Paolo Camagni
Riccardo Nikolassy

HOEPLI
TECNICA
PER LA SCUOLA

INFOMAT 3

Per il liceo scientifico
opzione scienze applicate

Algoritmi di calcolo numerico
Applicazioni tecnico-scientifiche in C++, Java e VBA
Analisi numerica con Octave e VBA
Principi teorici della computazione
Le reti e i servizi di rete

Edizione **OPENSCHOOL**

1	**LIBRO**DITESTO
2	**E-BOOK+**
3	RISORSE**ONLINE**
4	**PIATTAFORMA**

CD-ROM

HOEPLI

Infom@t

PAOLO CAMAGNI RICCARDO NIKOLASSY

Infom@t

Per il liceo scientifico opzione scienze applicate

Volume 3

Algoritmi di calcolo numerico
Applicazioni tecnico-scientifiche in C++, Java e VBA
Analisi numerica con Octave e VBA
Principi teorici della computazione
Le reti e i servizi di rete

EDITORE ULRICO HOEPLI MILANO

Via Hoepli 5, 20121 Milano (Italy)
tel. +39 02 864871 – fax +39 02 8052886
e-mail hoepli@hoepli.it

www.hoepli.it

LIBRO DIGITALE

Indice

Unità 1

Algoritmi di calcolo numerico

L 1 Calcolo approssimato della radice quadrata 2

Cenni sul calcolo numerico 2
Calcolo della radice quadrata 3
Alcuni metodi proposti da Newton 6
Verifica... le competenze 9

L 2 Calcolo di π con il metodo Monte Carlo e di Buffon 10

La ricerca di pi greco 10
Il metodo Monte Carlo 14
Il problema di Buffon 16
Verifica... le competenze 21

L 3 Calcolo approssimato del numero *e* 22

Generalità 22
Calcolo del numero *e* 24
Ricordare il numero *e* 26
Verifica... le competenze 27

L 4 Calcolo approssimato del seno di un angolo con Taylor e Maclaurin 28

Generalità 28
Algoritmo per il calcolo approssimato del seno 29
Verifica... le competenze 33

L 5 Calcolo approssimato della radice di un'equazione mediante la bisezione 34

Generalità 34
Metodo di bisezione 35
Verifica... le competenze 38

L 6 Calcolo approssimato delle aree 39

Generalità 39
Metodo dei rettangoli 40
Metodo dei trapezi 43
Metodo di Cavalieri-Simpson 44
Verifica... le competenze **47**

L 7 Le equazioni differenziali risolte con il metodo di Eulero 48

Generalità 48
Metodi di discretizzazione 49
Metodo di Eulero 50
Metodo di Eulero modificato 53
Verifica... le competenze 55
Verifica... le competenze 56
Simulazione guidata di compito in classe 57
CLIL 58

Area digitale

- Funzioni inline del linguaggio C++

Unità 2

Applicazioni tecnico-scientifiche in C++ e Java

L 1 Algoritmi crittografici 60

Introduzione alla crittografia 61
Tecniche crittografiche 62
Cifrario di Cesare 63
La scacchiera di Polibio 64

La crittografia e la Grande Guerra 67
La crittografia moderna a chiave asimmetrica 69
L'algoritmo RSA 72
Verifica... le competenze 73

L 2 Anagrammi e permutazioni lessicografiche 74
Premessa 74
Permutazioni semplici e con ripetizione 75
Anagrammi e permutazioni lessicografiche 76
Factoradic, codice di Lehemer e ordine lessicografico 79
Verifica... le competenze 87

L 3 Casualità, caos e numeri pseudocasuali 88
Casualità e caos 88
Processi deterministici e pseudocasuali 90
Numeri pseudocasuali in C++ 91
Numeri pseudocasuali in Java 91
Generare numeri in un range predefinito 92
Algoritmi che generano le sequenze 93
Linear Congruential Generator (LCG) 95
Il caos deterministico e la formica di Langton 98
Verifica... le competenze 101

L 4 La geometria dei frattali 103
Premessa 103
Cenni sull'approccio matematico ai frattali 104
Costruzione e disegno dei frattali 105
La dimensione frattale 106
La curva di Peano 107
Da Tartaglia a Sierpinski 109
L'insieme di Mandelbrot e di Julia 112
Verifica... le competenze 115

L 5 Speranza matematica e gioco d'azzardo 116
Premessa 116
Speranza matematica e gioco equo: note essenziali 117
Testa o croce 120
Il gioco del Craps 122
La legge dei grandi numeri 124
Verifica... le competenze 126

L 6 Le funzioni di hash e la gestione delle collisioni 127
Introduzione 127
L'impronta digitale di un testo 128
Tabelle di hashing 130
Strutture di dati dinamiche: List e Vector 135
La realizzazione della tabella di hashing con i vector 138
Verifica... le competenze 140

L 7 La valutazione di espressioni matematiche postfisse 141
Notazione infissa, prefissa e postfissa 141
Lista con accesso limitato: pile e code 143
Pila o stack 144
L'algoritmo per la valutazione della notazione polacca inversa 146
Cenni sulla struttura dati coda 149
Verifica... le competenze 150
Verifica... le competenze 151
Simulazione guidata di compito in classe 153
CLIL 154

Area digitale

- Il cifrario ROT 13
- Analisi delle frequenze
- Descrizione degli algoritmi RSA e AES
- Disposizioni semplici e con ripetizione
- Combinazioni semplici e con ripetizione
- Un algoritmo iterativo per generare le permutazioni
- Algoritmo LFG (Lagged Fibonacci Generator)
- Il triangolo di Sierpinski mediante affinità
- Il Quadrato di Minkowski, la Polvere di Cantor e alberi frattali
- Soluzioni gioco di estrazione carta
- Pascal e i dadi
- Speranza matematica del Craps
- Conversione da infissa a postfissa e conversione da notazione polacca inversa a infissa
- La torre di Brahma

- Esercizi per il recupero e il rinforzo

Unità 3

Octave: uno strumento per lo sviluppo di applicazioni tecnico-scientifiche

L 1 Octave: l'alternativa open source a MATLAB 156

Introduzione 156
Installazione di Octave 157
Primo utilizzo di Octave 159
Funzioni Octave 161
Verifica... le competenze 163

L 2 Vettori e matrici in Octave 164

I vettori 164
Operazioni sui vettori 167
Le matrici 168
Valutazione di una funzione 171
Verifica... le competenze 174

L 3 Programmare in Octave 176

Script file 176
Function 178
Operatori logici 179
Istruzioni if 180
Cicli in Octave 180
Un programma completo: bubble sort di un vettore 181
Ordinamento matrici 182
Salvare e ripristinare i dati 182
Verifica... le competenze 184

L 4 Realizzare grafici 2D e 3D 186

Premessa 186
Grafici 2D 187
Altri grafici 194
Grafici 3D 196
Verifica... le competenze 201
Verifica... le competenze 202
Simulazione guidata di compito in classe 203
CLIL 204

L 5 Applicazioni alla matematica

L 6 Elaborare le immagini

Puoi scaricare le Lezioni 5, 6 anche da hoepliscuola.it

Area digitale

- Il comando vhos e le variabili predefinite speciali
- Codici ed esempi di formattazione output
- Formati dei numeri disponibili
- Funzioni su vettori
- Il prodotto vettoriale
- Operazioni tra matrici
- Il diary
- Disegno di grafici sovrapposti e affiancati
- Esercizi per l'approfondimento

Unità 4

Applicazioni scientifiche in Excel

L 1 Applicazioni della matematica con Excel 206

Soluzione di sistemi lineari: metodo di Cramer 206
Soluzione di sistemi lineari: metodo di Gauss-Seidel 210
L'interpolazione con Excel 216
Verifica... le competenze 221

L 2 Applicazioni scientifiche in ambiente VBA di Excel 222

L'ambiente VBA di Excel 222
La macchina di Galton 225
Tracciare una retta sul piano cartesiano 229
Tracciare una funzione con il metodo per punti 234
Verifica... le competenze 238
Verifica... i saperi essenziali 239
Simulazione guidata di compito in classe 240
CLIL 242

Area digitale

- Esercizi per il recupero e il rinforzo
- Proposta di compito in classe

Unità 5

Le reti di computer e i servizi di rete

L 1 Le architetture di rete 244

Le architetture di rete 245
Il modello ISO-OSI 246
L'architettura di rete TCP/IP 249
Verifica... le conoscenze 251

L 2 Fondamenti di networking 252

Componenti base di una rete aziendale 253
Modelli di rete aziendale 254
Il cablaggio delle reti 255
I dispositivi di rete 255
Le topologie di rete 257
L'ADSL 260
Verifica... le conoscenze 261

L 3 Indirizzi IP e subnetting nelle reti locali 262

Struttura degli indirizzi IP 263
Classi di indirizzi IP 263
Piano di indirizzamento 265
Verifica... le competenze 272

L 4 La sicurezza nei sistemi informatici 274

Internet e la sicurezza informatica 275
Le minacce all'informazione 275
Minacce in rete 277
Sicurezza di un sistema informatico 278
Valutazine dei rischi 279
Principali tipologie di attacchi informatici 281
Sicurezza nei sistemi informativi distribuiti 283
Verifica... le conoscenze 285

L 5 Firma elettronica, digitale, certificati e PEC 286

Firma elettronica e digitale 286
Funzionamento della firma digitale 288
Firma elettronica remota 291
Il certificato digitale 292
Posta Elettronica Certificata (PEC) 294
La marca temporale 297
Verifica... le conoscenze 298
Verifica... i saperi essenziali 299
Simulazione guidata di compito in classe 301
CLIL 302

Area digitale

- Trucchi per i calcoli sugli indirizzi IP
- Tabella per i subnetting (RCF1878)
- Dati statistici sull'utilizzo di Internet nel mondo e in Europa
- l'Agenzia Europea ENISA, il CERT e il CERT Nazionale Italia
- Breve storia degli attacchi informatici
- Principali tecniche di DoS
- Problematiche di autenticazione dell'utente
- Firewall di rete
- L'algoritmo MD5 e gli algoritmi SHA
- Public Key Infrastructure (PKI)
- Normativa di riferimento della PEC
- Vari formati di marcatura temporale

- Esercizi interattivi

- Esercizi per il recupero e il rinforzo

Unità 6

Android e i dispositivi mobili

L 1 Android: un sistema operativo per applicazioni mobili 304

I dispositivi mobili e la piattaforma Android 304
La struttura di un'applicazione Android 307

Il ciclo di vita di una Activity 309
Il file APK 310
Verifica... le conoscenze 312

L 2 L'ambiente per lo sviluppo di applicazioni Android 313
Android Studio 313
Creare un'applicazione 314
L'ambiente di lavoro 317
Il Project Explorer 318
Il collaudo dell'applicazione mediante un emulatore 319
Configurazione di un dispositivo fisico 322
Mandare in esecuzione una app 323
Effettuare il debug con Android Studio 324
Toast 325
Verifica... le conoscenze 327
Verifica... le competenze 327

L 3 Realizzare un'applicazione utilizzando i widget 328
La modifica del layout 328
Widget di base 331
Altri widget molto utilizzati 334
Verifica... le conoscenze 336
Verifica... le competenze 337

L 4 Utilizzare i sensori nella app 338
SensorManager 338
I sensori per Android 340
La classe Sensor 341
Verifica... le competenze 345

L 5 Realizzare una app completa: la calcolatrice 346
La calcolatrice 346
Verifica... le competenze 349
Verifica... i saperi essenziali 350
Simulazione guidata di compito in classe 351
CLIL 352

Area digitale

- Versioni di Android
- I diversi tipi di tocco su display touch
- Scaricare e installare Android Studio
- Utilizzo dei listener
- Il layout degli elementi grafici
- Esempio riepilogativo: riassumi la mia identità

- Esercizi interattivi

- Esercizi per il recupero e il rinforzo

Unità 7
Principi teorici della computazione

L 1 Elementi di informatica teorica 354
Una caratterizzazione rigorosa del concetto di algoritmo 355
Le macchine di Turing 357
La macchina di Turing universale 362
La tesi di Turing-Church 363
Il problema della decisione di Hilbert 364
Verifica... le conoscenze 366

L 2 La qualità degli algoritmi: introduzione alla complessità computazionale 367
Introduzione 367
Misurare il tempo di calcolo 368
Complessità asintotica 372
Comportamento asintotico: notazione O ("o") grande 372
Istruzione dominante 377
Algoritmi ricorsivi 378
Verifica... le conoscenze 380
Verifica... le competenze 381

L 3 La complessità dei problemi 382
Algoritmi e problemi 382
Problemi computabili ma intrattabili 384
Problemi polinomiali ed esponenziali 385
La classe NP 385
La classe P coincide con la classe NP? 387
La classe NPC o NP completa 388
Risolvere i problemi intrattabili 388
Verifica... le conoscenze 389
Verifica... i saperi essenziali 390
Simulazione guidata di compito in classe 391
CLIL 393

Area digitale

- Esercizi interattivi

Presentazione

Infom@t è un nuovo corso destinato agli studenti dell'indirizzo **Scienze applicate** del **Liceo scientifico** per introdurli allo studio delle scienze e delle tecnologie informatiche.

L'opera è sviluppata facendo tesoro di un'impostazione didattica consolidata e apprezzata nei testi di Informatica degli stessi autori, nel rispetto delle Linee guida ministeriali e con una particolare attenzione alle conoscenze del software di base e applicativo dell'**office productivity** e alle competenze della logica e del **pensiero computazionale**.

STRUTTURA DEL TESTO, METODOLOGIA E STRUMENTI DIDATTICI

Il **terzo** volume è concettualmente organizzato rispettando le aree tematiche indicate dagli OSA ministeriali (CS, RS, IC, BD), che riguardano gli algoritmi di calcolo numerico, le applicazioni tecnico-scientifiche, principi teorici della computazione e complessità degli algoritmi e le reti di computer.

L'opera è strutturata in **sette Unità di Apprendimento** suddivise in **Lezioni**.
Ogni Unità si apre con le finalità e i contenuti dei diversi argomenti descritti negli **obiettivi disciplinari**. Ogni Lezione si apre presentando la tematica *In questa lezione impareremo*, seguita da una *mappa concettuale* che offre una sintetica anticipazione dei contenuti sviluppati, fungendo da schema riepilogativo e di sistematizzazione dei saperi per abituare lo studente all'apprendimento e alla sintesi contenuti attraverso un percorso ragionato.

Oltre al nuovo impianto grafico, la proposta didattica propone, per gli algoritmi di calcolo numerico, le codifiche in parallelo nei linguaggi di programmazione **C++** e **Java**.
La sezione dedicata alle applicazioni tecnico-scientifiche è stata arricchita con due nuovi argomenti che utilizzano le **strutture dati dinamiche** (liste e code), sfruttando le librerie messe a disposizione dai linguaggi di programmazione.
L'Unità dedicata alle reti è stata completamente rivista, aggiungendo una lezione di progettazione del piano di indirizzamento per le reti **TCP/IP**.
È stata aggiunta una Unità per la programmazione dei dispositivi mobili con sistema operativo **Android** che affronta in modo sintetico, ma completo, tutti gli aspetti della realizzazione di applicazioni.
Nello studio della complessità degli algoritmi è stata aggiunta una lezione sulla **macchina di Turing** dove viene anche presentato un emulatore software per poter collaudare gli algoritmi scritti con istruzioni rappresentate da sequenze di quintuple.

Nella trattazione vengono affrontati:
- le metodologie per il calcolo approssimato di aree, della radice quadrata, del valore di π, del numero *e*, del seno di un angolo, della radice di una equazione e della soluzione di equazioni differenziali;
- l'applicazione dell'informatica alla crittografia, alla generazione di permutazioni, allo studio della probabilità, della casualità e del caos, alla geometria frattale, alla generazione di funzioni di hash e nella valutazione di espressioni matematiche prefisse;
- **Octave** come alternativa open source a **MATLAB** per la soluzione di sistemi mediante l'elaborazione di matrici, la valutazione di funzioni e la loro rappresentazione grafica sia in 2D che in 3D;
- l'approfondimento del foglio di calcolo Excel come strumento per la realizzazione di soluzioni algoritmiche di calcolo numerico e di analisi di funzioni matematiche;
- le architetture di rete e i concetti fondamentali di networking, l'organizzazione degli indirizzi IP, le problematiche legate alla sicurezza e ai nuovi servizi offerti dalla rete;

- la programmazione per dispositivi mobili **Android**, con un insieme di lezioni che permettono di realizzare **app** anche complesse, connesse a database remoti;
- i principi teorici della computazione e l'analisi della complessità degli algoritmi.

L'esposizione dei contenuti è semplice e dettagliata. Il testo, integrato da contenuti online, accompagna lo studente in un preciso percorso didattico e costituisce un riferimento chiaro per il suo apprendimento e per l'acquisizione delle competenze previste dalle linee guida. I materiali collegati all'eBook+ sono attivabili attraverso le icone nella versione digitale.

Alla fine di ogni Lezione vengono proposti **esercizi** di valutazione delle conoscenze (domande a risposta multipla, vero/falso, riordino) e delle competenze (simulazioni informatiche, realizzazione di algoritmi e di programmi) raggiunte.

Alla fine di ogni Unità è presente una **sezione laboratoriale** ricca di proposte di esercizi per ogni livello di apprendimento, proponendo test e verifiche specifiche per gli alunni che necessitano di strumenti integrativi e metodi compensativi. Inoltre viene proposta una **nuova** sezione per la **preparazione al compito in classe**.
Chiude l'Unità una scheda CLIL che propone, in inglese, i concetti chiave dell'unità e alcuni quesiti di tipologie diverse

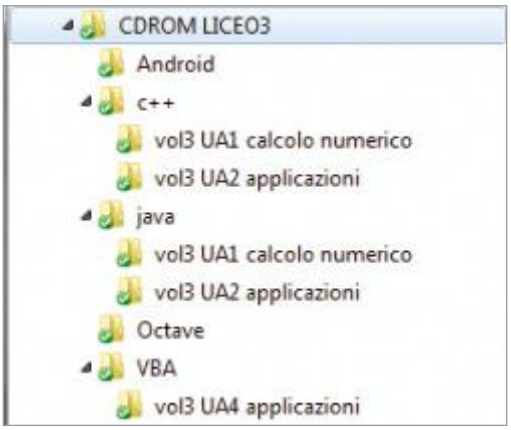
ESPANSIONI DIGITALI

La nuova edizione Openschool consente di:
– scaricare gratuitamente il libro digitale arricchito (eBook+); l'eBook+ permette in particolare di:
  - eseguire tutte le esercitazioni a risposta chiusa in modo interattivo;
  - scaricare gli approfondimenti tematici e gli esercizi per l'approfondimento;
  - scaricare lezioni integrative;

– disporre di ulteriori esercitazioni online, utilizzabili a discrezione del docente per classi virtuali gestibili attraverso la piattaforma Open.

