UNIVERSITÀ
degli STUDI
di CATANIA

Introduzione al corso di programmazione 1

Corso di programmazione I AA 2020/21

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella

Web: http://www.dmi.unict.it/farinella

Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

Overview

Informazioni generali

Descrizione sintetica del corso

Il corso presenta i **fondamenti di programmazione degli elaboratori**. In particolare:

- I concetti di base della **programmazione strutturata**.
- Concetti di base della **programmazione orientata agli oggetti** *OOP (Object-Oriented Programming)*.
  - **C++** è il linguaggio di riferimento.
- Tecniche di progettazione di software OOP.

Syllabus del corso (obiettivi, programma dettagliato, etc)

Propedeuticitá

Nessuna particolare propedeuticità. Tuttavia..

Possedere un laptop è altamente consigliato.

Propedeuticitá

Familiarità con un sistema operativo (Linux o Windows):

- Operazioni di base con i **files** (cancellazione, spostamento, etc)
- Familiarità con un **editor di testo**.
- Compilatore? Impareremo insieme ad usare gcc e dev-c++.

Strumenti per il corso

- **Slide**.
- **Piccoli programmi** da studiare e compilare in classe.
- Esercitazioni alla lavagna.
    - La prima parte dell' esame é costituita da una prova scritta (no elaboratore)! Quindi sarà importante esercitarsi su carta.

La aule dispongono di **prese elettriche** per collegare il proprio laptop

- codificare semplici esempi
- compilare, commentare, studiare, modificare gli esempi forniti dal docente

Contenuti

Programma

- Programma dettagliato (all'interno del syllabus)
- 3 Moduli
  - **Modulo A**: Elementi di Programmazione Imperativa ed Orientata agli Oggetti.
  - **Modulo B**: Caratteristiche avanzate del linguaggio C++.
  - **Modulo C**: Progettazione di software orientato agli oggetti.

Programma: Modulo A

Elementi di Programmazione Imperativa ed Orientata agli Oggetti.

- **Introduzione alla programmazione** (propedeutico alla P1): diagrammi di flusso, algoritmi, notazione NLS, variabili ed array.
- **Introduzione ai linguaggi di programmazione/C++** (propedeutico alla P1): linguaggi di programmazione di basso e alto livello, traduzione vs interpretazione. Editor, compilatore, debugging, esecuzione.
- **Costrutti del linguaggio C++** (P1): tipi di dato, operatori predefiniti, conversioni di tipo, costrutti if-then-else, switch, while e do-while.
- **Introduzione alla Programmazione Orientata agli Oggetti** (propedeutico alla P2): classi e oggetti, loro stato e comportamento, costruttori.
- **Strutture dati array e stringhe di caratteri** (P1): array di tipi predefiniti e di oggetti, array multidimensionali.

Programma: Modulo B

Caratteristiche avanzate del linguaggio C++.

- **Dichiarazione di classi** (P1 e P2): attributi, metodi, modificatori di accesso. Valore di ritorno e passaggio di parametri per valore e per riferimento, categorie di memorizzazione delle variabili, regole di visibilit.
- **Puntatori ed Array** (P1): operatori di indirizzo e di dereferenziazione, aritmetica dei puntatori, puntatori vs array, allocazione dinamica della memoria [...] Array multidimensionali, array allocati dinamicamente.
- **Elementi avanzati del linguaggio** (P2): const per puntatori e metodi, argomenti standard di metodi, clausola Namespace, composizione/aggregazione di oggetti, overloading di metodi, costruttori vs distruttori, costruttore di copia, funzioni e classi friend, attributi e metodi statici.

Progettazione di software orientato agli oggetti

- **Ereditarietà** (P2): Definizione, derivazioni e modalità di accesso, Gerarchie di classi, overriding di metodi, risolutore di scope, lista di inizializzazione, costruttori/distruttori vs ereditatarietà. Ereditariet multipla. Funzioni virtuali e classi astratte.
- **Polimorfismo e Classi astratte** (P2): Puntatori a classi derivate, funzioni virtuali, Late-binding, Polimorfismo. Gerarchie ereditarie di classi polimorfe. Funzioni virtuali pure; Classi astratte; Interfacce. RTTI: typeid e dynamic_cast.
- **Principi di progettazione orientata agli oggetti** (P2): Diagrammi UML per le classi, progettazione OOP
- **Overloading degli operatori** (P2): non-membro, friend, membro. Operatori: prefissi e postfissi "++" e "-". Cenni ad operatori di assegnamento "=", di indicizzazione "[" e "(" e di cast. Operatori di I/O.
- **Programmazione generica** (P2): Funzioni e classi generiche.

