

# DIPARTIMENTO DI MATEMATICA E INFORMATICA

## Corso di Programmazione II

Alessandro Ortis Image Processing Lab - iplab.dmi.unict.it

alessandro.ortis@unict.it www.dmi.unict.it/ortis/





#### Info&contacts

Alessandro Ortis, PhD

Email: <u>alessandro.ortis@unict.it</u>

Personal webpage: <a href="http://www.dmi.unict.it/ortis/">http://www.dmi.unict.it/ortis/</a>







## IPlab@UNICT





http://iplab.dmi.unict.it

Core Competences: **Computer Vision and Multimedia**Overall Team (~20 – Lead by Prof. S. Battiato)

- Results: >30 patents, >400 papers
- Applications to:
  - Multimedia Forensics and Security
  - EgoCentric Vision
  - Cultural Heritage
  - Social Media Mining
- R&D projects:
  - STM (Catania, Grenoble): Joint R&D lab (since 2007), Ph.D. (>10), grants (>10)
  - Funded projects: ENIAC (1), PO/FESR (4), MISE HZ2020(1), other (4)
- International Events: ICVSS (since 2007), MISS (since 2014) ICIAP 2017, ACIVS 2015, VISAPP





4 FULL PROFESSORS

**4 RESEARCH FELLOWS** 

>20 PHD STUDENTS

4 POST DOCS

4 ASSISTANT PROFESSORS





JOINTOPENLAB

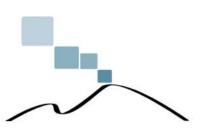
**TIM** 

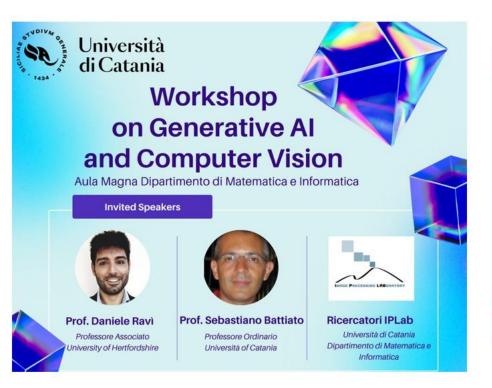














#### I nostri strumenti

#### Strumenti essenziali:

- Compilatore C++ (gcc, MinGW, ...)
- Editor di testo

#### Opzionalmente:

IDE (es., Visual Studio Code, Codeblocks, DevC++, etc.)

Possedere un laptop è altamente consigliato

DevC++ (Windows)

http://www.bloodshed.net/devcpp.html

IDE (Integrated Development Environment) per programmare in C/C++.

- Include il porting Mingw del compilatore GCC (GNU Compiler Collection).
- Rilasciato sotto licenza GPL (GNU General Public License).

Alcune note per l'uso di DevC++:

Selezionare, attraverso il menu a tendina, la versione appropriata del compilatore da usare (32 bit o 64 bit).

Combinazione di tasti CTRL+n per lanciare nuova finestra di editing per singolo file sorgente.

Salvare il file sorgente come <miofile>.cpp, ES: pippo.cpp

Compilazione: F9, Esecuzione del programma: F10

Compila ed esegui in successione: F11

Code::Blocks (Windows, Linux, Apple)

IDE (Integrated Development Environment) per programmare in C/C++ (anche Fortran): <a href="http://www.codeblocks.org">http://www.codeblocks.org</a>

- Multipiattaforma (Linux, Windows, Mac OS)
- Windows: Code::blocks include il porting Mingw del compilatore GCC (GNU Compiler Collection).
- Licenza GPL (GNU General Public License).

È consigliato l'uso di un semplice editor di testo e la compilazione mediante riga di comando.

Esempio di editor: SciTE <a href="https://www.scintilla.org/SciTE.html">https://www.scintilla.org/SciTE.html</a>

- Interfaccia semplice, minimale.
- Syntax highlighting per molti linguaggi, anche C e il C++.
- Dal menu Tool e possibile invocare il compilatore gcc, eseguire successivamente il programma (quindi analogo al tasto F11 di Dev-C++) e quindi e output in un frame dell'applicazione

#### Esempio di compilazione/esecuzione:

```
>> g++ pippo.cpp -o mio_programma
>> ./mioprogramma
```

# Da C a C++ (FAQ)

#### È facile imparare C++ conoscendo C?

Assolutamente si! Un primo passo è programmare in C++ come se si stesse programmando in C, notando le eventuali differenze. Ad esempio, il C++ fornisce un controllo dei tipi più safe.

# Da C a C++ (FAQ)

#### È facile imparare C++ conoscendo C?

Assolutamente si! Un primo passo è programmare in C++ come se si stesse programmando in C, notando le eventuali differenze. Ad esempio, il C++ fornisce un controllo dei tipi più safe.

#### Qual è la differenza tra C e C++?

C++ è un discendente diretto di C, con un controllo sui tipi più forte ed un più ampio spettro di paradigmi di programmazione disponibili (astrazione, OOP, programmazione generica).

# Da C a C++ (FAQ)

#### È facile imparare C++ conoscendo C?

Assolutamente si! Un primo passo è programmare in C++ come se si stesse programmando in C, notando le eventuali differenze. Ad esempio, il C++ fornisce un controllo dei tipi più safe.

#### Qual è la differenza tra C e C++?

C++ è un discendente diretto di C, con un controllo sui tipi più forte ed un più ampio spettro di paradigmi di programmazione disponibili (astrazione, OOP, programmazione generica).

#### Possiamo dire che C è un sottoinsieme di C++?

Non in senso matematico (quindi no!). Infatti ci sono codici validi in C che non sono validi in C++ oppure danno output diversi (vi sfido a trovarli!).

Suggerimenti: utilizzo di void\*, cast implicito, utilizzo delle struct...

```
struct mystruct {};
int main(){
    struct mystruct x;
    printf("%d",(int)sizeof(x));
    return 0;
}
```

### Obiettivi Formativi

- Acquisire metodi di programmazione
  - Imparare a ragionare da informatici
  - Paradigma OOP (Object Oriented Programming)
- Acquisire e sviluppare capacità di
  - Comprendere le proprietà fondamentali di diversi algoritmi e strutture dati per essi
  - Implementare gli algoritmi studiati in C++ in modo da ottenere soluzioni affidabili ed efficienti

Il Syllabus di ciascun corso si trova online:

https://web.dmi.unict.it/corsi/l-31/programmi

## Perché "Efficienti"?

 I computer diventano sempre più rapidi e potenti perché curarsi di scrivere programmi efficienti?

Perché l'efficienza è più importante della potenza

# Tempo $n^5$ (1 operaz. al secondo)

Dimensione dell'input (n)	2	5	10	20	30	40	70
Tempo di	32	52	28	37	9	39	54
calcolo	Secondi	minuti	ore	giorni	mesi	mesi	anni

• Portiamo a 10 op/sec la potenza di calcolo

Dimensione dell'input (n)	2	5	10	20	30	40	70
Tempo di	3,2	5,2	2,8	3,7	28	3,9	5,4
calcolo	Secondi	minuti	ore	giorni	giorni	mesi	anni

# Tempo $n^2$ (1 operaz. al secondo)

Dimensione dell'input (n)	2	5	10	20	30	40	70
Tempo di			100		15	27	82
calcolo	4 sec	25 sec	sec	7 min	min	min	min

 Meglio del metodo precedente implementato su una macchina 10 volte più veloce!

Dimensione	2	5	10	20	30	40	70
dell'input (n)		5	10	20	30	40	70
Tempo di	3,2	5,2	2,8	3,7	28	3,9	5,4
calcolo	Secondi	minuti	ore	giorni	giorni	mesi	anni

- 72 ore di lezione frontale (teoria+pratica)
- L'acquisizione dei metodi e delle competenze professionali della materia è sostenuta da
  - Frequenza delle lezioni (consigliata)
  - Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

## Tutorato 2023-2024

08/02/24	3	15:00-17:00
15/03/24	3	15:00-17:00
22/03/24	3	15:00-17:00
29/03/24	3	15:00-17:00
05/04/24	3	15:00-17:00
12/04/24	3	15:00-17:00
19/04/24	3	15:00-17:00
26/04/24	3	15:00-17:00
03/05/24	3	15:00-17:00
10/05/24	3	15:00-17:00
17/05/24	3	15:00-17:00
24/05/24	3	15:00-17:00
31/05/24	3	15:00-17:00
07/06/24	3	15:00-17:00
14/06/24	3	15:00-17:00
	-	

- 72 ore di lezione frontale (teoria+pratica)
- L'acquisizione dei metodi e delle competenze professionali della materia è sostenuta da
  - Frequenza delle lezioni (consigliata)
  - Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

- 72 ore di lezione frontale (teoria+pratica)
- L'acquisizione dei metodi e delle competenze professionali della materia è sostenuta da
  - Frequenza delle lezioni (consigliata)
  - -Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

- 72 ore di lezione frontale (teoria+pratica)
- L'acquisizione dei metodi e delle competenze professionali della materia è sostenuta da
  - Frequenza delle lezioni (consigliata)
  - -Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

- 72 ore di lezione frontale (teoria+pratica)
- L'acquisizione dei metodi e delle competenze professionali della materia è sostenuta da
  - Frequenza delle lezioni (consigliata)
  - Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

- 72 ore di lezione frontale (teoria+pratica)
  cquisizione dei metodi e delle compere de sessionali della materia è sosten requenza delle lezioni (consigliata)
  - Studio individuale (tanto)
  - Studio del testo
  - Partecipazione al tutorato (?)

#### Testi

Luis Joyanes Aguilar

Fondamenti di Programmazione in C++

McGraw-Hill

T. H Cormen

**Introduction to Algorithms** 

The MIT Press

#### Consigliati:

C++ Tecniche avanzate di programmazione Apogeo

Algoritmi in C++ (terza edizione) Pearson Education Italia

Effective C++ e More Effective C++ Addison-Wesley

## Altri riferimenti utili

- https://cppreference.com
- https://cplusplus.com
- https://www.hackerrank.com/domains/cpp

## Relazione con altri insegnamenti

Propedeuticità richieste: per sostenere l'esame di Programmazione 2, dovrete aver già superato:

- Programmazione 1
- Strutture Discrete

## Programma delle lezioni - I

#### Parte 1: Programmazione ad oggetti

- Classi e Oggetti
- Classi derivate
- Templates
- Sovraccaricamento degli operatori

## Programma delle lezioni - II

#### Parte 2: Cenni di complessità e tecniche ricorsive

- Cenni di complessità asintotica
- Ricorsione ed algoritmi di ordinamento basati sulla ricorsione

## Programma delle lezioni - III

#### **Parte 3: Strutture Dati**

- Liste
- Pile e code
- Alberi
- Grafi

#### Parte 1, 2 e 3:

Tantissimi esercizi

## Kahoot!



https://kahoot.com/

Prova Scritta Pratica Orale

Tipicamente le tre prove avvengono nell'arco della stessa giornata, se il numero di prenotati lo permette, per evitare sovrapposizione con gli appelli di altre materie del primo anno.

La prenotazione è obbligatoria.

Prova Scritta Pratica Orale

#### 1. Prova scritta:

- Durata 1h
- 30 quesiti a risposta multipla
- Tutti i fogli utilizzati vanno consegnati

Prova Scritta Pratica Orale

#### 2. Prova pratica di laboratorio:

- Realizzazione di un codice in C++
- Durata: circa 2 ore
- Compilazione mediate g++ su una macchina generica
- Sarà possibile consultare la documentazione di C++ durante la prova (no Internet!)

Prova Scritta Pratica Orale

#### 3. Prova Orale:

- Compilazione ed esecuzione del progetto
- Quesiti sull'implementazione
- Quesiti su tutto il programma didattico

Il voto finale è il risultato di una valutazione complessiva delle tre prove.

#### Avvisi e comunicazioni

 Pagina degli avvisi del Corso di Laurea in Informatica: <u>LINK</u>

 Pagina degli avvisi dei DOCENTI del Corso di Laurea in Informatica: <u>LINK</u>

## Domande?