RISORSE ONLINE E PIATTAFORMA DIDATTICA

Sul sito **www.hoepliscuola.it** (hoepliscuola.it) sono disponibili **numerosi materiali**. In particolare, per lo studente: **approfondimenti**, **esercizi** di recupero, rinforzo e approfondimento.
Inoltre, i **file** richiamati nelle lezioni e nelle esercitazioni contenuti nel CD-ROM allegato al volume sono scaricabili anche dal sito.



CD-ROM

Il CD-ROM allegato al volume contiene i file degli esempi nonché il materiale necessario per eseguire le procedure guidate passo passo degli esercizi svolti e da svolgere proposti a fine lezione e a fine unità.

Struttura del corso per immagini

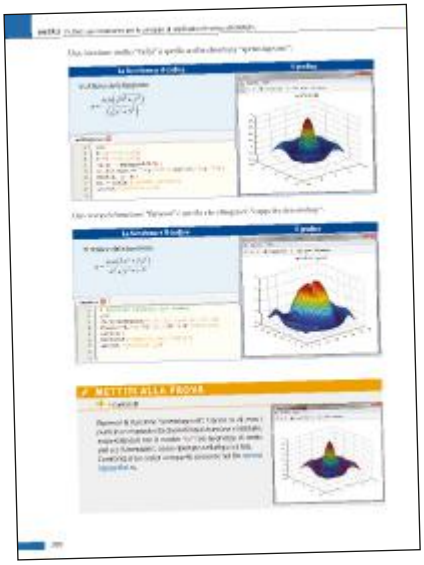
APERTURA UNITÀ
L'Unità si apre con l'indice delle lezioni sviluppate e l'indicazione degli obiettivi generali suddivisi in conoscenze, competenze e abilità.

APERTURA LEZIONE
La Lezione si apre con una breve sintesi degli argomenti trattati e con la schematizzazione dei contenuti attraverso una MAPPA CONCETTUALE.

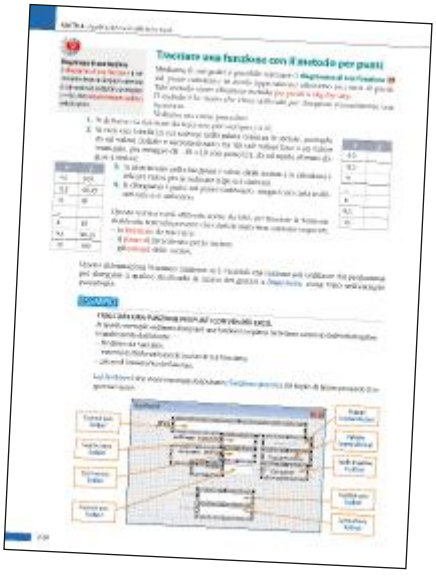
ATTENZIONE
Individua aspetti su cui focalizzare l'attenzione.

EVIDENZIAZIONE
Connota dei concetti da ricordare.

WIKI
Specifica il significato di un termine.

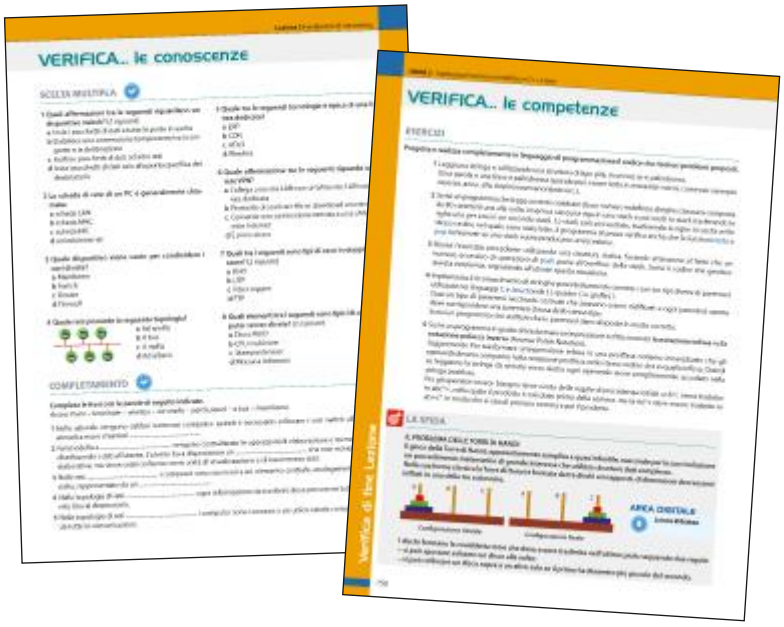


METTITI ALLA PROVA
Appendice esercitativa che prende spunto dal problema di partenza accrescendone le funzionalità e il campo applicativo.

