



Laboratorio di Tecnologie dell'Informazione

Ing. Marco Bertini
marco.bertini@unifi.it
<http://www.micc.unifi.it/bertini/>



Presentazione del corso



Orario

- Lunedì: 14:00 - 17:00, aula 001 / 113+114
- Mercoledì: 8:15 - 11:15, aula 001
- Ricevimento: Giovedì - 14:00 - 18:00
su appuntamento:
marco.bertini@unifi.it
<http://www.micc.unifi.it/bertini/>
- Ufficio: MICC, Viale Morgagni 65, Firenze
<http://www.micc.unifi.it/>



Orario

- Lunedì: 14:00 - 17:00, aula 001 / 113+114
- Mercoledì: 8:15 - 11:15, aula 001

- Ricevime
su appun
marco.be
<http://www>

- Ufficio: M
<http://www>

Vacanze di Pasqua
In occasione delle prossime festività pasquali c'è tipicamente una sospensione della didattica di 1 settimana.



Scopo del corso

- Acquisire una conoscenza di base di meccanismi di analisi e programmazione object oriented.
- Imparare la programmazione object oriented in C++.
- Acquisire conoscenze relativi ad alcuni pattern di progettazione del software.



Scopo del corso

- Acquisire le conoscenze sufficienti per sviluppare un semplice videogioco di tipo "Rogue".
- Imparare a programmare in C++.
- Acquisire le conoscenze di base per lo sviluppo di progetti di programmazione orientati a oggetti.





Programma del corso

- Il linguaggio C++:
 - Data Abstraction
 - Classi e oggetti
 - I metodi
 - Operator Overloading
 - Class Inheritance e Multiple Inheritance
 - Funzioni virtuali e classi di base astratte
 - Polimorfismo
 - Programmazione generica e template
 - STL
 - La gestione delle eccezioni
-



Programma del corso

- Meccanismi di analisi e programmazione object oriented
 - incapsulamento
 - delega
 - inversione di responsabilità
 - sostituibilità
 - ereditarietà di implementazione e di interfaccia
 - problema della classe di base fragile
 - allocazione delle responsabilità, coesione e accoppiamento



Programma del corso

- Introduzione ai design pattern
- Design pattern fondamentali:
 - Observer
 - Factory
 - Class Adapter
 - Object Adapter



Modalità di svolgimento dell'esame - I

- Stesso schema del corso di Fondamenti di Informatica
- L'esame si compone di una prova scritta e una orale.
- La prova scritta consiste in alcuni elaborati di programmazione e nella discussione di contenuti del programma. La prova è organizzata “a batteria” in due parti di 45-60 minuti ciascuna: nella prima parte si devono dare risposte a questioni di natura teorica, nella seconda viene svolto un esercizio di programmazione.
- La prova scritta è svolta su carta. Al termine, viene presentata e discussa la soluzione. Successivamente i candidati ricevono la fotocopia del loro elaborato.
- Per accedere alla prova orale, il candidato deve correggere il proprio elaborato, riportando le correzioni in maniera visibile sulla fotocopia. Il candidato deve anche realizzare il programma corretto e funzionante che corregge l'elaborato e lo completa facendone un programma autocontenuto. Il candidato deve infine fornire una autovalutazione del proprio elaborato, in base al valore attribuito a ciascuna parte della prova, alla discussione della soluzione, all'esperienza acquisita nella correzione e realizzazione effettiva del programma.



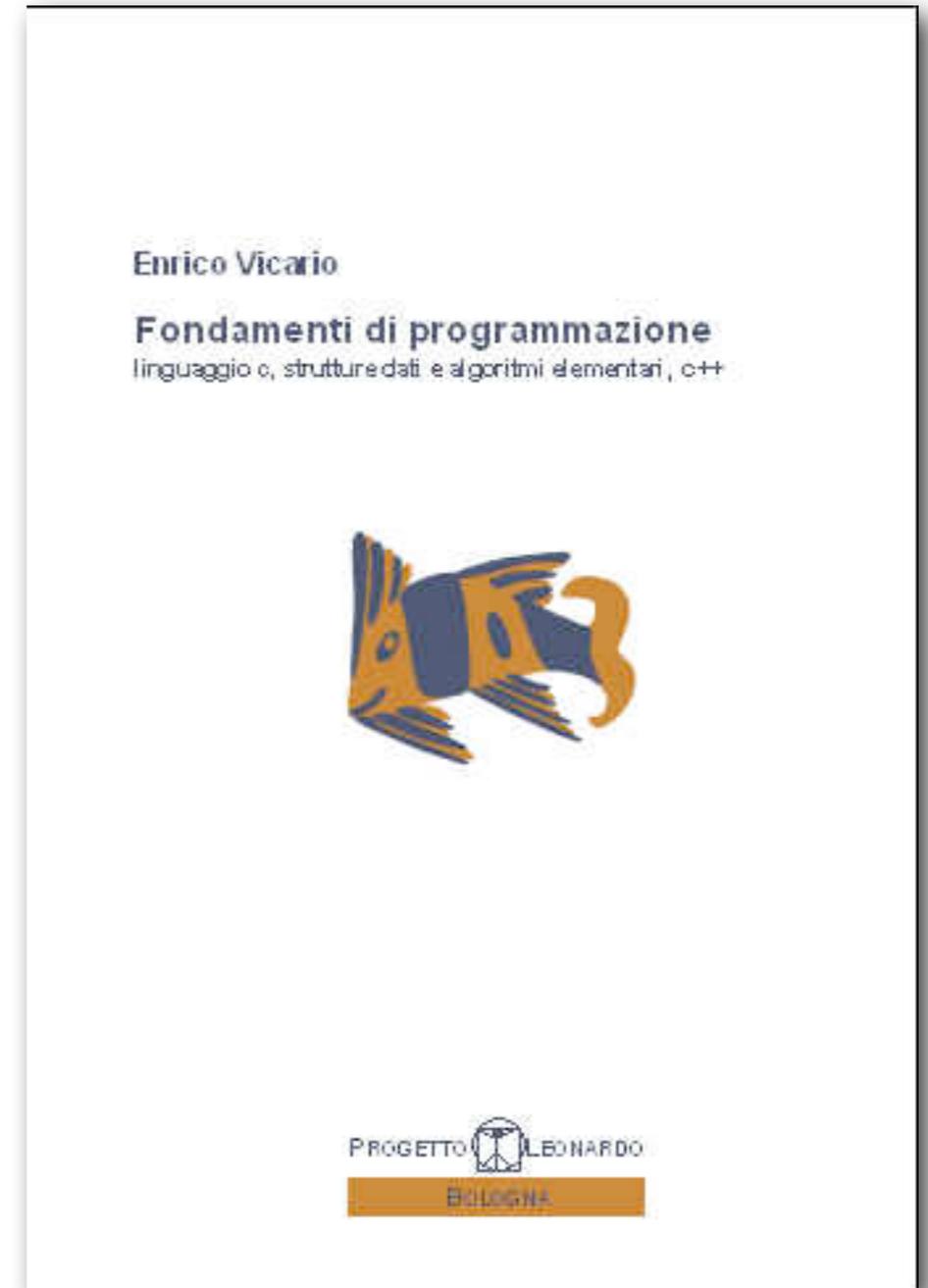
Modalità di svolgimento dell'esame - 2

- In alternativa è possibile stabilire degli elaborati relativi alla creazione di software.
- Il tema dell'elaborato deve essere concordato preventivamente.
- E' preferibile sviluppare un proprio progetto, in alternativa possibili idee di elaborato sono
 1. un'applicazione per la gestione di agende come iCal, senza tutta la parte di gestione di calendari multipli e rete, usando WxWidgets (o QT) per la GUI.
 2. un task manager semplice (<http://lifehacker.com/tag/todo-manager/> per ispirarsi)
 3. un programma per prendere note (come Tomboy).
 4. un gioco. Niente campi minati/gioco della scopa, altri giochi a piacere: OK.
- Info e link utili sulla pagina web del corso



