Programmazione

Prof. Marco Bertini
marco.bertini@unifi.it
http://www.micc.unifi.it/bertini/

Presentazione del corso

"Teach Yourself Programming in Ten Years"
- Peter Norvig, Director of Research at Google Inc.

Codici del corso

- B024281 (B047) - modulo: Programmazione
  E' il secondo modulo del corso integrato "Fondamenti di Informatica/Programmazione"

- B026220(B047) - Programmazione
  Corso a se stante, per agli studenti di CdL diversi di Ingegneria di Informatica

- I contenuti del corso e modalità di superamento dell'esame sono le medesime

Orario

- Lunedì: 14:00 - 17:00, aula 003
- Giovedì: 14:00 - 16:00, aula 002
- Ricevimento: Giovedì - 16:00 - 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/

Sito corso

- http://www.micc.unifi.it/bertini/
- http://e-l.unifi.it

Marco Bertini / teachin

Dipartimento di Ingegneria dell'Informazione - Univer

Menu

Home page

Research

Conference and institutional activity

Teaching

Parallel Computing

Programmazione

Laboratorio di Programmazione

Laboratorio di tecnologie dell'informazione

Fondamenti di Informatica I

Informatica - Scuola di Specializzazione Chirurgia Generale

Sistemi di Elaborazione

Programmazione

Novità

- 1 Marzo 2016: creazione della pagina

Orario e aule - Ricevimento - Annun

Inizio lezioni a.a. 2015-2016: 3 Marzo 2016.
Termine lezioni: 9 Giugno 2016.

- Mercoledì: 9.15-13.15 Aule 113+114.
  - Il laboratorio è diviso su d alle 11:15, dalle 11:15 all sono divisi in due gruppi (che comincia da A ad F, qu comincia da G a Z).

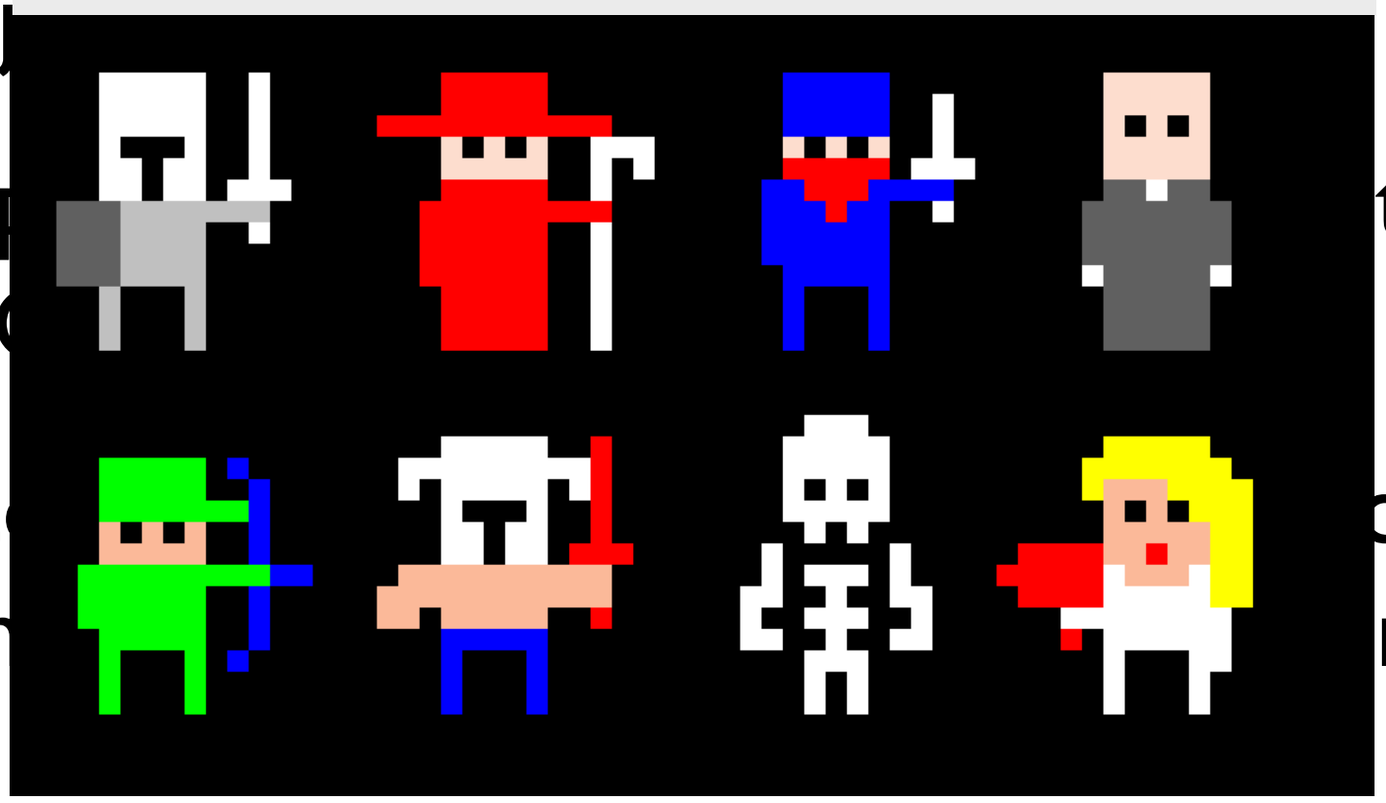
Scopo del corso

- 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 schemi di progettazione del software.

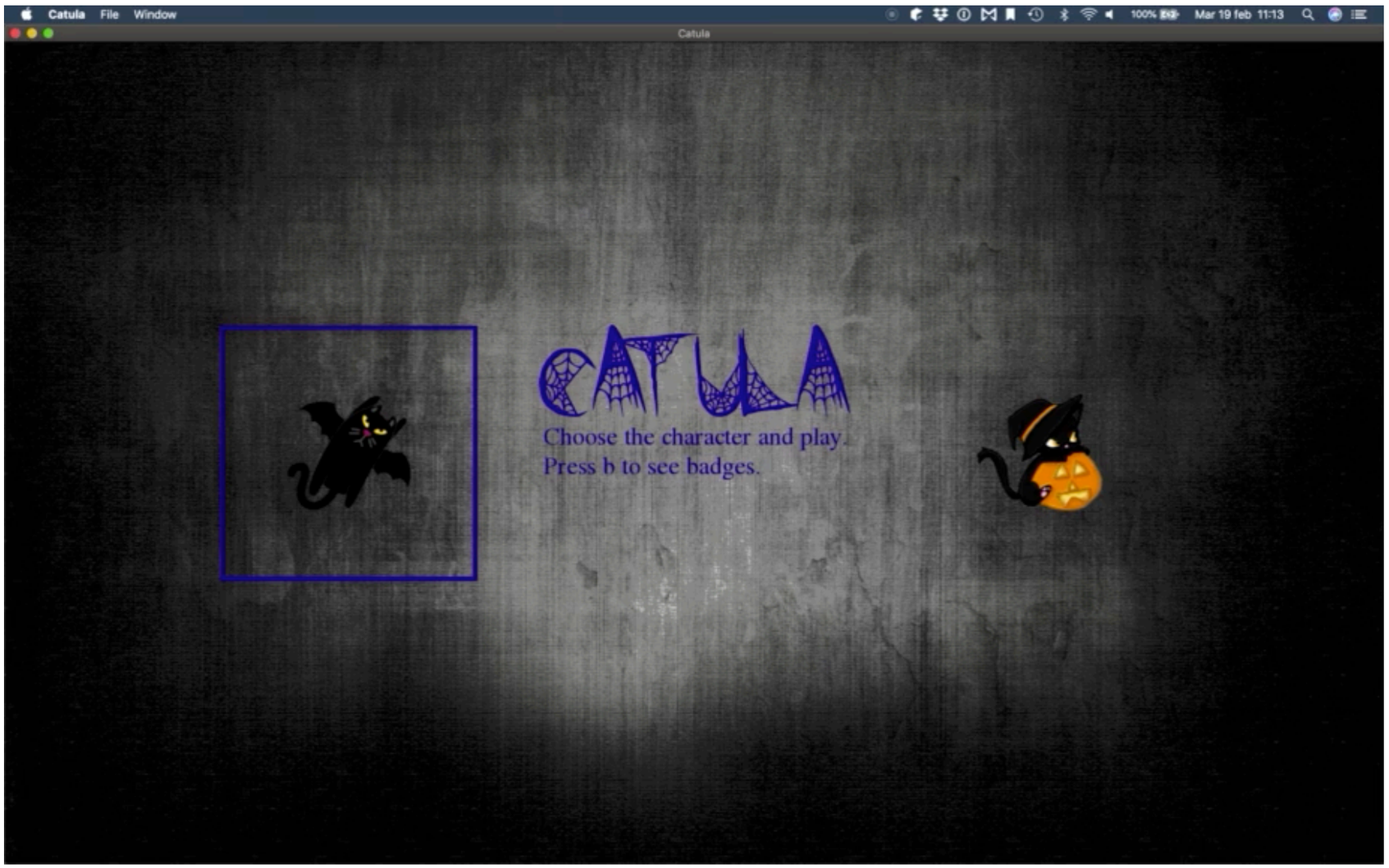
Scopo del corso

- Ac di
  me zione
  obj
- Imp t oriented
  in
- Ac cuni
  sch re.

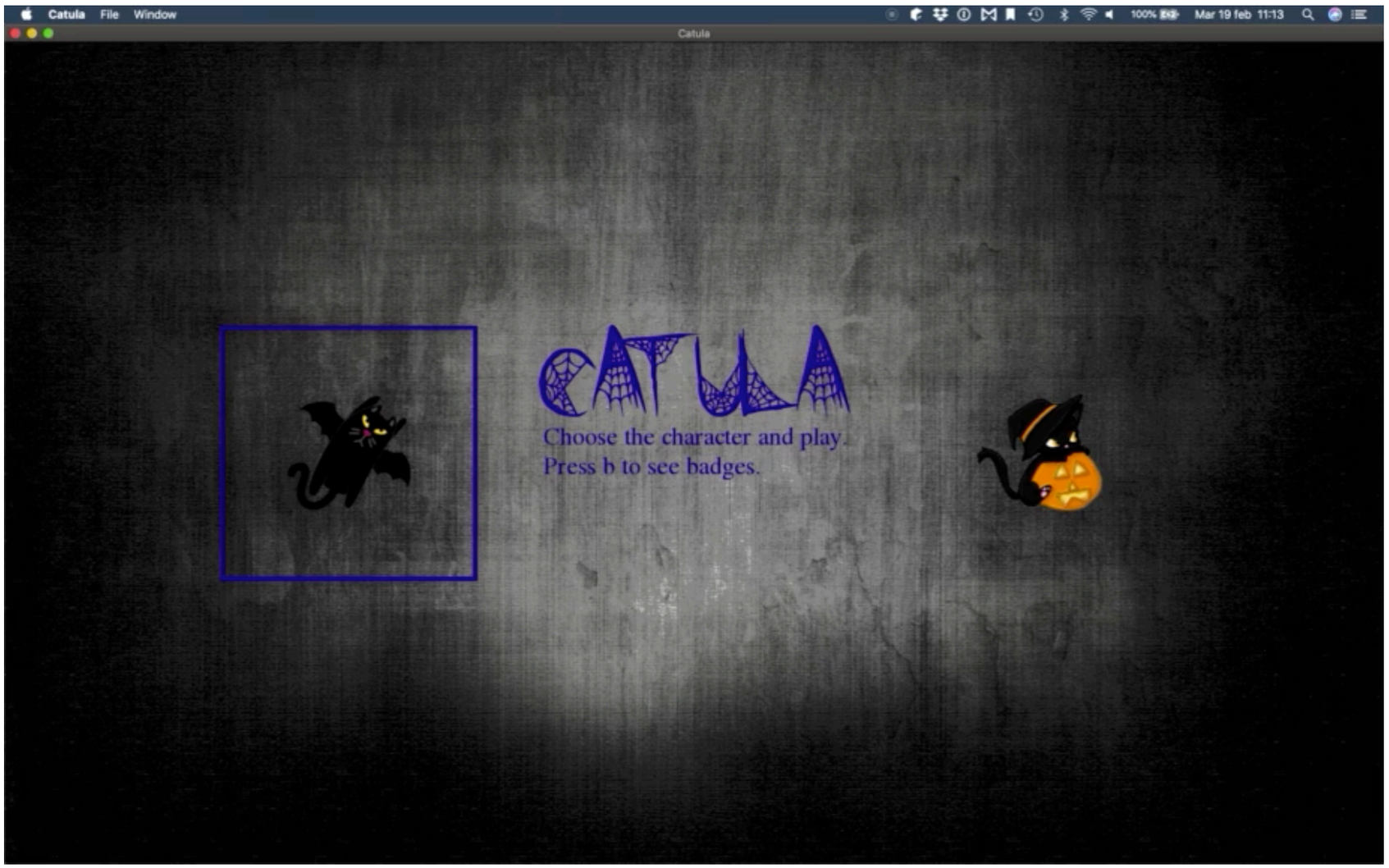
Avere sufficienti conoscenze per sviluppare un semplice videogame di tipo “Rogue”.



Esempi: elaborati di programmazioni



Esempi: elaborati di programmazioni



Esempi: elaborati di programmazioni

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:
  - Adapter
  - Factory
  - Observer

Modalità di svolgimento dell'esame - I

- L'esame si compone di una prova scritta (~2h durata) 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 ~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 (fogli A4, scritti su una sola facciata).
  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 i programmi corretti, funzionanti e auto-contenuti che implementano quanto richiesto nel compito.
  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. È necessario raggiungere un punteggio minimo di 15 punti per ognuna delle due sezioni del compito per poter fare l'orale.

È prevista una prova scritta intermedia (Lunedì 6 Maggio, pomeriggio) relativa al linguaggio C++ (~90m durata).
La correzione sarà effettuata dal docente e il superamento della prova consente di ridurre la prova scritta finale alla sola parte di programmazione.

- L'esame si compone di una prova scritta (~2h durata) 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 ~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 (fogli A4, scritti su una sola facciata).
  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 i programmi corretti, funzionanti e auto-contenuti che implementano quanto richiesto nel compito.
  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. È necessario raggiungere un punteggio minimo di 15 punti per ognuna delle due sezioni del compito per poter fare l'orale.

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. L'elaborato può comprendere quello del corso "Laboratorio di Programmazione".
- È 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

Libri di testo

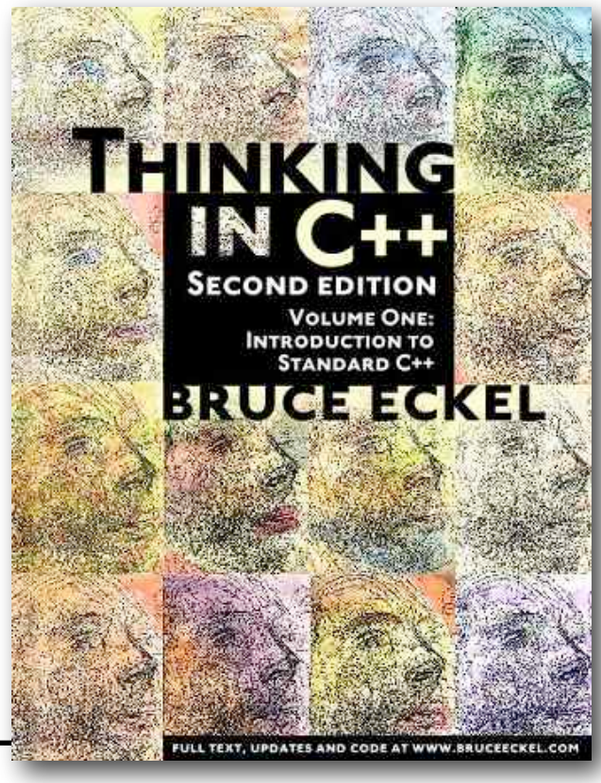
- M. Bertini, "Programmazione Object-Oriented in C++. Design Pattern e introduzione alle buone pratiche di programmazione", Editrice Esculapio,

- B. Stroustrup, "C++. Guida essenziale per programmatori", Pearson

Libri di testo

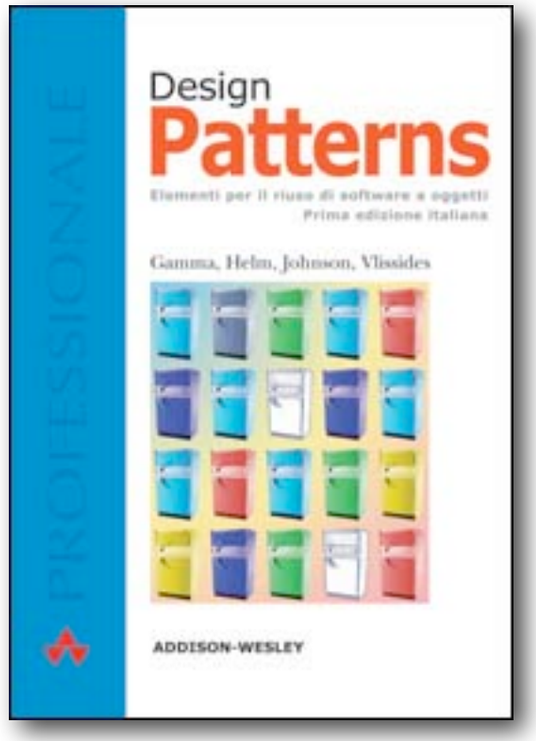
- L.J. Aguilar, "Fondamenti di programmazione in C++. Algoritmi, strutture dati e oggetti", McGraw-Hill

- 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:

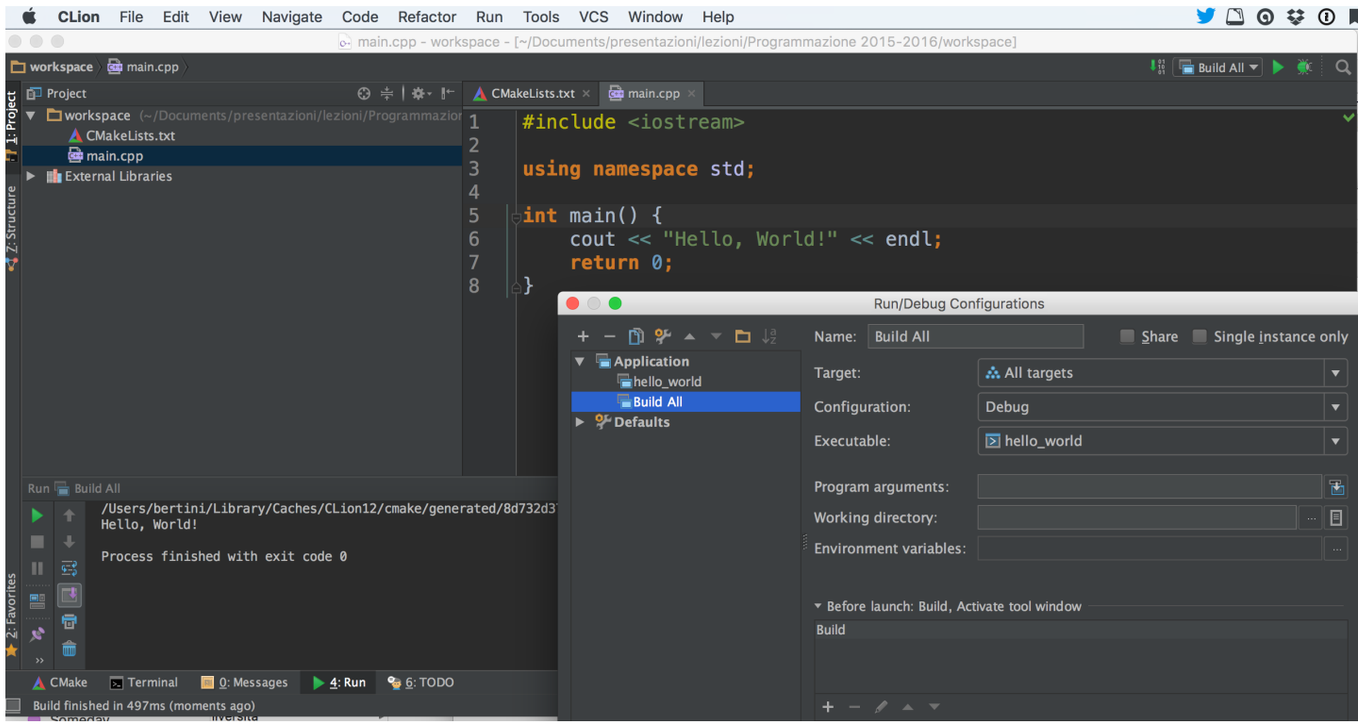
- IO Stream: **Input/output via <iostream> and <cstdio>** (EN); **Learn About Input and Output** (EN); **Input/Output with files** (EN); **serializzazione** (EN)
- Namespace: **Namespaces** (EN)
- Casting: **Type Casting** (EN)

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++**

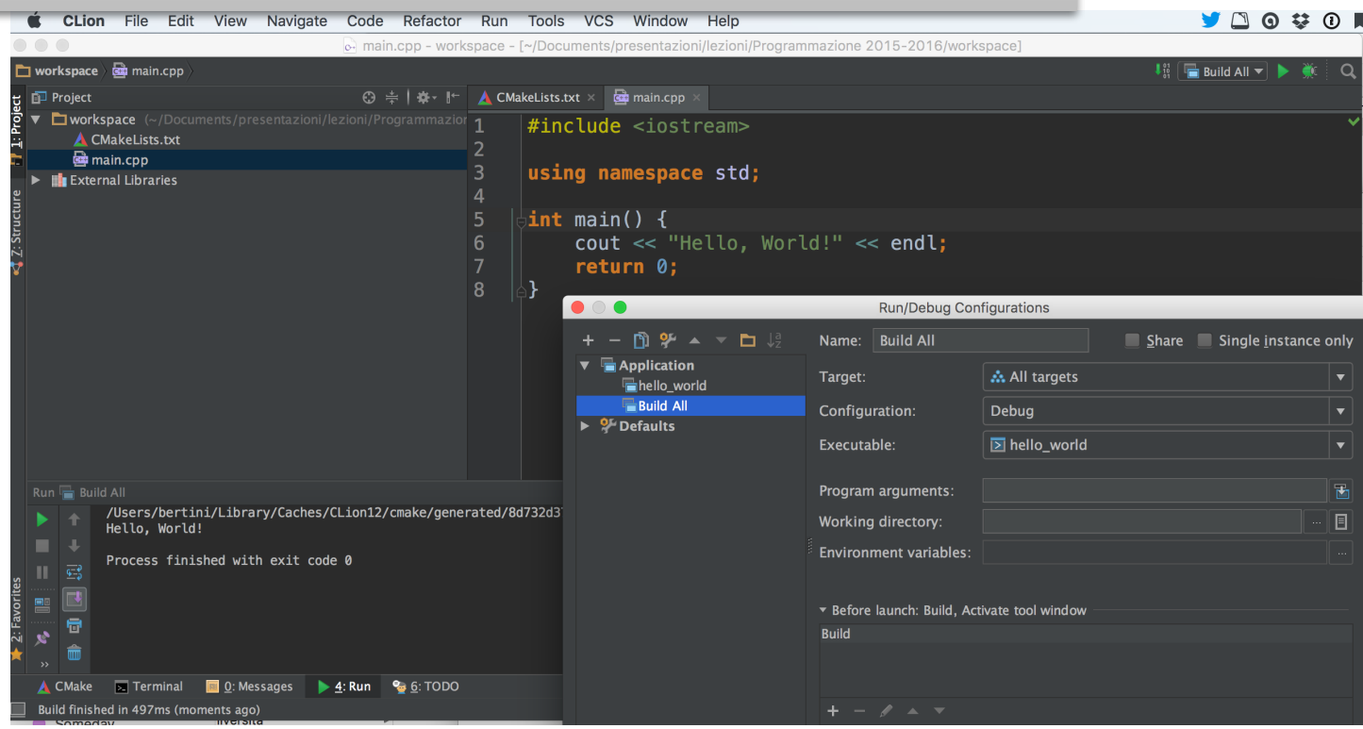
Compilatori e IDE

- In laboratorio verrà usato il compilatore GNU C++ e CLion 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

- È consigliato l'uso di sistemi *nix come macOS/OSX o Linux.
  Nel caso si usi Windows è consigliata l'installazione di una macchina virtuale Linux con VirtualBox



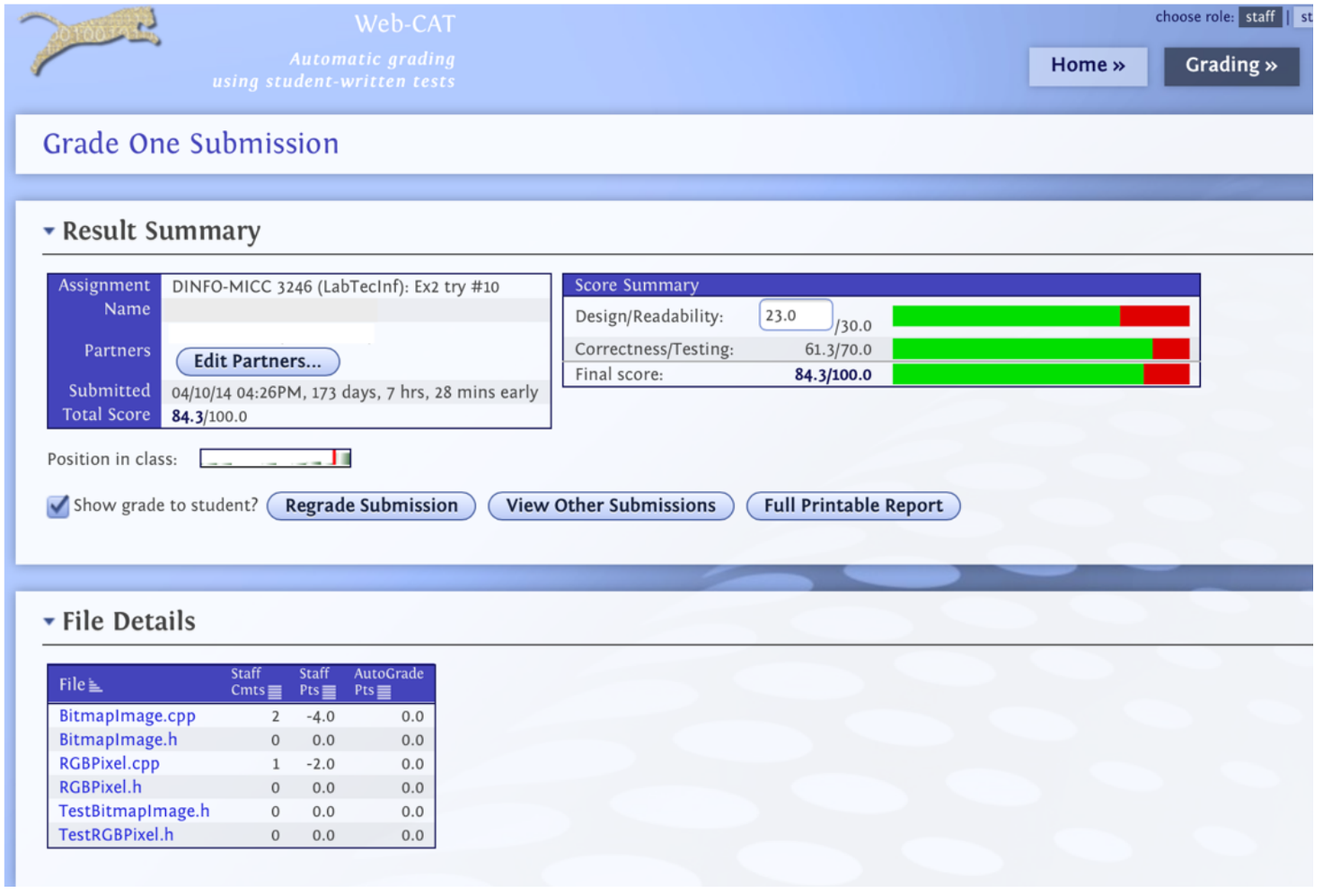
  - chi usa Windows deve installare MinGW (per GCC) + MSYS

