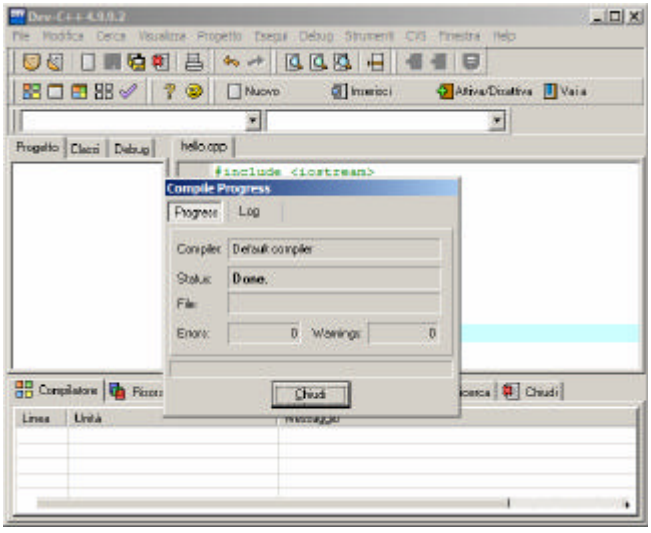
Dev-C++: compilazione



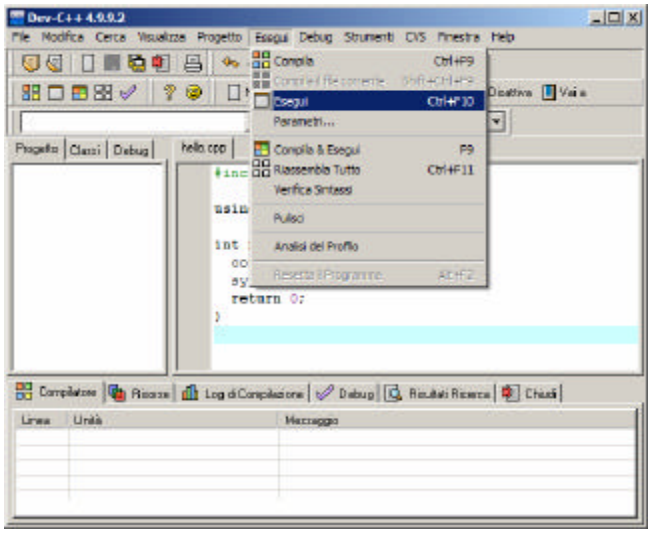
Dev-C++: compilazione



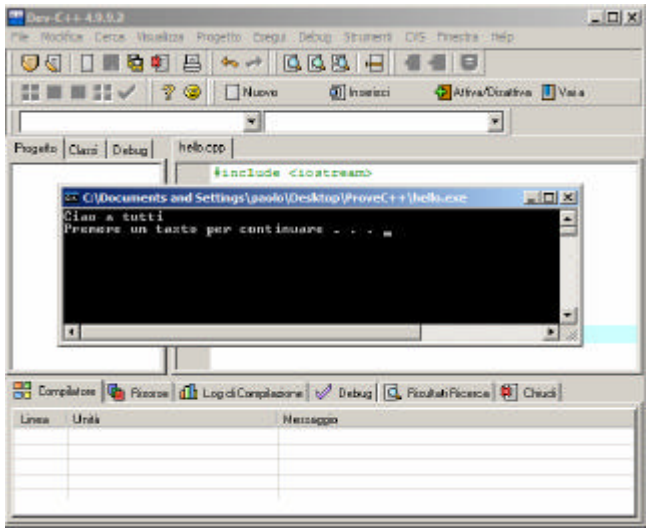
Dev-C++: compilazione



Dev-C++: esecuzione



Dev-C++: esecuzione



Dev-C++: esecuzione

Il file eseguibile (hello.exe) può essere eseguito anche direttamente: