

# INFORMATICA @ UNIURB: OFFERTA FORMATIVA 3 + 2

**Prof. Marco Bernardo**

Università degli Studi di Urbino Carlo Bo  
Dipartimento di Scienze Pure e Applicate  
Sezione di Informatica e Matematica

Scuola di Scienza, Ingegneria e Filosofia dell'Informazione



1506  
UNIVERSITÀ  
DEGLI STUDI  
DI URBINO  
CARLO BO



CORSO DI LAUREA TRIENNALE

# INGEGNERIA INFORMATICA E SCIENZE COMPUTAZIONALI

#NOPLACEFORBUGS



CORSO DI LAUREA MAGISTRALE

# INFORMATICA E INNOVAZIONE DIGITALE

#MAKEITDIGITAL

<https://informatica.uniurb.it/>

# Cosa Intendiamo per Informatica?

# Cosa Intendiamo per Informatica?

- **Informatica**: studio dell'elaborazione automatica delle informazioni.
- La gente percepisce solo gli aspetti tecnologici, mentre chi ci lavora deve conoscere anche gli aspetti metodologici e scientifici.

# Cosa Intendiamo per Informatica?

- **Informatica**: studio dell'elaborazione automatica delle informazioni.
- La gente percepisce solo gli aspetti tecnologici, mentre chi ci lavora deve conoscere anche gli aspetti metodologici e scientifici.
- **Aspetti tecnologici** – *Information & Communication Technology*: computer, sistemi operativi, basi di dati, reti di calcolatori, ...

# Cosa Intendiamo per Informatica?

- **Informatica**: studio dell'elaborazione automatica delle informazioni.
- La gente percepisce solo gli aspetti tecnologici, mentre chi ci lavora deve conoscere anche gli aspetti metodologici e scientifici.
- **Aspetti tecnologici** – *Information & Communication Technology*: computer, sistemi operativi, basi di dati, reti di calcolatori, ...
- **Aspetti metodologici** – *Software Architecture & Engineering*: metodologie, linguaggi e ambienti di programmazione, ...

# Cosa Intendiamo per Informatica?

- **Informatica**: studio dell'elaborazione automatica delle informazioni.
- La gente percepisce solo gli aspetti tecnologici, mentre chi ci lavora deve conoscere anche gli aspetti metodologici e scientifici.
- **Aspetti tecnologici** – *Information & Communication Technology*: computer, sistemi operativi, basi di dati, reti di calcolatori, ...
- **Aspetti metodologici** – *Software Architecture & Engineering*: metodologie, linguaggi e ambienti di programmazione, ...
- **Aspetti scientifici** – *Computer Science*: teoria della computazione, algoritmica, teoria degli automi, linguaggi formali, ...

# Cosa Intendiamo per Computer?

# Cosa Intendiamo per Computer?

- **Computer**: insieme di componenti elettromeccaniche **programmabili** per immettere, immagazzinare, elaborare ed emettere **informazioni** sotto forma di numeri, testi, immagini, audio e video.
- Ci sono tanti insiemi di componenti elettromeccaniche che possono eseguire *molteplici* funzioni (elettrodomestici, ascensori, veicoli, ...) ma solo i computer ne possono eseguire un numero potenzialmente *illimitato*, cioè solo i computer sono **macchine programmabili!**

# Cosa Intendiamo per Computer?

- **Computer**: insieme di componenti elettromeccaniche **programmabili** per immettere, immagazzinare, elaborare ed emettere **informazioni** sotto forma di numeri, testi, immagini, audio e video.
- Ci sono tanti insiemi di componenti elettromeccaniche che possono eseguire *molteplici* funzioni (elettrodomestici, ascensori, veicoli, ...) ma solo i computer ne possono eseguire un numero potenzialmente *illimitato*, cioè solo i computer sono **macchine programmabili!**
- **Hardware**: insieme delle componenti elettromeccaniche che formano un computer, cioè le risorse computazionali disponibili (*parti fisiche*).
- **Software**: insieme dei programmi che possono girare su un computer, cioè istruzioni date alle risorse computazionali per eseguire compiti (*parti immateriali* – nome derivante dai telai Jacquard).

# Quali Sono le Basi Teoriche?

- Dato un **problema computazionale**, la **teoria della computazione** studia l'esistenza di una **soluzione algoritmica** e, se questa c'è, quante **risorse computazionali** sono necessarie in termini di:
  - Tempo d'esecuzione.
  - Spazio di memoria.
  - Banda di comunicazione.

# Quali Sono le Basi Teoriche?

- Dato un **problema computazionale**, la **teoria della computazione** studia l'esistenza di una **soluzione algoritmica** e, se questa c'è, quante **risorse computazionali** sono necessarie in termini di:
  - Tempo d'esecuzione.
  - Spazio di memoria.
  - Banda di comunicazione.
- **Teoria della calcolabilità**: **problemi decidibili vs. indecidibili** a seconda dell'esistenza o meno di una soluzione algoritmica.
- **Teoria della complessità**: **problemi (decidibili) trattabili vs. intrattabili** a seconda della quantità polinomiale o esponenziale di risorse.

# Quali Sono le Basi Teoriche?

- Dato un **problema computazionale**, la **teoria della computazione** studia l'esistenza di una **soluzione algoritmica** e, se questa c'è, quante **risorse computazionali** sono necessarie in termini di:
  - Tempo d'esecuzione.
  - Spazio di memoria.
  - Banda di comunicazione.
- **Teoria della calcolabilità**: **problemi decidibili vs. indecidibili** a seconda dell'esistenza o meno di una soluzione algoritmica.
- **Teoria della complessità**: **problemi (decidibili) trattabili vs. intrattabili** a seconda della quantità polinomiale o esponenziale di risorse.
- Se un problema computazionale è decidibile, potrebbero esserci più algoritmi diversi che lo risolvono in modo *più o meno efficiente* (quicksort, mergesort, heapsort vs. insertsort, selectsort, bubblesort).

# Cosa Intendiamo per Algoritmo?

# Cosa Intendiamo per Algoritmo?

- **Algoritmo**: sequenza **finita** di passi, rappresentati in modo da essere intelligibili da un **esecutore**, che risolvono un problema computazionale nella sua **generalità** (cioè per ogni istanza dei suoi dati di ingresso).
- Il matematico persiano **Muhammad ibn Musa al-Khwarizmi** (780-850) fu l'autore di uno dei più antichi trattati di algebra (al-jabr) in cui descrisse come risolvere equazioni lineari e quadratiche.

# Cosa Intendiamo per Algoritmo?

- **Algoritmo**: sequenza **finita** di passi, rappresentati in modo da essere intelligibili da un **esecutore**, che risolvono un problema computazionale nella sua **generalità** (cioè per ogni istanza dei suoi dati di ingresso).
- Il matematico persiano **Muhammad ibn Musa al-Khwarizmi** (780-850) fu l'autore di uno dei più antichi trattati di algebra (al-jabr) in cui descrisse come risolvere equazioni lineari e quadratiche.
- Sebbene la rappresentazione di un algoritmo abbia lunghezza finita, la durata della sua esecuzione può essere illimitata (p.e. cicli).
- L'esecutore non è necessariamente un agente elettromeccanico, può essere un agente biologico o un agente cyberfisico (p.e. ricette).

# Cosa Intendiamo per Algoritmo?

- **Algoritmo**: sequenza **finita** di passi, rappresentati in modo da essere intelligibili da un **esecutore**, che risolvono un problema computazionale nella sua **generalità** (cioè per ogni istanza dei suoi dati di ingresso).
- Il matematico persiano **Muhammad ibn Musa al-Khwarizmi** (780-850) fu l'autore di uno dei più antichi trattati di algebra (al-jabr) in cui descrisse come risolvere equazioni lineari e quadratiche.
- Sebbene la rappresentazione di un algoritmo abbia lunghezza finita, la durata della sua esecuzione può essere illimitata (p.e. cicli).
- L'esecutore non è necessariamente un agente elettromeccanico, può essere un agente biologico o un agente cyberfisico (p.e. ricette).
- **Programma**: algoritmo espresso in un dato ling. di programmazione, il cui esecutore sarà un computer (software, applicazione, codice, ...).