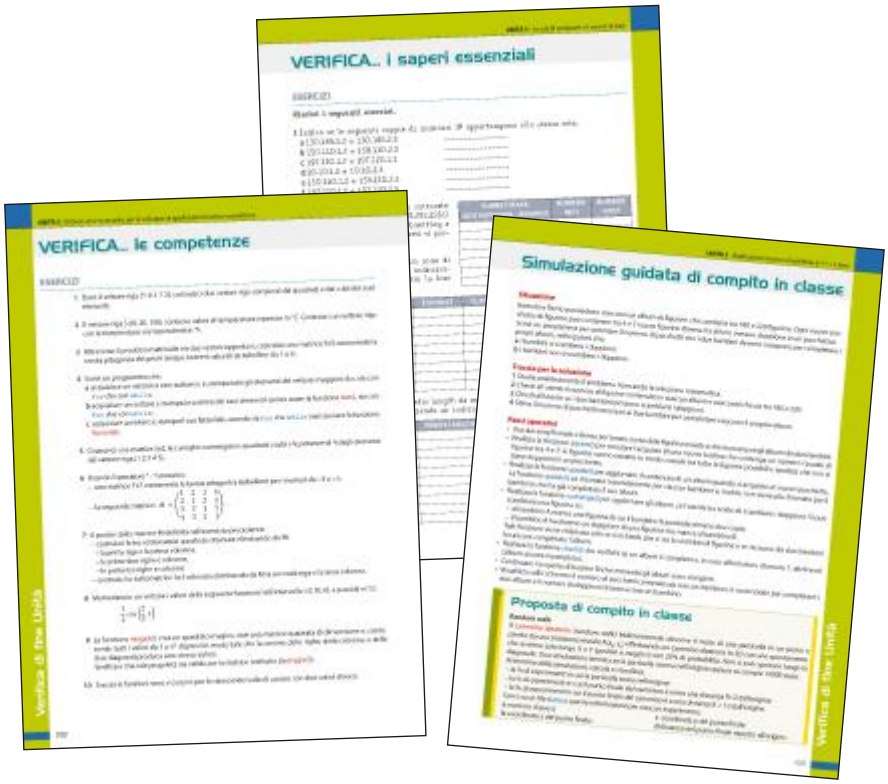


ESEMPIO
Gli esempi chiariscono i concetti appena esposti e svolgono la funzione di traccia di svolgimento per lo studente.

PER SAPERNE DI PIÙ
Schede di approfondimento degli argomenti sviluppati nel volume.



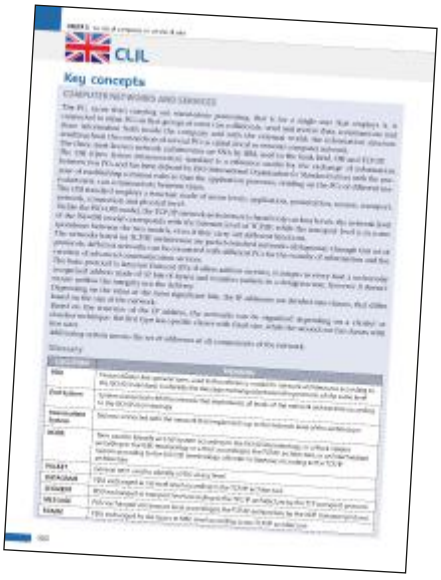
VERIFICA LE CONOSCENZE / LE COMPETENZE
Presenti a fine Lezione, propongono varie tipologie di test (quelli a risposta chiusa sono interattivi e autocorrettivi) e di esercizi, con l'indicazione dei problemi connessi ai compiti di realtà.



VERIFICA I SAPERI ESSENZIALI/VERIFICA LE COMPETENZE
Al termine di ogni Unità è presente una sezione che contiene esercizi sommativi suddivisi in livelli di difficoltà per la verifica dei saperi acquisiti, con indicati i problemi connessi ai compiti di realtà.

SIMULAZIONE GUIDATA DI COMPITO IN CLASSE
Presente alla fine di ogni Unità, consente di testare la preparazione prima della verifica in classe.

CLIL
Chiude ogni Unità una scheda CLIL, che propone in inglese, i concetti chiave dell'unità e alcuni quesiti di diverse tipologie.



L'OFFERTA DIDATTICA **HOEPLI**

L'edizione **Openschool** Hoepli offre a docenti e studenti tutte le potenzialità di Openschool Network (ON), il nuovo sistema integrato di contenuti e servizi per l'apprendimento.

Edizione **OPENSCHOOL**




LIBRO DI TESTO

Il libro di testo è l'**elemento cardine** dell'offerta formativa, uno strumento didattico **agile** e **completo**, utilizzabile **autonomamente** o in combinazione con il ricco **corredo digitale** offline e online. Secondo le più recenti indicazioni ministeriali, volume cartaceo e apparati digitali **sono integrati in un unico percorso didattico.** Le espansioni accessibili attraverso l'eBook+ e i materiali integrativi disponibili nel sito dell'editore sono puntualmente richiamati nel testo tramite apposite icone.

eBOOK+

L'**eBook+** è la versione digitale e interattiva del libro di testo, utilizzabile su **tablet, LIM e computer**. Aiuta a comprendere e ad approfondire i contenuti, rendendo l'apprendimento più attivo e coinvolgente. Consente di leggere, annotare, sottolineare, effettuare ricerche e accedere direttamente alle numerose **risorse digitali integrative**.
➜ Scaricare l'eBook+ è molto **semplice**. È sufficiente seguire le istruzioni riportate nell'ultima pagina di questo volume.

RISORSE ONLINE

Il sito della casa editrice offre una ricca dotazione di **risorse digitali** per l'approfondimento e l'aggiornamento. Nella pagina web dedicata al testo è disponibile **MyBookBox**, il contenitore virtuale che raccoglie i materiali integrativi che accompagnano l'opera.
➜ Per accedere ai materiali è sufficiente registrarsi al sito **www.hoepliscuola.it** e inserire il codice coupon che si trova nella terza pagina di copertina. **Per il docente** nel sito sono previste ulteriori risorse didattiche dedicate.

PIATTAFORMA DIDATTICA

La **piattaforma didattica** è un ambiente digitale che può essere utilizzato in modo duttile, a misura delle esigenze della classe e degli studenti. Permette in particolare di **condividere contenuti** ed **esercizi** e di partecipare a **classi virtuali**. Ogni attività svolta viene salvata sul **cloud** e rimane sempre disponibile e aggiornata. La piattaforma consente inoltre di consultare la versione online degli eBook+ presenti nella propria libreria.
➜ È possibile accedere alla piattaforma attraverso il sito **www.hoepliscuola.it**.