Libro di testo

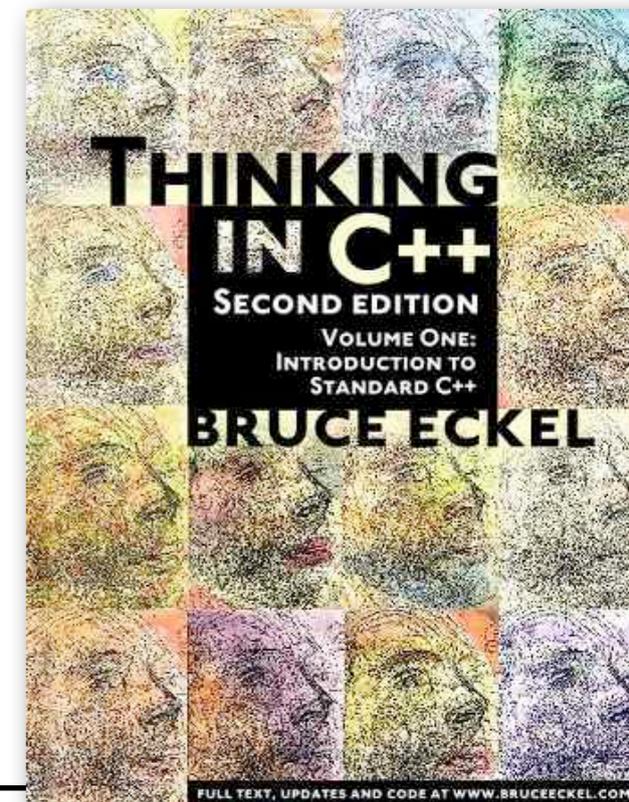
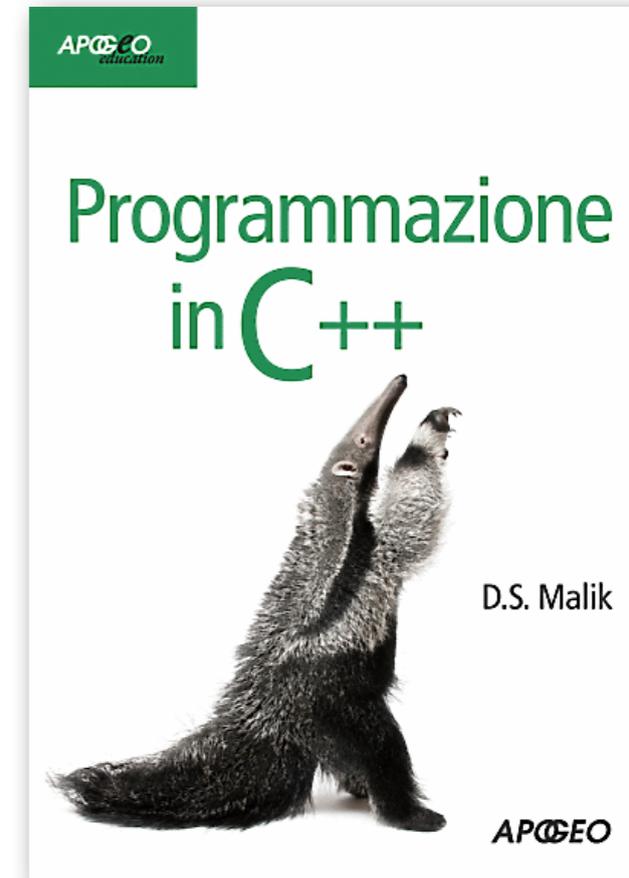
- E. Vicario, “Fondamenti di Programmazione: linguaggio c, strutture dati e algoritmi elementari, c++” Editrice Esculapio, Bologna, maggio 2006.





Libri consigliati

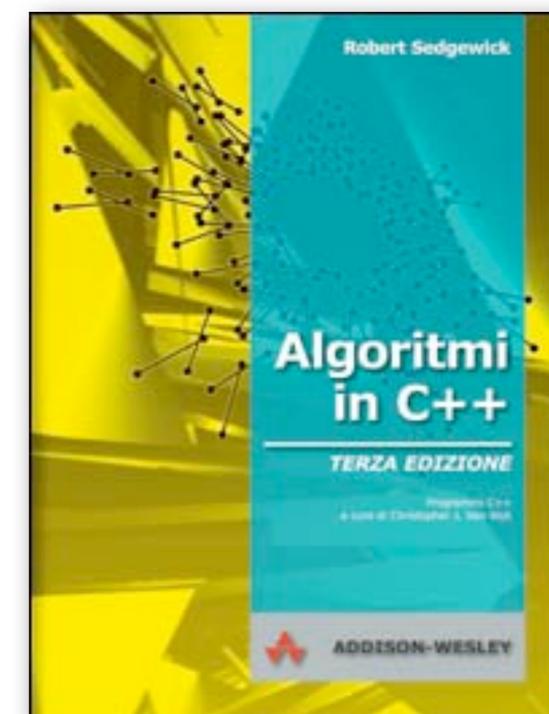
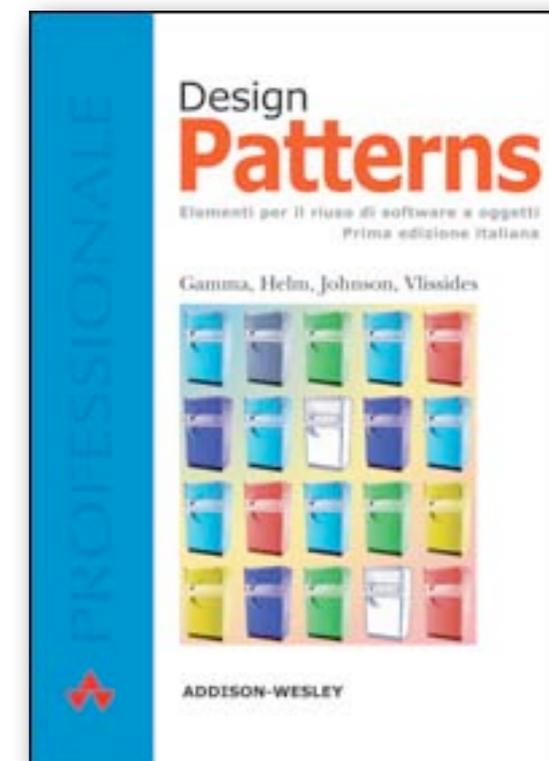
- D.S. Malik, "Programmazione in C++", Apogeo
- B. Eckel, "Thinking in C++", disponibile gratuitamente su: <http://www.mindview.net/Books/DownloadSites>