  - potete usare altre combinazioni IDE + compilatore sui vostri PC

- Link su tutorial/info installazione sono sulla pagina web del corso

Esercitazioni a casa

- Saranno dati esercizi di programmazione da risolvere a casa.
- Le soluzioni saranno controllate in modo automatico tramite apposito strumento web.



Esercitazioni a casa

- Saranno dati esercizi di programmazione da risolvere a casa.
- Le soluzioni saranno controllate in modo automatico tramite apposito strumento web.

Laboratorio di programmazione

Cos'è ?

- Il Laboratorio di Programmazione consiste prevalentemente nello svolgimento da parte dello studente di un compito didattico aggiuntivo nell'ambito dell'insegnamento Fondamenti di Informatica / Programmazione (C.I.).
- Saranno fatte lezioni in laboratorio per apprendere l'uso di strumenti utili nella programmazione.

Codice del corso:
B024260 (B047) - Laboratorio di Programmazione

- Il Laboratorio di Programmazione consiste prevalentemente nello svolgimento da parte dello studente di un compito didattico aggiuntivo nell'ambito dell'insegnamento Fondamenti di Informatica / Programmazione (C.I.).
- Saranno fatte lezioni in laboratorio per apprendere l'uso di strumenti utili nella programmazione.

Orario

- Mercoledì 13 Marzo: 14:00-18:00 Aule 111+112+113 (ex 112+113+114).
- Mercoledì 3 Aprile: 14:00-18:00 Aule 111+112+113.
- Mercoledì 10 Aprile: 14:00-18:00 Aule 111+112+113.
- Mercoledì 8 Maggio: 14:00-18:00 Aule 111+112+113.

Modalità di svolgimento dell'esame

- L'esame consiste in un elaborato di programmazione in C++ da concordare col docente.
- Durante lo sviluppo dell'elaborato verranno usati gli strumenti visti a lezione, come sistemi di versionamento del codice e unit testing. L'elaborato può essere combinato con quello da svolgere per il corso "Programmazione", previo accordo col docente.
- È possibile sviluppare un proprio progetto, previo accordo col docente
    1. Semplici programmi che implementano funzioni di interfacce grafiche (es. splash screen).
    2. Classi per la lettura e processamento di immagini
    3. Codice per calcolo matriciale.
    4. Interfacce grafiche per videogame.
- Info e link utili sulla pagina web del corso

So You Want to be a Programmer

Qualche consiglio da Peter Norvig

Alcune cose da fare

- **Programmare**. Imparare facendo è il miglior metodo di apprendimento.
- **Leggere codice** (o parlare con altri programmatori).
- Lavorare su **progetti con altri** programmatori.