UNITÀ 1

Algoritmi di calcolo numerico

LEZIONE 1
Calcolo approssimato della radice quadrata

LEZIONE 2
Calcolo di π con il metodo Monte Carlo e di Buffon

LEZIONE 3
Calcolo approssimato del numero *e*

LEZIONE 4
Calcolo approssimato del seno di un angolo con Taylor e Maclaurin

LEZIONE 5
Calcolo approssimato della radice di un'equazione mediante la bisezione

LEZIONE 6
Calcolo approssimato delle aree

LEZIONE 7
Le equazioni differenziali risolte con il metodo di Eulero

CONOSCENZE

- Comprendere le basi del calcolo numerico
- Ripercorrere nella storia la ricerca del valore di π
- Conoscere i concetti fondamentali sul calcolo approssimato delle aree
- Conoscere i concetti fondamentali sui metodi di discretizzazione

COMPETENZE

- Saper risolvere il problema di Buffon
- Codificare l'algoritmo babilonese e di Newton per il calcolo della radice quadrata
- Utilizzare i polinomi di Maclaurin e di Taylor per approssimare la funzione sen(x)
- Utilizzare il metodo Monte Carlo per il calcolo delle aree
- Codificare l'algoritmo approssimato per il calcolo di sen(x)

ABILITÀ

- Implementare il metodo di bisezione
- Implementare il metodo dei rettangoli
- Implementare il metodo dei trapezi
- Implementare il metodo di Cavalieri-Simpson
- Implementare il metodo di Eulero e del punto centrale

AREA TEMATICA (CS)

Il file relativi a esempi, esercizi e a eventuali soluzioni di questa unità sono contenuti nel CD-ROM e sono anche scaricabili dal sito hoepliscuola.it

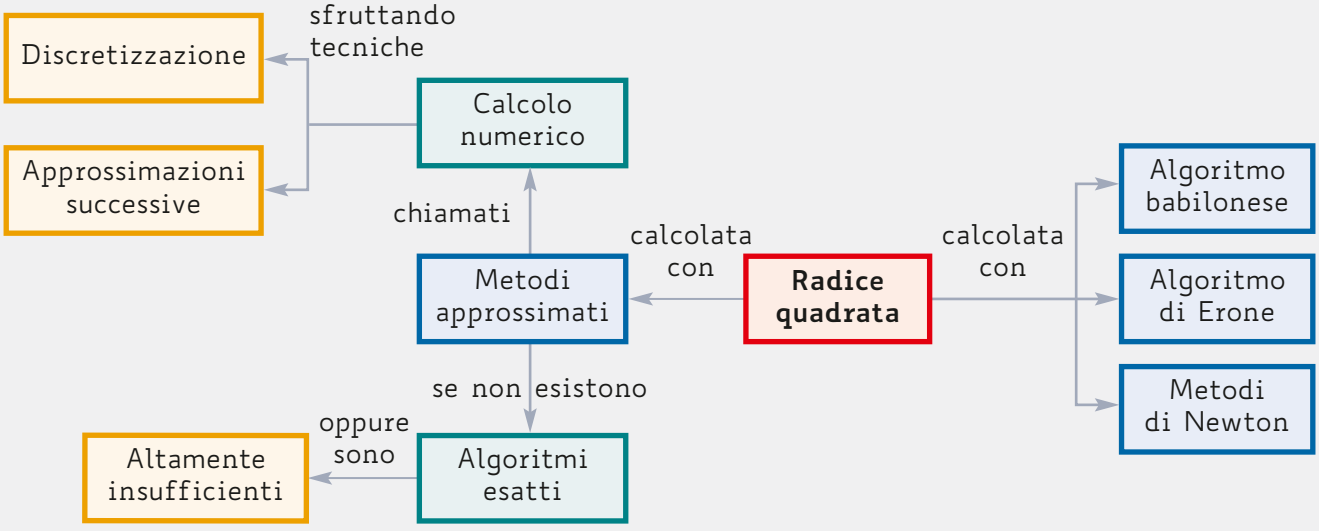
1 Calcolo approssimato della radice quadrata

IN QUESTA LEZIONE IMPAREREMO...

- le basi del calcolo numerico
- l'algoritmo babilonese per il calcolo della radice quadrata
- alcuni algoritmi di Newton

MAPPA CONCETTUALE

didattica inclusiva



Cenni sul calcolo numerico

La matematica svolge un ruolo determinante nell'analisi dei problemi del mondo reale e il processo di soluzione associa al problema reale un modello matematico che ne *approssima la soluzione*: molto spesso, però il modello matematico che si ottiene non è esprimibile con funzioni elementari, ma con sistemi complessi che richiedono l'uso di metodi di calcolo approssimato: questi metodi prendono il nome di metodi di calcolo numerico.
Sono caratterizzati da algoritmi che consentono di ottenere una soluzione del problema in forma approssimata (con il grado di precisione generalmente legato al numero delle iterazioni e quindi al tempo di calcolo) del problema specifico producendo un valore altrimenti impossibile da determinare con metodi algebrici.

Lo schema di calcolo è generalmente abbastanza semplice ma richiede numerosi passaggi ripetitivi e il calcolatore si presta bene alla loro esecuzione.

Algoritmo numerico approssimato
In sintesi un algoritmo numerico approssimato viene utilizzato:
- per calcolare l'approssimazione di un valore numerico non altrimenti determinabile con metodi algebrici;
- per calcolare l'approssimazione di un valore numerico non facilmente determinabile in forma esatta con metodi algebrici;
- per calcolare l'approssimazione di un valore numerico con una precisione maggiore rispetto a un algoritmo ottenuto come implementazione di un metodo algebrico esatto.

Il calcolo approssimato viene utilizzato sia quando non esistono algoritmi esatti di risoluzione, sia quando gli algoritmi esatti non sono efficienti avendo un alto costo computazionale in termini di tempo o di memoria; inoltre gli algoritmi numerici approssimati si sono dimostrati in grado di calcolare approssimazioni delle soluzioni con maggiore velocità e precisione rispetto ad algoritmi derivati dal metodo algebrico esatto.
In alcuni casi, pur esistendo algoritmi algebrici per la determinazione di soluzioni esatte, la dimensione del problema rende comunque necessario il ricorso a un programma di calcolo: si pensi ad esempio alla ricerca della soluzione di un sistema lineare composto da molte equazioni e molte incognite (in natura molti problemi reali hanno come modello matematico sistemi lineari con centinaia di equazioni e centinaia di incognite!).
Le tecniche utilizzate dagli **algoritmi approssimanti**, o algoritmi euristici, si basano su:
- discretizzazione: passaggio da domini continui a domini discreti;
- approssimazioni successive: metodi iterativi.

Come primo esempio di metodo iterativo descriveremo un noto algoritmo per il calcolo della radice quadrata.

Calcolo della radice quadrata

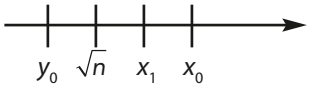
Qualsiasi calcolatrice ha a disposizione la funzione che calcola il valore approssimato della radice quadrata di un qualsiasi numero con l'accuratezza di molte cifre decimali: questo valore viene calcolato grazie a un semplice algoritmo di calcolo numerico che ripete per un numero definito di volte alcune semplici operazione aritmetiche.
Vediamo di descrivere l'origine di questo algoritmo: calcoliamo la radice quadrata di un numero positivo n, cioè $x = \sqrt{n}$. Scegliamo a piacere un numero $x_0 \geq \frac{n}{2}$ che sarà la prima approssimazione per eccesso del valore cercato.

Prendiamo quindi $x_0 = \frac{n}{2}$.

Calcoliamo ora un secondo valore con la seguente formula: $y_0 = \frac{n}{x_0}$ e quindi sarà $n = x_0 y_0$ dove la media geometrica tra x_0 e y_0 viene presa come il primo valore che approssima per difetto la $\sqrt{n}$, che calcoliamo con la seguente espressione:

$$x_1 = \frac{x_0 + y_0}{2} = \frac{1}{2}\left(x_0 + \frac{n}{x_0}\right)$$

La media aritmetica di due numeri è sempre maggiore o uguale alla loro media geometrica, quindi il numero x1 ottenuto sarà sempre compreso tra $\sqrt{n}$ e x0 e approssimerà meglio la radice ricercata rispetto al valore x0 precedentemente assunto, dato che x0 > x1 > $\sqrt{n}$.



Ripetendo la medesima operazione più volte, otteniamo una sequenza di numeri che convergono proprio alla $\sqrt{n}$.
Dapprima calcoliamo

$$y_1 = \frac{n}{x_1}$$

quindi

$$x_2 = \frac{x_1 + y_1}{2} = \frac{1}{2}\left(x_1 + \frac{n}{x_1}\right)$$

Generalizzando i passi descritti in precedenza è possibile dimostrare per induzione che:

$$\lim_{i \to \infty} x_i = \sqrt{n}$$

ESEMPIO

R = 9
Q1 = 9/2 = 4,5
Q2 = (4,5 + 9/(4,5))/2 = (6,5)/2 = 3,25
Q3 = (3,25 + 9/(3,25)) = (3,25 + 2,769)/2 = (6,019)/2 = 3,0095

Questo algoritmo è anche conosciuto come algoritmo babilonese anche se spesso viene attribuito a matematici posteriori, come Archita (428-365 a.C.) oppure a Erone di Alessandria (vissuto tra il I e il II secolo d.C.)

Scriviamo ora il codice dell'algoritmo che calcola `Q = radice_q(num)`.
Inizializziamo la prima iterazione al valore `num/2` e in un ciclo che viene ripetuto `max = 10` volte calcoliamo i successivi termini con $Q = \left(QPrec + \frac{num}{QPrec}\right) \cdot \frac{1}{2}$ e assegniamo questo valore come `QPrec` per ripetere le operazioni.
Scriviamo dapprima la funzione `radice_q()` che verrà successivamente richiamata dal `main()`:

Codifica C++

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
#define MAX 10                    // nr iterazioni

double radice_q(double num){
  double Q, QPrec;
  int n = 0;
  QPrec = num / 2;                // primo termine
  for (n=0; n < MAX ; n++){
    Q = (QPrec + num / QPrec)/2; // nuovo termine
    QPrec = Q;
  }
  return Q;
}
```

Codifica Java

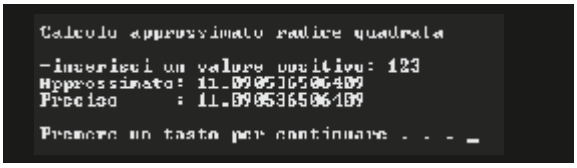
```
import java.util.Scanner;
public class Radice1{
  public static int MAX = 10;          // nr iterazioni
  static double radice_q(double num){
    double Q = 0 , QPrec;
    int n = 0;
    QPrec = num / 2;                   // primo termine
    for (n=0; n < MAX ; n++){
      Q = (QPrec + num / QPrec) / 2;   // nuovo termine
      QPrec = Q;
    }
    return Q;
  }
```

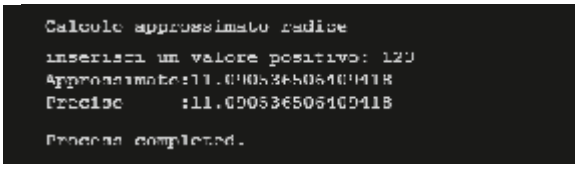
Il codice del `main()` è riportato sotto nella figura a sinistra. Il programma di prova è riportato sotto nella figura a destra.

```
int main(){
  double numero;
  cout.precision(15); // fisso num decimali
  cout <<"Calcolo approssimato radice quadrata\n";
  cout << "-inserisci un valore positivo: ";
  cin  >> numero;

  cout << "\nApprossimato: " << radice_q(numero);
  cout << "\nPreciso     : " << sqrt(numero);
}
```

```
  public static void main (String []arg){
    Scanner in = new Scanner(System.in);
    double numero;
    System.out.println("Calcolo approssimato radice \n");
    System.out.print("inserisci un valore positivo: ");
    numero = in.nextInt();
    System.out.println("Approssimato:"+radice_q(numero));
    System.out.println("Preciso     :"+Math.sqrt(numero));
  }
}
```

Visualizziamo il valore calcolato dal nostro algoritmo e quello calcolato dalla funzione di libreria per effettuare il confronto.





Il codice è memorizzato nei file radice1.cpp e Radice1.java.

Si può osservare che già dopo sole 10 iterazioni i risultati coincidono per valori di n inferiori a 45.000. All'aumentare di n aumenta l'intervallo e quindi i due valori si discostano sempre di più.

METTITI ALLA PROVA

• Calcolo numerico

Modifica il programma facendo in modo che l'utente inserisca il numero di iterazioni che devono essere eseguite: compila una tabella indicando l'errore che si commette per dieci unità di misura diverse sia come numero di iterazioni che come valore da calcolare.

Se con numeri bassi il risultato viene ottenuto con poche iterazioni è necessario migliorare l'algoritmo introducendo una seconda condizione d'uscita che effettua il controllo sull'entità del miglioramento effettuato a ogni passo, cioè confronta due iterazioni successive.

Indichiamo con $errore = \frac{Q - Qprec}{Q}$ e stabiliamo un parametro EPS = 0,00001 come valore di errore minimo al di sotto del quale l'elaborazione deve terminare.
Il nuovo algoritmo è il seguente:

Codifica C++

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
#define MAX 1000                    //numero di iterazioni
#define EPS 0.00001                 //precisione desiderata
double radice_q(double num){
  double Q, QPrec, errore;
  int n, esci = 0;
  QPrec = num / 2;
  for(n = 0; n < MAX && !esci; n++){
    Q = (QPrec + num / QPrec)/2;  //nuovo termine
    errore = fabs((Q - QPrec)/Q); //val.assoluto errore
    if(errore < EPS)
      esci = 1;
    else QPrec = Q;
  }
  if(n >= MAX)
    printf("Numero iterazioni eccessivo\n");
  return Q;
}
```