# Quali Sono gli Impatti?

- **Impatto socio-economico** dell'informatica a partire dagli anni 1950:
  - Eseguire calcoli complicati in tempi brevi.
  - Elaborare/recuperare/trasmettere grandi quantità di dati in tempi brevi.
  - Trasferire attività ripetitive dalle persone alle macchine.
  - Trasferire attività complesse dalle persone alle macchine.

# Quali Sono gli Impatti?

- **Impatto socio-economico** dell'informatica a partire dagli anni 1950:
  - Eseguire calcoli complicati in tempi brevi.
  - Elaborare/recuperare/trasmettere grandi quantità di dati in tempi brevi.
  - Trasferire attività ripetitive dalle persone alle macchine.
  - Trasferire attività complesse dalle persone alle macchine.
- L'**invenzione della stampa** nel 1400 iniziò a favorire nel mondo una maggiore **diffusione della conoscenza**.

# Quali Sono gli Impatti?

- **Impatto socio-economico** dell'informatica a partire dagli anni 1950:
  - Eseguire calcoli complicati in tempi brevi.
  - Elaborare/recuperare/trasmettere grandi quantità di dati in tempi brevi.
  - Trasferire attività ripetitive dalle persone alle macchine.
  - Trasferire attività complesse dalle persone alle macchine.
- L'**invenzione della stampa** nel 1400 iniziò a favorire nel mondo una maggiore **diffusione della conoscenza**.
- La **rivoluzione industriale** nel 1700 ampliò le nostre **capacità fisiche** attraverso l'introduzione di **macchine automatiche**.

# Quali Sono gli Impatti?

- **Impatto socio-economico** dell'informatica a partire dagli anni 1950:
  - Eseguire calcoli complicati in tempi brevi.
  - Elaborare/recuperare/trasmettere grandi quantità di dati in tempi brevi.
  - Trasferire attività ripetitive dalle persone alle macchine.
  - Trasferire attività complesse dalle persone alle macchine.
- L'**invenzione della stampa** nel 1400 iniziò a favorire nel mondo una maggiore **diffusione della conoscenza**.
- La **rivoluzione industriale** nel 1700 ampliò le nostre **capacità fisiche** attraverso l'introduzione di **macchine automatiche**.
- La **trasformazione digitale**, basata su elettronica e informatica, sta estendendo in misura crescente le nostre **capacità cognitive** mediante **dispositivi programmabili** come computer e smartphone nonché la rapida propagazione di **dati digitali** via Internet.

- Un altro aspetto socio-economico è dato dal **software open source**.
- Comunità mondiali di persone che sviluppano e mantengono sw.

- Un altro aspetto socio-economico è dato dal **software open source**.
- Comunità mondiali di persone che sviluppano e mantengono sw.
- Essere open source (cioè ispezionabile) è un prerequisito tecnico per essere un **software libero**.
- Free Software Foundation fondata da **Richard Stallman** (1985):
  - Libertà di eseguire un programma per qualsiasi scopo.
  - Libertà di studiare un programma e di modificarlo.
  - Libertà di distribuire copie di un programma a chi ne abbia bisogno.
  - Libertà di migliorare un programma e di distribuirne pubblicamente i miglioramenti in modo tale che chiunque ne possa beneficiare.

- Un altro aspetto socio-economico è dato dal **software open source**.
- Comunità mondiali di persone che sviluppano e mantengono sw.
- Essere open source (cioè ispezionabile) è un prerequisito tecnico per essere un **software libero**.
- Free Software Foundation fondata da **Richard Stallman** (1985):
  - Libertà di eseguire un programma per qualsiasi scopo.
  - Libertà di studiare un programma e di modificarlo.
  - Libertà di distribuire copie di un programma a chi ne abbia bisogno.
  - Libertà di migliorare un programma e di distribuirne pubblicamente i miglioramenti in modo tale che chiunque ne possa beneficiare.
- Esempi: GNU/Linux, Firefox, WordPress, LibreOffice, Moodle, ...
- Distribuzione di **software a pacchetti**.

- Il **pensiero computazionale** è il **contributo culturale** dell'informatica, inteso come processo mentale finalizzato alla risoluzione di problemi.
- Espressione coniata da **Jeannette Wing** (2006).

- Il **pensiero computazionale** è il **contributo culturale** dell'informatica, inteso come processo mentale finalizzato alla risoluzione di problemi.
- Espressione coniata da **Jeannette Wing** (2006).
- Combinazione di:
  - Metodi caratteristici dell'informatica:
    - Analisi **algoritmica** dei problemi.
    - Rappresentazione **digitale** dei dati.
    - Sviluppo di soluzioni **automatizzate**.
  - Capacità intellettuali generali:
    - Affrontare la complessità.
    - Confrontare le alternative.

- Il **pensiero computazionale** è il **contributo culturale** dell'informatica, inteso come processo mentale finalizzato alla risoluzione di problemi.
- Espressione coniata da **Jeannette Wing** (2006).
- Combinazione di:
  - Metodi caratteristici dell'informatica:
    - Analisi **algoritmica** dei problemi.
    - Rappresentazione **digitale** dei dati.
    - Sviluppo di soluzioni **automatizzate**.
  - Capacità intellettuali generali:
    - Affrontare la complessità.
    - Confrontare le alternative.
- Variante computazionale del **pensiero astratto** della matematica:  
*il software è immateriale!*

- Quando inizia la storia dell'informatica?

# Breve Panoramica Storica

- Quando inizia la storia dell'informatica?
- Pionieri tra 1600 e 1800 (**tecnologia meccanica**):
  - **Blaise Pascal** (1623–1662):
    - Macchina meccanica capace di fare addizioni e sottrazioni (1642).

# Breve Panoramica Storica

- Quando inizia la storia dell'informatica?
- Pionieri tra 1600 e 1800 (**tecnologia meccanica**):
  - **Blaise Pascal** (1623–1662):
    - Macchina meccanica capace di fare addizioni e sottrazioni (1642).
  - **Gottfried Wilhelm von Leibniz** (1646–1716):
    - Macchina meccanica comprensiva di moltiplicazioni e divisioni (1672).
    - **Sistema binario**, characteristic universalis, calculus ratiocinator.

# Breve Panoramica Storica

- Quando inizia la storia dell'informatica?
- Pionieri tra 1600 e 1800 (**tecnologia meccanica**):
  - **Blaise Pascal** (1623–1662):
    - Macchina meccanica capace di fare addizioni e sottrazioni (1642).
  - **Gottfried Wilhelm von Leibniz** (1646–1716):
    - Macchina meccanica comprensiva di moltiplicazioni e divisioni (1672).
    - **Sistema binario**, *characteristica universalis*, *calculus ratiocinator*.
  - **Charles Babbage** (1791–1871):
    - Difference Engine (1822): tabulazione di funzioni polinomiali e quindi approssimazione di funzioni logaritmiche e trigonometriche.
    - **Analytical Engine** (1834): archetipo di tutti i moderni computer sia per la sua architettura che per il suo insieme di istruzioni.

# Breve Panoramica Storica

- Quando inizia la storia dell'informatica?
- Pionieri tra 1600 e 1800 (**tecnologia meccanica**):
  - **Blaise Pascal** (1623–1662):
    - Macchina meccanica capace di fare addizioni e sottrazioni (1642).
  - **Gottfried Wilhelm von Leibniz** (1646–1716):
    - Macchina meccanica comprensiva di moltiplicazioni e divisioni (1672).
    - **Sistema binario**, *characteristica universalis*, *calculus ratiocinator*.
  - **Charles Babbage** (1791–1871):
    - Difference Engine (1822): tabulazione di funzioni polinomiali e quindi approssimazione di funzioni logaritmiche e trigonometriche.
    - **Analytical Engine** (1834): archetipo di tutti i moderni computer sia per la sua architettura che per il suo insieme di istruzioni.
  - **Ada Byron Lovelace** (1815–1852):
    - Traduzione del e note sul progetto dell'Analytical Engine (1843).
    - Numeri di Bernoulli: *lei fu la prima persona a programmare nella storia!*

- Alcuni pionieri del 1900 (tecnologia elettronica):
  - Norbert Wiener (1894–1964):
    - Cibernetica: studio interdisciplinare di sistemi naturali e artificiali in termini di controllo, retroazione e comunicazione.

- Alcuni pionieri del 1900 (tecnologia elettronica):
  - **Norbert Wiener** (1894–1964):
    - Cibernetica: studio interdisciplinare di sistemi naturali e artificiali in termini di controllo, retroazione e comunicazione.
  - **Claude Shannon** (1916–2001):
    - Connessione tra circuiti elettrici e algebra di George Boole (1847).
    - Teoria dell'informazione: studio della codifica e della trasmissione di informazioni digitali, la cui unità elementare venne da lui chiamata bit.

- Alcuni pionieri del 1900 (tecnologia elettronica):
  - **Norbert Wiener** (1894–1964):
    - Cibernetica: studio interdisciplinare di sistemi naturali e artificiali in termini di controllo, retroazione e comunicazione.
  - **Claude Shannon** (1916–2001):
    - Connessione tra circuiti elettrici e algebra di George Boole (1847).
    - Teoria dell'informazione: studio della codifica e della trasmissione di informazioni digitali, la cui unità elementare venne da lui chiamata bit.
  - **Alan Turing** (1912–1954):
    - Macchina di Turing (MT): prima descrizione del concetto di algoritmo.
    - Macchina di Turing universale (MTU): visione del software in termini di schema di computazione non più cablato nell'hardware.
    - Crittografia e intelligenza artificiale.

- Alcuni pionieri del 1900 (tecnologia elettronica):
  - **Norbert Wiener** (1894–1964):
    - Cibernetica: studio interdisciplinare di sistemi naturali e artificiali in termini di controllo, retroazione e comunicazione.
  - **Claude Shannon** (1916–2001):
    - Connessione tra circuiti elettrici e algebra di George Boole (1847).
    - Teoria dell'informazione: studio della codifica e della trasmissione di informazioni digitali, la cui unità elementare venne da lui chiamata bit.
  - **Alan Turing** (1912–1954):
    - Macchina di Turing (MT): prima descrizione del concetto di algoritmo.
    - Macchina di Turing universale (MTU): visione del software in termini di schema di computazione non più cablato nell'hardware.
    - Crittografia e intelligenza artificiale.
  - **John Von Neumann** (1903–1957):
    - Dai plugboard computer agli stored program computer (MTU).
    - Dall'aritmetica decimale seriale all'aritmetica binaria parallela.
    - Teoria dei giochi e fisica quantistica.

- Pionieri italiani:
  - **Adriano Olivetti** (1901–1960):
    - **Elea 9003** (1957): *primo computer interamente basato su transistor*, progettato alla Olivetti da **Mario Tchou** (1924–1961).
    - **P101** (1964): *primo personal computer della storia*, progettato alla Olivetti da **Pier Giorgio Perotto** (1930–2002).

- Pionieri italiani:

- **Adriano Olivetti** (1901–1960):

- **Elea 9003** (1957): *primo computer interamente basato su transistor*, progettato alla Olivetti da **Mario Tchou** (1924–1961).
- **P101** (1964): *primo personal computer della storia*, progettato alla Olivetti da **Pier Giorgio Perotto** (1930–2002).

- **CEP** – Calcolatrice Elettronica Pisana (1961):

- Primo computer scientifico progettato in Italia, all'Università di Pisa, un'iniziativa promossa da **Enrico Fermi** e **Adriano Olivetti**.

- Pionieri italiani:

- **Adriano Olivetti** (1901–1960):

- **Elea 9003** (1957): *primo computer interamente basato su transistor*, progettato alla Olivetti da **Mario Tchou** (1924–1961).
    - **P101** (1964): *primo personal computer della storia*, progettato alla Olivetti da **Pier Giorgio Perotto** (1930–2002).

- **CEP** – Calcolatrice Elettronica Pisana (1961):

- Primo computer scientifico progettato in Italia, all'Università di Pisa, un'iniziativa promossa da **Enrico Fermi** e **Adriano Olivetti**.

- **Federico Faggin** (1941):

- Progettò il primo **microprocessore** della storia alla Intel (1971).
    - Sviluppò i primi **touchpad** e **touchscreen** (fine anni 1980).

# Evoluzione dell'Informatica

- Velocità di calcolo e capacità di memoria crescenti come pure costi e dimensioni calanti dell'hw (ENIAC vs. computer moderni):  
tubi a vuoto → transistor → circuiti integrati → VLSI → ...
- Esigenza di infrastrutture/protocolli sw e linguaggi di programmazione per supportare l'evoluzione tecnologica e la produttività dell'utenza.

# Evoluzione dell'Informatica

- Velocità di calcolo e capacità di memoria crescenti come pure costi e dimensioni calanti dell'hw (ENIAC vs. computer moderni):  
tubi a vuoto → transistor → circuiti integrati → VLSI → ...
- Esigenza di infrastrutture/protocolli sw e linguaggi di programmazione per supportare l'evoluzione tecnologica e la produttività dell'utenza.
- Sistemi operativi, basi di dati, reti di calcolatori (1969: Internet).
- Crescente complessità dei sistemi di elaborazione:  
*sequenziali* → *concorrenti* → *distribuiti* → *decentralizzati* → ...

# Evoluzione dell'Informatica

- Velocità di calcolo e capacità di memoria crescenti come pure costi e dimensioni calanti dell'hw (ENIAC vs. computer moderni):  
tubi a vuoto → transistor → circuiti integrati → VLSI → ...
- Esigenza di **infrastrutture/protocolli sw** e **linguaggi di programmazione** per supportare l'evoluzione tecnologica e la produttività dell'utenza.
- Sistemi operativi, basi di dati, reti di calcolatori (1969: Internet).
- Crescente complessità dei sistemi di elaborazione:  
*sequenziali* → *concorrenti* → *distribuiti* → *decentralizzati* → ...
- **Mobile computing**: dispositivi non vincolati a locazioni fisiche (IoT).
- **Global computing**: astrazione di un singolo computer globale accessibile ovunque in ogni momento (cloud, edge, fog).
- **Autonomic computing**: caratteristiche di autogestione per adattarsi a cambiamenti imprevisti (sensori, attuatori, politiche, AI).

- Linguaggi di programmazione:
  - *(Imperativi) procedurali*: Fortran, Cobol, Algol, Basic, Pascal, C, ...
  - *(Imperativi) a oggetti*: Simula, Smalltalk, C++, Java, C#, ...
  - *Dichiarativi (funzionali/logici)*: Lisp, Scheme, ML, Haskell, Prolog, ...
  - *Concorrenti*: Modula, Ada, Occam, Erlang, Scala, ...

- Linguaggi di programmazione:
  - *(Imperativi) procedurali*: Fortran, Cobol, Algol, Basic, Pascal, C, ...
  - *(Imperativi) a oggetti*: Simula, Smalltalk, C++, Java, C#, ...
  - *Dichiarativi (funzionali/logici)*: Lisp, Scheme, ML, Haskell, Prolog, ...
  - *Concorrenti*: Modula, Ada, Occam, Erlang, Scala, ...
- Altri linguaggi:
  - *Script*: shell, Perl/Raku, Tcl/Tk, Python, PHP, JavaScript, Ruby, ...
  - *Interrogazione*: SQL, ...
  - *Markup*: HTML, XML,  $\text{\LaTeX}$ , ...
  - *Modellazione*: UML, AADL, ...
  - *Smart contract*: Solidity, Move, Vyper, ...

- Linguaggi di programmazione:
  - (*Imperativi*) *procedurali*: Fortran, Cobol, Algol, Basic, Pascal, C, ...
  - (*Imperativi*) *a oggetti*: Simula, Smalltalk, C++, Java, C#, ...
  - *Dichiarativi (funzionali/logici)*: Lisp, Scheme, ML, Haskell, Prolog, ...
  - *Concorrenti*: Modula, Ada, Occam, Erlang, Scala, ...
- Altri linguaggi:
  - *Script*: shell, Perl/Raku, Tcl/Tk, Python, PHP, JavaScript, Ruby, ...
  - *Interrogazione*: SQL, ...
  - *Markup*: HTML, XML,  $\text{\LaTeX}$ , ...
  - *Modellazione*: UML, AADL, ...
  - *Smart contract*: Solidity, Move, Vyper, ...
- Supporto per **quantum computing** (esecuzione più veloce).
- Supporto per **reversible computing** (minore consumo energetico).