Testi consigliati

Testi consigliati

1. **H.M. Deitel, P. J. Deitel, C++ Fondamenti di programmazione 2 Ed.** – Apogeo
   - Rappresenta la traduzione in italiano della **quinta edizione** del testo Inglese (*C++ How to program*).
   - Testo originale (e aggiornato): C++ How to Program 10th edition (2017), aggiornato al C++11 e C++14. (NB: il link punta ad una edizione "economica" in vendita su amazon).
   - *Adatto ai principianti*.
2. Eckel, Thinking in C++, Vol. I, 2nd Edition (disponibile gratuitamente online).
   - *Adatto ai principianti*.
   - NB: Poco aggiornato, ultima edizione anno 2000.
3. **Horstmann, C++ for everyone**, 2Ed. (2010) Wiley (*).
   - *Adatto ai principianti*.

Testi consigliati

4. **Stroustrup, Programming: Principles and Practice Using C++. (Addison-Wesley ISBN 978-0321-992789**
5. **Stroustrup, C++ Linguaggio, libreria standard, principi di programmazione 4Ed**, Addison Wesley.
    - Testo in Inglese aggiornato al 2013: The C++ Programming Language, 4th edition (*)

Esame e voto finale

Esame di Programmazione I

P1 Prova teorica di programmazione imperativa.

- Prova in itinere (**Dicembre 2020**). **Identica a P1**.

⇓

P2 Prova pratica di laboratorio (in genere 7-10 gg dopo la prova scritta).

⇓

P3 Prova teorica (colloquio orale) di programmazione orientata agli oggetti (generalmente 7 gg dopo la prova pratica).

NB: E' consentito sostenere ogni singola prova nell'arco dell'intero Anno Accademico in appelli e/o sessioni separate.

P1. Prova teorica di programmazione imperativa.

SU CARTA

- **Due esercizi differenti**.
- Ogni esercizio prevede la scrittura di un **metodo** in C++ (Esempio 2017, Esempio 2018).
- La prova viene valutata in trentesimi (V1).

P1. Prova teorica di programmazione imperativa.

- La prova scritta può essere **ripetuta** in un qualsiasi appello dello stesso AA, per migliorare la votazione conseguita.
  - **la votazione precedente viene annullata** d'ufficio al momento della consegna del nuovo compito.
- La votazione (V1) si intende **accettata** se lo studente partecipa ad una qualsiasi prova di laboratorio (P2).
  - Ovvero la prova scritta **non può essere ripetuta per migliorare il voto**, fino all'inizio del nuovo anno accademico.

P2. Prova pratica di laboratorio

- **Esercizio** di programmazione orientata agli oggetti.
- **Da sviluppare in C++** all'elaboratore in laboratorio.
- Esempio

P2. Prova pratica di laboratorio

- La prova è giudicata **insufficiente** nei seguenti casi
  - il codice dell'elaborato produce **errori di compilazione**.
  - il codice non produce errori di compilazione ma **nessuno dei punti assegnati** è stato svolto, quindi non produce output.
  - il codice **non produce alcun output esatto** e/o si blocca a causa di errori logici e/o dell'uso della memoria. .
- Prova insufficiente: $V2 = -1$ (**non ammesso** alla prova P3), oppure $V2 = -2$ (**ammesso** alla prova P3).
- Prova sufficiente: $V2 = 0$ oppure $V2 = 1$ oppure $V2 = 2$ (**ammesso alla prova P3**).

P3. Prova teorica di programmazione orientata agli oggetti

- **Colloquio** valutato in trentesimi (V3).
- Verte sui concetti tipici della OOP (esempio: modellazione di software tramite diagrammi UML delle classi)

Il colloquio può anche essere utile a **verificare parti** oggetto delle **precedenti prove P1 e P2**.

PI. Prova in itinere

- La prova è **equivalente alla P1** (Prova teorica di programmazione imperativa).
- Quindi superare la prova in itinere **equivale a superare la prova scritta P1**.

Prova in Itinere AA 20/21: Dicembre 2020

Voto finale.

$$V = \frac{V1+V3}{2} + V2 + P$$

Dove P è la somma dei voti (negativi) relativi alle prove di laboratorio non superate durante l'anno accademico.

Ad esempio..

- $V1 = 24$ (Prova scritta).
- $V2 = +1$ (Prova pratica).
- $P = -1 + (-1) = -2$ (prova di laboratorio insufficiente per due volte).
- $V3 = 28$ (colloquio orale).
- $\Rightarrow \mathbf{V} = \frac{24+28}{2} + 1 - 2 = \mathbf{25}$.