Libri utili

- E.Gamma, R.Helm, R.Johnson, J.M.Vlissides, “Design Patterns”, Pearson Education
- R. Sedgewick, “Algoritmi in C++”, Pearson Education





Link utili

- Sulla pagina del corso sono forniti link utili, relativi agli argomenti svolti a lezione ed in generale su programmazione C++ e design pattern

(EN); [Wikipedia: ereditarietà](#) (IT); C++ FAQ: [inheritance](#), [multiple inheritance](#) e [virtual inheritance](#) (EN). [Copy constructor](#), [operatore = sovraccaricato](#) e [shallow copy](#) (EN); [copia di oggetti](#) (Wikipedia, EN); [overloading di <<](#) (EN).

- [Templates - vecchio](#) (8.5 MB)
Materiale aggiuntivo: [discussione in cui si mostra perché le definizioni delle funzioni template devono stare insieme alle loro dichiarazioni](#) (in particolare leggere l'ultimo intervento, EN); [Why we can't afford export](#) (PDF, EN); [Why can't I separate the definition of my templates class from it's declaration and put it inside a .cpp file?](#) (EN)
- [STL - Standard Template Library - vecchio](#)(11 MB)
Materiale aggiuntivo: [Standard Template Library Programmer's Guide](#) (EN); [STL containers](#) (EN); [STL algorithms](#) (EN); The C++ Standard Library - A Tutorial and Reference: [sito web del libro](#), con decine di esempi (EN); [Critica degli iteratori](#) (EN)
- [Eccezioni - vecchio](#)(5 MB)
Materiale aggiuntivo: [C++ Exception Safety: Issues and Best Practices](#) (EN); [Critica delle eccezioni](#) (EN)
- [Design patterns + Adapter - vecchio](#) (4 MB)
Materiale aggiuntivo: [Portland Pattern Repository](#) (EN); [Adapter pattern sul Portland Pattern Repository](#) (EN); [Adapter Design Pattern: tutorial, video ed esempi](#) (EN); [More C++ Idioms](#) (EN)
- [Design patterns: Observer - vecchio](#)(6 MB)
Materiale aggiuntivo: [Observer pattern su Wikipedia](#) (EN); [Observer pattern sul Portland Pattern Repository](#) (EN); [Observer Design Pattern: tutorial, video ed esempi](#) (EN)
- [Design pattern: Factory Method e Abstract Factory - vecchio](#) (12 MB)
Materiale aggiuntivo: [Factory Method pattern sul Portland Pattern Repository](#) (EN); [Abstract Factory pattern sul Portland Pattern Repository](#) (EN); [Factory Method pattern: tutorial, video ed esempi](#) (EN); [Abstract Factory pattern: tutorial, video ed esempi](#) (EN); [Singleton pattern sul Portland Pattern Repository](#) (EN)

Per motivi di tempo può non essere possibile vedere tutti gli aspetti del linguaggio C++ durante il corso. Consiglio la lettura completa di uno dei due libri di testo consigliati, o quantomeno dei seguenti tutorial:

- IO Stream: [Input/output via <iostream> and <cstdio>](#) (EN); [Learn About Input and Output](#) (EN); [Input/Output with files](#) (EN); [serializzazione](#) (EN)
- Namespace: [Namespaces](#) (EN)
- Casting: [Type Casting](#) (EN)

Blog interessanti sul C++ e la programmazione in generale:

- [Sutter's Mill](#)
- [c++ truths](#)
- [C++ Soup!](#)
- [Learning C++](#)
- [Antonio Gulli's coding playground](#)
- [The C++ Source](#)
- [Reddit C++](#)



Compilatori e IDE

- In laboratorio verrà usato il compilatore GNU C++ ed Eclipse+CDT come ambiente di sviluppo
- chi usa Windows deve installare MinGW (per GCC) + MSYS
- potete usare altre combinazioni IDE+compilatore sui vostri PC
- Link su tutorial/info installazione sono sulla pagina web del corso