- Ibridazione con molte altre discipline:

- Matematica **computazionale**.
- Fisica **computazionale**.
- Chimica **computazionale**.
- Biologia **computazionale**.
- **Informatica** giuridica.
- **Informatica** umanistica.

- Ibridazione con molte altre discipline:
  - Matematica **computazionale**.
  - Fisica **computazionale**.
  - Chimica **computazionale**.
  - Biologia **computazionale**.
  - **Informatica** giuridica.
  - **Informatica** umanistica.
- Trasformazione del mondo economico, finanziario e monetario:
  - Finanza decentralizzata e **smart contract**.
  - Beni mobili e immobili trasformati in **token digitali**.
  - Criptovalute, stablecoin e **central bank digital currency**.

- Ibridazione con molte altre discipline:
  - Matematica **computazionale**.
  - Fisica **computazionale**.
  - Chimica **computazionale**.
  - Biologia **computazionale**.
  - **Informatica** giuridica.
  - **Informatica** umanistica.
- Trasformazione del mondo economico, finanziario e monetario:
  - Finanza decentralizzata e **smart contract**.
  - Beni mobili e immobili trasformati in **token digitali**.
  - Criptovalute, stablecoin e **central bank digital currency**.
- Intelligenza artificiale: **risposta giusta o risposta più probabile?**
  - Reti neurali.
  - Machine learning.
  - Deep learning.
  - Big data.
  - Digital twins.

# Perché Teoria e Metodologia? Disastri Informatici

- Le tecnologie dell'informazione e della comunicazione sono *pervasive*.
- **Malfunzionamenti, inefficienze, falle di sicurezza, interfacce scadenti:** insoddisfazione dell'utenza e maggiori costi nella produzione sw.
- Perdite di vite umane e disastri ambientali se il sw è safety-critical: missioni spaziali, traffico aereo, centrali nucleari, ...
- [https://en.wikipedia.org/wiki/List\\_of\\_software\\_bugs](https://en.wikipedia.org/wiki/List_of_software_bugs)

# Perché Teoria e Metodologia? Disastri Informatici

- Le tecnologie dell'informazione e della comunicazione sono *pervasive*.
- **Malfunzionamenti, inefficienze, falle di sicurezza, interfacce scadenti:** insoddisfazione dell'utenza e maggiori costi nella produzione sw.
- Perdite di vite umane e disastri ambientali se il sw è safety-critical: missioni spaziali, traffico aereo, centrali nucleari, ...
- [https://en.wikipedia.org/wiki/List\\_of\\_software\\_bugs](https://en.wikipedia.org/wiki/List_of_software_bugs)
- **Metodologie** per guidare progettazione, sviluppo e dispiegamento sw.
- Il testing del software non garantisce l'assenza di errori.
- È necessaria la **verifica del software**:
  - **Annotazione del programma** (logica di Floyd-Hoare, separation logic) e **verifica deduttiva** (precondizioni-postcondizioni, invarianti).
  - **Modello del programma**, formalizzazione delle proprietà (logica modale, logica temporale) e **model ckecking** (più generazione di controesempi).



1506  
UNIVERSITÀ  
DEGLI STUDI  
DI URBINO  
CARLO BO



CORSO DI LAUREA TRIENNALE

# INGEGNERIA INFORMATICA E SCIENZE COMPUTAZIONALI

#NOPLACEFORBUGS



CORSO DI LAUREA MAGISTRALE

# INFORMATICA E INNOVAZIONE DIGITALE

#MAKEITDIGITAL

<https://informatica.uniurb.it/>

- Il corso di laurea **triennale** in **Informatica Applicata** di UniUrb venne attivato nell'a.a. **2001/2002** come “progetto di Ateneo con il coinvolgimento e la partecipazione di diverse facoltà”.

- Il corso di laurea **triennale** in **Informatica Applicata** di UniUrb venne attivato nell'a.a. **2001/2002** come “progetto di Ateneo con il coinvolgimento e la partecipazione di diverse facoltà”.
- Il Rettore Carlo Bo nominò nel 2000 una commissione incaricata di:
  - “fornire indicazioni sulla organizzazione delle attività didattiche”;
  - “suggerire i raccordi fondamentali con realtà produttive e di ricerca”.
- Presieduta dal Preside Mauro Magnani e composta da docenti di UniBo (Lorenzo Donatiello, Bruno Riccò) e PoliTo (Enrico Macii), oltre che dal referente EMEA di Microsoft (Umberto Paolucci).

- Il corso di laurea **triennale** in **Informatica Applicata** di UniUrb venne attivato nell'a.a. **2001/2002** come “progetto di Ateneo con il coinvolgimento e la partecipazione di diverse facoltà”.
- Il Rettore Carlo Bo nominò nel 2000 una commissione incaricata di:
  - “fornire indicazioni sulla organizzazione delle attività didattiche”;
  - “suggerire i raccordi fondamentali con realtà produttive e di ricerca”.
- Presieduta dal Preside Mauro Magnani e composta da docenti di UniBo (Lorenzo Donatiello, Bruno Riccò) e PoliTo (Enrico Macii), oltre che dal referente EMEA di Microsoft (Umberto Paolucci).
- L'aggettivo **Applicata** voleva denotare la sinergia tra più ambiti:
  - **Scienze informatiche** (aspetti teorici e metodologici).
  - **Ingegneria dell'informazione** (aspetti sistemistici e applicativi).
  - **Ingegneria elettronica**.

- Il corso di laurea **magistrale** fu poi attivato nell'a.a. **2020/2021**, sempre con l'epigrafe **Informatica Applicata**.

- Il corso di laurea **magistrale** fu poi attivato nell'a.a. **2020/2021**, sempre con l'epigrafe **Informatica Applicata**.
- Nuove epigrafi e immagine dall'a.a. **2023/2024**:
  - **Informatica – Scienza e Tecnologia**.
  - **Informatica e Innovazione Digitale**.

- Il corso di laurea **magistrale** fu poi attivato nell'a.a. **2020/2021**, sempre con l'epigrafe **Informatica Applicata**.
- Nuove epigrafi e immagine dall'a.a. **2023/2024**:
  - **Informatica – Scienza e Tecnologia**.
  - **Informatica e Innovazione Digitale**.
- Finalmente allineamento di forma e sostanza dall'a.a. **2026/2027**:
  - **Ingegneria Informatica e Scienze Computazionali**.
  - **Informatica e Innovazione Digitale**.
- Fin dall'inizio il piano degli studi e il corpo docente hanno coperto sia gli aspetti scientifici che quelli ingegneristici dell'informatica.

- Primo anno:
  - Programmazione Procedurale.
  - Algoritmi e Strutture Dati.
  - Reti Logiche.
  - Architettura degli Elaboratori.

- Primo anno:
  - Programmazione Procedurale.
  - Algoritmi e Strutture Dati.
  - Reti Logiche.
  - Architettura degli Elaboratori.
- Secondo anno:
  - Programmazione e Modellazione a Oggetti.
  - Ingegneria e Architettura del Software.
  - Sistemi Operativi.

- Primo anno:
  - Programmazione Procedurale.
  - Algoritmi e Strutture Dati.
  - Reti Logiche.
  - Architettura degli Elaboratori.
- Secondo anno:
  - Programmazione e Modellazione a Oggetti.
  - Ingegneria e Architettura del Software.
  - Sistemi Operativi.
- Terzo anno:
  - Programmazione Logica e Funzionale.
  - Linguaggi di Programmazione e Verifica del Software.
  - Basi di Dati.
  - Reti di Calcolatori.

# Piano degli Studi della Triennale

- Primo anno:
  - Programmazione Procedurale.
  - Algoritmi e Strutture Dati.
  - Reti Logiche.
  - Architettura degli Elaboratori.
- Secondo anno:
  - Programmazione e Modellazione a Oggetti.
  - Ingegneria e Architettura del Software.
  - Sistemi Operativi.
- Terzo anno:
  - Programmazione Logica e Funzionale.
  - Linguaggi di Programmazione e Verifica del Software.
  - Basi di Dati.
  - Reti di Calcolatori.
- Pilastro **programmazione software** + pilastro **sistemi di elaborazione**.

- Linguaggi di programmazione insegnati nel corso dei tre anni:
  - C  $\rightsquigarrow$  linguaggio imperativo procedurale.
  - Java, C++, C#, Python  $\rightsquigarrow$  linguaggi imperativi a oggetti.
  - Haskell  $\rightsquigarrow$  linguaggio dichiarativo funzionale.
  - Prolog  $\rightsquigarrow$  linguaggio dichiarativo logico.
- Altri linguaggi insegnati:
  - UML  $\rightsquigarrow$  linguaggio di modellazione di software a oggetti.
  - SQL  $\rightsquigarrow$  linguaggio di interrogazione di basi di dati.

- Linguaggi di programmazione insegnati nel corso dei tre anni:
  - C  $\leadsto$  linguaggio imperativo procedurale.
  - Java, C++, C#, Python  $\leadsto$  linguaggi imperativi a oggetti.
  - Haskell  $\leadsto$  linguaggio dichiarativo funzionale.
  - Prolog  $\leadsto$  linguaggio dichiarativo logico.
- Altri linguaggi insegnati:
  - UML  $\leadsto$  linguaggio di modellazione di software a oggetti.
  - SQL  $\leadsto$  linguaggio di interrogazione di basi di dati.
- Ogni insegnamento di area informatica si compone di due parti:
  - Una parte **teorica**, in aule dotate di computer con videoproiettore, lavagna di ardesia con webcam fissa, touch board con smart webcam.
  - Una parte **pratica**, in un laboratorio dotato di personal computer coi sistemi operativi **Linux** (open source) e **Windows** (proprietario).
- L'esame comprende un progetto, una prova scritta e una prova orale.

- Formazione matematico-fisica:
  - Logica, Algebra e Geometria (I).
  - Analisi Matematica 1 (I).
  - Analisi Matematica 2 (II).
  - Probabilità e Statistica Matematica (II).
  - Fisica Generale (II).

- Formazione matematico-fisica:
  - Logica, Algebra e Geometria (I).
  - Analisi Matematica 1 (I).
  - Analisi Matematica 2 (II).
  - Probabilità e Statistica Matematica (II).
  - Fisica Generale (II).
- Elettronica (II):
  - Elaborazione Numerica dei Segnali.
  - Sistemi per l'Internet of Things.

- Formazione matematico-fisica:
  - Logica, Algebra e Geometria (I).
  - Analisi Matematica 1 (I).
  - Analisi Matematica 2 (II).
  - Probabilità e Statistica Matematica (II).
  - Fisica Generale (II).
- Elettronica (II):
  - Elaborazione Numerica dei Segnali.
  - Sistemi per l'Internet of Things.
- Ambiti tra cui scegliere (III):
  - Geo-tecnologico: Tecnologie Web + Geomatica.
  - Quantistico-simulativo: Fisica e QC + Simulazione Numerica.
  - Media digitali: Internet Studies + Sociologia della Comunicazione.
  - Giuridico-sociale: Informatica Giuridica + Analisi delle Reti Sociali.

- Formazione matematico-fisica:
  - Logica, Algebra e Geometria (I).
  - Analisi Matematica 1 (I).
  - Analisi Matematica 2 (II).
  - Probabilità e Statistica Matematica (II).
  - Fisica Generale (II).
- Elettronica (II):
  - Elaborazione Numerica dei Segnali.
  - Sistemi per l'Internet of Things.
- Ambiti tra cui scegliere (III):
  - Geo-tecnologico: Tecnologie Web + Geomatica.
  - Quantistico-simulativo: Fisica e QC + Simulazione Numerica.
  - Media digitali: Internet Studies + Sociologia della Comunicazione.
  - Giuridico-sociale: Informatica Giuridica + Analisi delle Reti Sociali.
- Lingua Inglese B1, insegnamenti a scelta, seminari/tirocini/stage, tesi.

# All'Inizio del Percorso Triennale

- L'immatricolazione avviene in modalità telematica.
- L'iscrizione alla triennale **richiede** il sostenimento di una prova di verifica della preparazione iniziale (VPI) attraverso **TOLC-S** o **TOLC-I**.
- A inizio settembre vengono organizzati dei **precorsi di matematica** sia per preparare alla prova di VPI che per colmare eventuali lacune.

# All'Inizio del Percorso Triennale

- L'immatricolazione avviene in modalità telematica.
- L'iscrizione alla triennale **richiede** il sostenimento di una prova di verifica della preparazione iniziale (VPI) attraverso TOLC-S o TOLC-I.
- A inizio settembre vengono organizzati dei **precorsi di matematica** sia per preparare alla prova di VPI che per colmare eventuali lacune.
- Se la prova viene sostenuta ma non superata nei tempi previsti, vengono attribuiti degli obblighi formativi aggiuntivi (OFA).
- Chi ha OFA da assolvere **non** può sostenere gli esami di insegnamenti del secondo e terzo anno.

# All'Inizio del Percorso Triennale

- L'immatricolazione avviene in modalità telematica.
- L'iscrizione alla triennale **richiede** il sostenimento di una prova di verifica della preparazione iniziale (VPI) attraverso **TOLC-S** o **TOLC-I**.
- A inizio settembre vengono organizzati dei **precorsi di matematica** sia per preparare alla prova di VPI che per colmare eventuali lacune.
- Se la prova viene sostenuta ma non superata nei tempi previsti, vengono attribuiti degli obblighi formativi aggiuntivi (**OFA**).
- Chi ha OFA da assolvere **non** può sostenere gli esami di insegnamenti del secondo e terzo anno.
- Il percorso triennale prevede l'acquisizione di 180 CFU, dove **1 CFU = 25 ore** di attività da parte di chi studia (frequenza lezioni, studio individuale, tirocinio/stage, tesi).

# Durante il Percorso Triennale

- Le lezioni degli insegnamenti di area informatica del primo anno non presuppongono il possesso di conoscenze pregresse ...
- ... ma è fondamentale studiare con regolarità.
- La frequenza a lezioni ed esercitazioni è fortemente consigliata.

# Durante il Percorso Triennale

- Le lezioni degli insegnamenti di area informatica del primo anno **non presuppongono il possesso di conoscenze pregresse ...**
- ... ma è fondamentale **studiare con regolarità**.
- La **frequenza** a lezioni ed esercitazioni è fortemente **consigliata**.
- Due periodi didattici:
  - Primo semestre: da metà settembre a fine dicembre (14 settimane).
  - Secondo semestre: da metà febbraio a fine maggio (14 settimane).
- Tre sessioni d'esame:
  - Sessione invernale: da inizio gennaio a metà febbraio (6 settimane).
  - Sessione estiva: da inizio giugno a metà luglio (6 settimane).
  - Sessione autunnale: da fine agosto a metà settembre (4 settimane).
- Seminari tenuti da figure esperte e stage presso aziende o enti.

- La laurea triennale consente l'accesso:
  - alle **lauree magistrali in Inf./Ing. Inf.** col riconoscimento dei 180 CFU;
  - all'Esame di Stato per l'iscrizione all'**Albo degli Ingegneri Juniores**;
  - al **mondo del lavoro** con ottime prospettive (fonte AlmaLaurea).

# Alla Fine del Percorso Triennale

- La laurea triennale consente l'accesso:
  - alle **lauree magistrali in Inf./Ing. Inf.** col riconoscimento dei 180 CFU;
  - all'Esame di Stato per l'iscrizione all'**Albo degli Ingegneri Juniores**;
  - al **mondo del lavoro** con ottime prospettive (fonte AlmaLaurea).
- Profili professionali (classificazione Istat):
  - Tecnici programmatori.
  - Tecnici esperti in applicazioni.
  - Tecnici web.
  - Tecnici gestori di basi di dati.
  - Tecnici gestori di reti e di sistemi telematici.
- Sbocchi occupazionali:
  - Libera professione nel settore ICT.
  - Società di consulenza, progettazione, fornitura di servizi digitali.
  - Aziende e organizzazioni pubbliche e private con esigenze informatiche.

- Primo anno:
  - Programmazione di Dispositivi Mobili e Interfacce Utente.
  - Sistemi Distribuiti e Decentralizzati.
  - Sicurezza Informatica.
  - Machine Learning.

- Primo anno:
  - Programmazione di Dispositivi Mobili e Interfacce Utente.
  - Sistemi Distribuiti e Decentralizzati.
  - Sicurezza Informatica.
  - Machine Learning.
- Secondo anno:
  - Programmazione per l'Internet of Things.
  - Applicazioni Distribuite e Cloud Computing.

# Percorso Biennale della Laurea Magistrale

- Primo anno:
  - Programmazione di Dispositivi Mobili e Interfacce Utente.
  - Sistemi Distribuiti e Decentralizzati.
  - Sicurezza Informatica.
  - Machine Learning.
- Secondo anno:
  - Programmazione per l'Internet of Things.
  - Applicazioni Distribuite e Cloud Computing.
- Formazione matematico-fisica:
  - Metodi Numerici per l'Algebra Lineare e l'Analisi Funzionale (I).
  - Elaborazione dei Dati Sperimentali (II).

# Percorso Biennale della Laurea Magistrale

- Primo anno:
  - Programmazione di Dispositivi Mobili e Interfacce Utente.
  - Sistemi Distribuiti e Decentralizzati.
  - Sicurezza Informatica.
  - Machine Learning.
- Secondo anno:
  - Programmazione per l'Internet of Things.
  - Applicazioni Distribuite e Cloud Computing.
- Formazione matematico-fisica:
  - Metodi Numerici per l'Algebra Lineare e l'Analisi Funzionale (I).
  - Elaborazione dei Dati Sperimentali (II).
- Curricula tra cui scegliere (I, II):
  - Intelligenza Artificiale: Principi + Deep Learning.
  - Analisi Sociologica delle Tecnologie Digitali: Reti Sociali + Etica Dig.

# Percorso Biennale della Laurea Magistrale

- Primo anno:
  - Programmazione di Dispositivi Mobili e Interfacce Utente.
  - Sistemi Distribuiti e Decentralizzati.
  - Sicurezza Informatica.
  - Machine Learning.
- Secondo anno:
  - Programmazione per l'Internet of Things.
  - Applicazioni Distribuite e Cloud Computing.
- Formazione matematico-fisica:
  - Metodi Numerici per l'Algebra Lineare e l'Analisi Funzionale (I).
  - Elaborazione dei Dati Sperimentali (II).
- Curricula tra cui scegliere (I, II):
  - Intelligenza Artificiale: Principi + Deep Learning.
  - Analisi Sociologica delle Tecnologie Digitali: Reti Sociali + Etica Dig.
- Lingua Inglese B2, insegnamenti a scelta, tirocini/stage, tesi.

- Direzioni strategiche ed esigenze socio-economiche:
  - Digitalizzazione e dematerializzazione.
  - Crescita del mercato digitale.
  - Industrie e imprese innovative.
  - AI per la green economy.
  - App per smart device.
- Didattica erogata in **modalità mista simultanea presenza-distanza** per venire incontro alle esigenze di chi già lavora.

- Direzioni strategiche ed esigenze socio-economiche:
  - Digitalizzazione e dematerializzazione.
  - Crescita del mercato digitale.
  - Industrie e imprese innovative.
  - AI per la green economy.
  - App per smart device.
- Didattica erogata in **modalità mista simultanea presenza-distanza** per venire incontro alle esigenze di chi già lavora.
- La laurea magistrale consente l'accesso:
  - ai **dottorati di ricerca in Informatica**;
  - all'Esame di Stato per l'iscrizione all'**Albo degli Ingegneri Seniores**;
  - al **mondo del lavoro** con ottime prospettive (fonte AlmaLaurea).
- Profili professionali:
  - Mobile/IoT system and application designer/developer.
  - Security/Network specialist.
  - Big data analyst.

- Identità digitale per l'utilizzo di tutti i servizi telematici.
- Posta elettronica, cloud storage, webconference (Google suite).
- WiFi disponibile in tutti gli edifici dell'Ateneo.
- Gestione dematerializzata della carriera studentesca.
- Piattaforma Moodle per materiale didattico, prove di autovalutazione e forum di discussione docenti-studenti (blended learning).

# Servizi e Opportunità di UniUrb

- Identità digitale per l'utilizzo di tutti i servizi telematici.
- Posta elettronica, cloud storage, webconference (Google suite).
- WiFi disponibile in tutti gli edifici dell'Ateneo.
- Gestione dematerializzata della carriera studentesca.
- Piattaforma Moodle per materiale didattico, prove di autovalutazione e forum di discussione docenti-studenti (blended learning).
- Competenze trasversali attraverso corsi erogati dal CISDEL – Centro Integrato Servizi Didattici ed E-Learning di UniUrb.
- Servizi linguistici (CLA), sportivi, di tutorato, di supporto psicologico.
- Servizi di ristorazione e alloggio gestiti da ERDiS.
- Borse di studio e agevolazioni tasse di iscrizione.



1506  
UNIVERSITÀ  
DEGLI STUDI  
DI URBINO  
CARLO BO



CORSO DI LAUREA TRIENNALE

# INGEGNERIA INFORMATICA E SCIENZE COMPUTAZIONALI

#NOPLACEFORBUGS



CORSO DI LAUREA MAGISTRALE

# INFORMATICA E INNOVAZIONE DIGITALE

#MAKEITDIGITAL

<https://informatica.uniurb.it/>

# Programmi Ricorsivi e Principio di Induzione

- Un problema complesso viene suddiviso in sottoproblemi più semplici così la soluzione viene ottenuta combinando quelle dei sottoproblemi.

# Programmi Ricorsivi e Principio di Induzione

- Un problema complesso viene suddiviso in sottoproblemi più semplici così la soluzione viene ottenuta combinando quelle dei sottoproblemi.
- Quando i sottoproblemi sono della *stessa natura del problema originale* si può adottare uno schema di soluzione **ricorsivo**:
  - Individuare uno o più **casi base**, per i quali si può ricavare **direttamente** la soluzione del problema.
  - Definire uno o più **casi generali** attraverso un insieme di sottoproblemi della stessa natura di quello originale ma **più vicini ai casi base**.

# Programmi Ricorsivi e Principio di Induzione

- Un problema complesso viene suddiviso in sottoproblemi più semplici così la soluzione viene ottenuta combinando quelle dei sottoproblemi.
- Quando i sottoproblemi sono della *stessa natura del problema originale* si può adottare uno schema di soluzione **ricorsivo**:
  - Individuare uno o più **casi base**, per i quali si può ricavare **direttamente** la soluzione del problema.
  - Definire uno o più **casi generali** attraverso un insieme di sottoproblemi della stessa natura di quello originale ma **più vicini ai casi base**.
- La ricorsione è uno strumento molto potente:
  - Affrontare problemi altrimenti ingestibili (problema delle torri di Hanoi).
  - Risolvere problemi in maniera più efficiente (mergesort e quicksort).
  - Definire in modo naturale certe strutture dati (liste e alberi).
- La sua base teorica è il **principio di induzione**.

- $\mathbb{N}$  è **infinito** e quindi non può essere definito elencandone gli elementi.
- Definizione assiomatica di Dedekind-Peano in teoria degli insiemi:
  - 1 Esiste un elemento  $0 \in \mathbb{N}$ .
  - 2 Esiste una funzione totale  $succ : \mathbb{N} \rightarrow \mathbb{N}$ .
  - 3 Per ogni  $n \in \mathbb{N}$ ,  $succ(n) \neq 0$ .
  - 4 Per ogni  $n, n' \in \mathbb{N}$ , se  $n \neq n'$  allora  $succ(n) \neq succ(n')$ .
  - 5 Se  $M$  è un sottoinsieme di  $\mathbb{N}$  tale che:
    - $0 \in M$ ;
    - per ogni  $n \in \mathbb{N}$ ,  $n \in M$  implica  $succ(n) \in M$ ;allora  $M = \mathbb{N}$  (quindi  $\mathbb{N}$  è il più piccolo insieme chiuso su  $0$  e  $succ$ ).

- $\mathbb{N}$  è **infinito** e quindi non può essere definito elencandone gli elementi.
- Definizione assiomatica di Dedekind-Peano in teoria degli insiemi:
  - 1 Esiste un elemento  $0 \in \mathbb{N}$ .
  - 2 Esiste una funzione totale  $succ : \mathbb{N} \rightarrow \mathbb{N}$ .
  - 3 Per ogni  $n \in \mathbb{N}$ ,  $succ(n) \neq 0$ .
  - 4 Per ogni  $n, n' \in \mathbb{N}$ , se  $n \neq n'$  allora  $succ(n) \neq succ(n')$ .
  - 5 Se  $M$  è un sottoinsieme di  $\mathbb{N}$  tale che:
    - $0 \in M$ ;
    - per ogni  $n \in \mathbb{N}$ ,  $n \in M$  implica  $succ(n) \in M$ ;
 allora  $M = \mathbb{N}$  (quindi  $\mathbb{N}$  è il più piccolo insieme chiuso su  $0$  e  $succ$ ).
- Gli elementi di  $\mathbb{N}$  sono dunque  $0$ ,  $succ(0)$ ,  $succ(succ(0))$ ,  $\dots$  dove  $succ(0)$  è denotato da  $1$ ,  $succ(succ(0))$  è denotato da  $2$ , e così via.
- La sequenza finita di simboli della defin. genera un insieme infinito.

- L'ultimo assioma è il **principio di induzione**, uno dei più potenti strumenti della matematica discreta per definizioni e dimostrazioni.
- Definizione formale dell'intera aritmetica: **aritmetica di Peano**.

- L'ultimo assioma è il **principio di induzione**, uno dei più potenti strumenti della matematica discreta per definizioni e dimostrazioni.
- Definizione formale dell'intera aritmetica: **aritmetica di Peano**.
- Sia  $pred : \mathbb{N}_{\neq 0} \rightarrow \mathbb{N}$  tale che  $pred(succ(n)) = n$  per ogni  $n \in \mathbb{N}$  e  $succ(pred(n)) = n$  per ogni  $n \in \mathbb{N}_{\neq 0}$ .
- Definizione formale dell'addizione:

$$m \oplus n = \begin{cases} m & \text{se } n = 0 \\ succ(m) \oplus pred(n) & \text{se } n \neq 0 \end{cases}$$

- Esempio:  $5 \oplus 2 = 6 \oplus 1 = 7 \oplus 0 = 7$ .

- L'ultimo assioma è il **principio di induzione**, uno dei più potenti strumenti della matematica discreta per definizioni e dimostrazioni.
- Definizione formale dell'intera aritmetica: **aritmetica di Peano**.
- Sia  $pred : \mathbb{N}_{\neq 0} \rightarrow \mathbb{N}$  tale che  $pred(succ(n)) = n$  per ogni  $n \in \mathbb{N}$  e  $succ(pred(n)) = n$  per ogni  $n \in \mathbb{N}_{\neq 0}$ .
- Definizione formale dell'addizione:

$$m \oplus n = \begin{cases} m & \text{se } n = 0 \\ succ(m) \oplus pred(n) & \text{se } n \neq 0 \end{cases}$$

- Esempio:  $5 \oplus 2 = 6 \oplus 1 = 7 \oplus 0 = 7$ .
- Definizione formale delle relazioni d'ordine:
  - $m \leq n$  se e solo se esiste  $m' \in \mathbb{N}$  tale che  $m \oplus m' = n$ .
  - $m < n$  se e solo se  $m \leq n$  con  $m \neq n$ .
  - $m \geq n$  se e solo se  $n \leq m$  ed  $m > n$  se e solo se  $n < m$ .

- Definizione formale della sottrazione ( $m \geq n$ ):

$$m \ominus n = \begin{cases} m & \text{se } n = 0 \\ \text{pred}(m) \ominus \text{pred}(n) & \text{se } n > 0 \end{cases}$$

- Definizione formale della sottrazione ( $m \geq n$ ):

$$m \ominus n = \begin{cases} m & \text{se } n = 0 \\ \text{pred}(m) \ominus \text{pred}(n) & \text{se } n > 0 \end{cases}$$

- Definizione formale della moltiplicazione:

$$m \otimes n = \begin{cases} 0 & \text{se } n = 0 \\ m \oplus (m \otimes \text{pred}(n)) & \text{se } n > 0 \end{cases}$$

- Definizione formale della sottrazione ( $m \geq n$ ):

$$m \ominus n = \begin{cases} m & \text{se } n = 0 \\ \text{pred}(m) \ominus \text{pred}(n) & \text{se } n > 0 \end{cases}$$

- Definizione formale della moltiplicazione:

$$m \otimes n = \begin{cases} 0 & \text{se } n = 0 \\ m \oplus (m \otimes \text{pred}(n)) & \text{se } n > 0 \end{cases}$$

- Definizione formale della divisione ( $n \neq 0$ ):

$$m \oslash n = \begin{cases} 0 & \text{se } m < n \\ \text{succ}((m \ominus n) \oslash n) & \text{se } m \geq n \end{cases}$$

- Definizione formale della sottrazione ( $m \geq n$ ):

$$m \ominus n = \begin{cases} m & \text{se } n = 0 \\ \text{pred}(m) \ominus \text{pred}(n) & \text{se } n > 0 \end{cases}$$

- Definizione formale della moltiplicazione:

$$m \otimes n = \begin{cases} 0 & \text{se } n = 0 \\ m \oplus (m \otimes \text{pred}(n)) & \text{se } n > 0 \end{cases}$$

- Definizione formale della divisione ( $n \neq 0$ ):

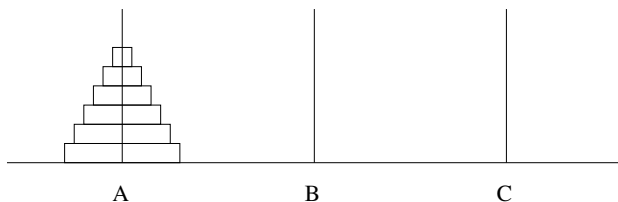
$$m \oslash n = \begin{cases} 0 & \text{se } m < n \\ \text{succ}((m \ominus n) \oslash n) & \text{se } m \geq n \end{cases}$$

- Esempi:

- $5 \ominus 2 = 4 \ominus 1 = 3 \ominus 0 = 3.$
- $5 \otimes 2 = 5 \oplus (5 \otimes 1) = 5 \oplus (5 \oplus (5 \otimes 0)) = 5 \oplus (5 \oplus 0) = 5 \oplus 5 = \dots = 10.$
- $5 \oslash 2 = \text{succ}((5 \ominus 2) \oslash 2) = \dots = \text{succ}(3 \oslash 2) = \text{succ}(\text{succ}((3 \ominus 2) \oslash 2)) = \dots = \text{succ}(\text{succ}(1 \oslash 2)) = \text{succ}(\text{succ}(0)) = \text{succ}(1) = 2.$

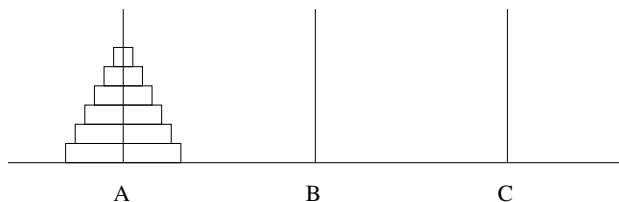
# Il Problema delle Torri di Hanoi

- Siano date tre aste denominate A, B, C.
- Sull'asta A sono accatastati  $n \geq 1$  dischi di diametro diverso in ordine di diametro decrescente dal basso verso l'alto:



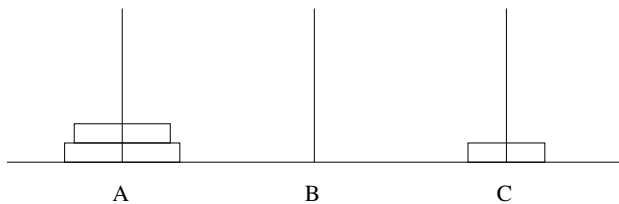
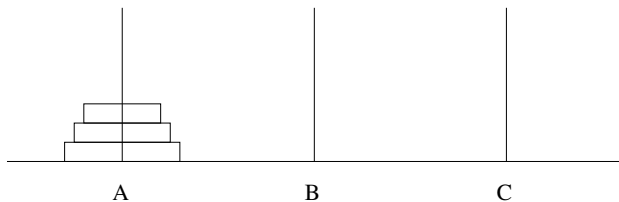
# Il Problema delle Torri di Hanoi

- Siano date tre aste denominate A, B, C.
- Sull'asta A sono accatastati  $n \geq 1$  dischi di diametro diverso in ordine di diametro decrescente dal basso verso l'alto:

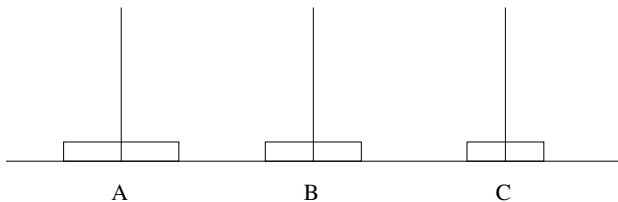
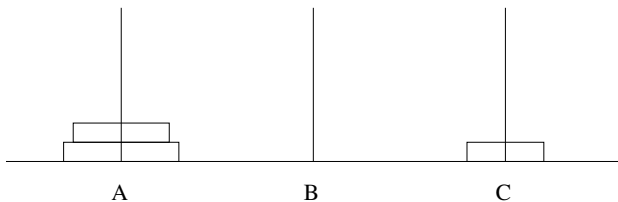


- Spostare i dischi da A a C rispettando le due seguenti regole:
  - È possibile spostare un solo disco alla volta.
  - Un disco non può mai trovarsi sopra uno di diametro inferiore.
- Soluzione banale quando  $n = 1$  oppure  $n = 2$ , meno quando  $n = 3$ .

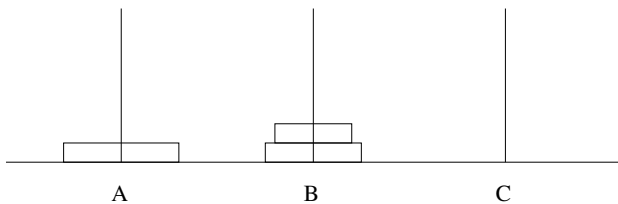
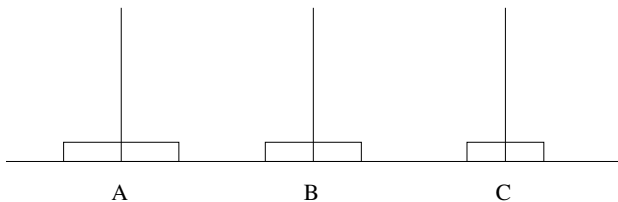
- Mossa 1 quando  $n = 3$ :



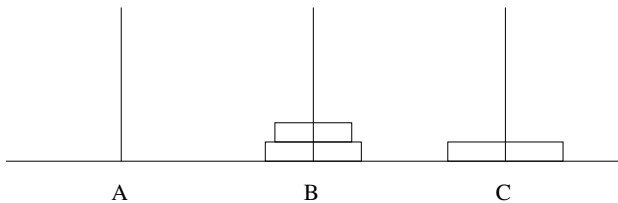
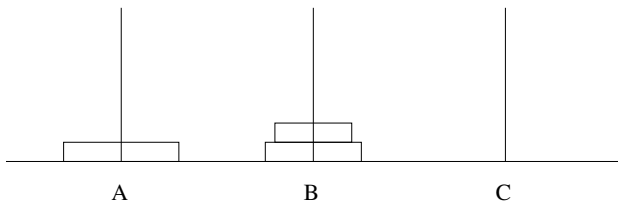
- Mossa 2 quando  $n = 3$ :



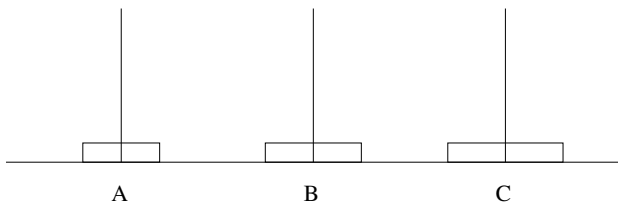
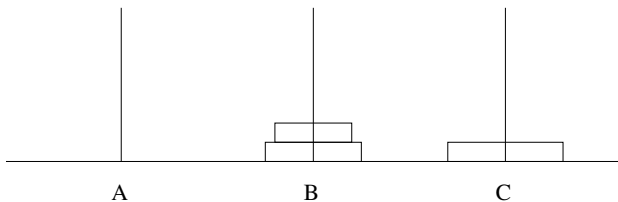
- Mossa 3 quando  $n = 3$ :



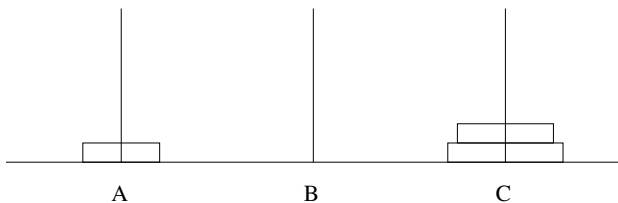
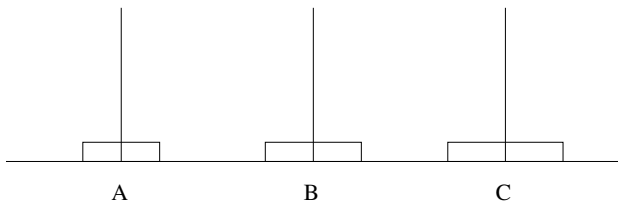
- Mossa 4 quando  $n = 3$ :



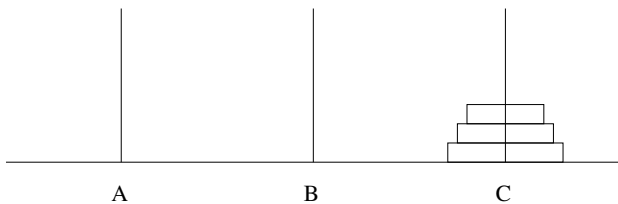
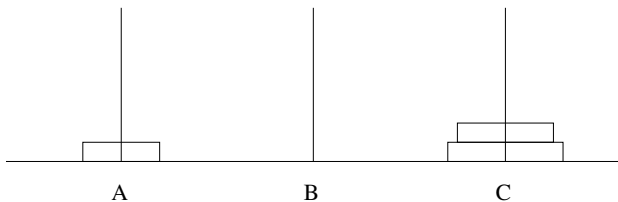
- Mossa 5 quando  $n = 3$ :



- Mossa 6 quando  $n = 3$ :



- Mossa 7 quando  $n = 3$ :



- Quando  $n \geq 4$  diventa praticamente impossibile da risolvere a mano.
- Il numero di mosse aumenta esponenzialmente al crescere di  $n$ .

- Quando  $n \geq 4$  diventa praticamente impossibile da risolvere a mano.
- Il numero di mosse aumenta esponenzialmente al crescere di  $n$ .
- Soluzione ricorsiva:
  - Se  $n = 1$ :
    - Spostare l'unico disco dalla prima alla terza asta.
  - Se  $n > 1$ :
    - Spostare gli  $n - 1$  dischi più piccoli dalla prima alla seconda asta *usando questa soluzione ricorsiva*.
    - Spostare il disco più grande dalla prima alla terza asta.
    - Spostare gli  $n - 1$  dischi più piccoli dalla seconda alla terza asta *usando questa soluzione ricorsiva*.
- Ad ogni passo ricorsivo cambia il ruolo delle tre aste.

## ● Implementazione nel linguaggio C:

```
void hanoi(int n,          /* n >= 1 */
           char partenza, /* 'A' nella chiamata iniziale */
           char arrivo,   /* 'C' nella chiamata iniziale */
           char intermedia) /* 'B' nella chiamata iniziale */
{
    if (n == 1)
        printf("Sposta da %c a %c.\n",
              partenza,
              arrivo);
    else
    {
        hanoi(n - 1,
              partenza,
              intermedia,
              arrivo);
        printf("Sposta da %c a %c.\n",
              partenza,
              arrivo);
        hanoi(n - 1,
              intermedia,
              arrivo,
              partenza);
    }
}
```



1506  
UNIVERSITÀ  
DEGLI STUDI  
DI URBINO  
CARLO BO



CORSO DI LAUREA TRIENNALE

# INGEGNERIA INFORMATICA E SCIENZE COMPUTAZIONALI

#NOPLACEFORBUGS



CORSO DI LAUREA MAGISTRALE

# INFORMATICA E INNOVAZIONE DIGITALE

#MAKEITDIGITAL

<https://informatica.uniurb.it/>