Calendario Esami Programmazione I

NB: Date non ufficiali / da confermare!!

	Appello	Prova scritta	Lab.	Colloquio
1	**Prova in Itinere**	??/12/2020	#	#
2	**I Sessione, I App.**	??/01/2021	?	?
3	**I Sessione, II App.**	??/02/2021	?	?
4	**II Sessione, I App.**	??/06/2021	?	?
5	**II Sessione, II App.**	??/07/2021	?	?
6	**III Sessione, I App.**	??/08/2021	?	?
7	**III Sessione, II App.**	??/09/2021	?	?

Docente, tutor e comunicazioni

Docente

Sito Web Docente:
http://www.dmi.unict.it/farinella
Pagina web del corso di Programmazione 1:
http://www.dmi.unict.it/farinella/Prog1

- Link al **syllabus** del corso.
- **Slide** mostrate a lezione.
- **Esercizi svolti**, **codice** di esempio, etc.
- **Testi di esame** (P1 e P2) assegnati nel corso dell'AA ed eventuali svolgimenti.

Tutor AA 2020/2021

Dr. Francesco Ragusa.

E-mail: francesco.ragusa@.unict.it

Esercitazioni in aula:

- Da Ottobre a Dicembre 2020: esercizi di programmazione strutturata alla lavagna.
- Da Gennaio a Marzo 2021: esercizi di programmazione orientata agli oggetti al calcolatore.

Avvisi e Comunicazioni

Canale Telegram: LINK

FAQ (Frequently Asked Questions)

- **Q1**: *Ho superato la prova scritta del primo appello della prima sessione di esami. Posso sostenere la prova di laboratorio al secondo appello e la prova orale al primo appello della sessione estiva?*
  - **A1**: Certamente. Le tre prove si possono sostenere in appelli differenti nel corso dell'Anno accademico 2020/2021, purchè entro la fine di esso (Dicembre 2020).
- **Q2**: *Ho superato la prova in itinere. Quali altre prove devo sostenere?*
  - **A2**: La prova in itinere equivale ad un qualsiasi esame scritto (P1). Rimangono da sostenere le prove P2 e P3.

FAQ (Frequently Asked Questions)

- **Q3**: *Vorrei sostenere la prova scritta in occasione dello appello X della sessione Y. Devo prenotarmi per l'appello sul portale studenti?*
  - **A3**: SI. Altrimenti, in assenza di prenotazione, nel caso in cui si sostengono con successo le rimanenti prove (P2 e P3) nello stesso appello, non si potrà procedere alla registrazione dell'esame. In questo caso il voto finale sarà "preservato" e registrato successivamente, in occasione del prossimo appello utile, prèvia prenotazione dello studente.
- **Q4**: *Possono recarmi in laboratorio per esercitarmi al calcolatore?*
  - **A4**: Certamente. Laboratorio 236, edificio MII, primo piano. Aperto agli studenti. SO Windows e Linux. Informazioni sui laboratori: LINK

FAQ (Frequently Asked Questions)

- **Q5**: *È obbligatorio acquistare un libro testo per studiare programmazione I?*
  - **A5**: No. Tuttavia è altamente consigliato ai principianti. Per i non principianti un testo può rappresentare un utile riferimento.
- **Q6**: *È obbligatorio possedere un computer/laptop?*
  - **A6**: No. Tuttavia è altamente consigliato perchè la programmazione è come lo sport, si impara e si migliora praticando!! Un laptop vi consente di fare esercizio autonomamente e con maggiore flessibilità (ora, luogo). Inoltre potete portare il laptop a lezione (Nelle aule 1–4 sono presenti i plug di rete elettrica).

FAQ (Frequently Asked Questions)

- **Q7**: *Qual è la funzione del tutor?*
  - **A7**: Il tutor è una figura altamente qualificata ingaggiata per organizzare esercitazioni e, a richiesta, di chiarimento relativamente ad argomenti che risultano "difficili" o poco chiari.
- **Q8**: *Posso scrivere una mail al docente se ho un problema con un programma (errori di compilazione, esecuzione, etc)?*
  - **A8**: E' meglio discutere con il docente di presenza a fine lezione oppure durante le ore di ricevimento.

FAQ (Frequently Asked Questions)

- **Q9**: *Giorno, ora e luogo di ricevimento?*
  - **A9**: Il ricevimento si tiene generalmente il VENERDI POMERIGGIO 15-17.