Codifica Java

```
import java.util.Scanner;

public class Radice2{
  public static int MAX = 10;          // numero iterazioni
  public static double EPS = 0.00001; // precisione voluta

  static double radice_q(double num){
   double Q=0, QPrec, errore;
   int n, esci = 0;
   QPrec = num / 2;
   for(n = 0; (n < MAX) && (esci==0); n++){
    Q = (QPrec + num / QPrec) / 2;   // nuovo termine
    errore = Math.abs((Q - QPrec)/Q);// valore assoluto
    if(errore < EPS)                 // condizione uscita
      esci = 1;
    else QPrec = Q;
   }
   if(n >= MAX)
     System.out.println("Numero iterazioni eccessivo\n");
   return Q;
  }
```

Il codice è memorizzato nei file radice2.cpp e Radice2.java.

Il controllo dell'errore viene effettuato calcolando il valore assoluto della differenza dei valori approssimati per difetto e per eccesso e confrontandolo con il massimo errore accettabile rappresentato dalla costante EPS.

Rimane invariato il programma chiamante.
L'algoritmo di Erone si presta perfettamente a una sua interpretazione in chiave ricorsiva.
La condizione di terminazione è rappresentata dal raggiungimento del valore di approssimazione EPS richiesta mentre la chiamata ricorsiva viene effettuata dopo aver calcolato il passo iterativo della formula di Erone.

Codifica C++

```
#include <iostream>
#include <math.h>
using namespace std;
#define EPS 0.001                    // precisione

double radice_q(double a, double &x1){
  if((fabs(x1 - a/x1)/2) >= EPS)     // ctr errore
  {
    x1 = (x1 + a/x1)/2;
    return radice_q(a, x1);          // ricorsione
  }
  else
    return x1;
 }
```

Codifica Java

```
import java.util.Scanner;

public class RadiceEroneR{

  public static double EPS = 0.00001; // precisione

  static double radice_q(double a, double x1){
  if((Math.abs(x1 - a/x1)/2) >= EPS) // ctr errore
  {
    x1 = (x1 + a / x1) / 2;
    return radice_q(a, x1);          // ricorsione
  }
  else
    return x1;
 }
```

Il codice è memorizzato nei file radiceErone.cpp e RadiceErone.java.

Alcuni metodi proposti da Newton

Metodo delle tangenti

Anche Newton si dedicò a proporre soluzioni per il calcolo approssimato della radice: un primo algoritmo, chiamato anche metodo delle tangenti, utilizza la seguente funzione:

$$f(x) = x^2 - a$$

da cui si ottiene:

$$x_{n+1} = \frac{x_n}{2} + \frac{a}{2x_n}$$

ponendo il primo termine

$$x_0 = 1{,}0$$

Di seguito riportiamo la codifica dove indichiamo come costante manifesta il numero di iterazioni da effettuare:

Codifica C++

```
#include <iostream>
#include <cmath>
using namespace std;
#define ITERAZIONI 10

double radice_q(double a){
  int volte = 0;
  double x = 1.0;
  for (volte = 0; volte <= ITERAZIONI; volte++)
    x = (x / 2) + (a / (2 * x ));
  return x;
}
```

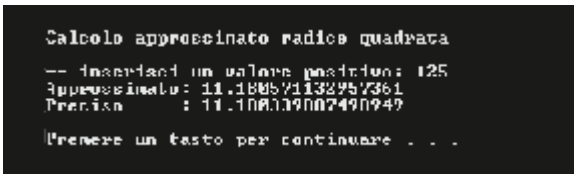
Codifica Java

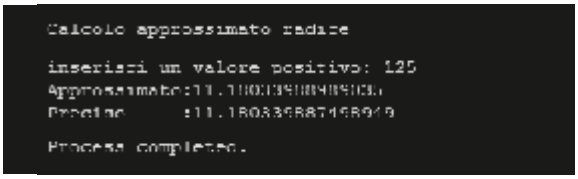
```
import java.util.Scanner;

public class RadiceNewton{
  public static int MAX = 10;

  static double radice_q(double a){
    int volte = 0;
    double x = 1.0;
    for (volte = 0; volte <= MAX; volte++)
      x = (x / 2) + (a / (2 * x ));
    return x;
  }
```

Viene mandato in esecuzione sempre dal medesimo `main()`.

> Questo metodo da il meglio di se quando il valore iniziale è abbastanza vicino al valore della radice *n*-esima, altrimenti possono essere necessarie molte iterazioni per raggiungere un valore accettabile.

Un'esecuzione fornisce il seguente output:





Il codice è memorizzato nei file radiceNewton.cpp e RadiceNewton.java.

Metodo per il calcolo dell'inverso

Un algoritmo alternativo sempre attribuito a Newton è quello che ci permette di trovare l'inverso della radice utilizzando la seguente funzione:

$$g(x) = \frac{1}{x^2} - a$$

da cui si ottiene:

$$x_{n+1} = 0{,}5 \cdot x_n \cdot (3 - a x_n^2)$$

ponendo il primo termine

$$x_0 = 0{,}5$$

Analogamente al precedente, riportiamo la codifica dove indichiamo come costante manifesta il numero di iterazioni da effettuare.

Codifica C++

```
#include <iostream>
#include <cmath>
using namespace std;
#define ITERAZIONI 10

double inv_radice_q(double num){
  int volte = 0;
  float x = 0.002;
  for (volte = 0; volte <= ITERAZIONI; volte++){
    x = 0.5 * x * (3 - num * x * x);
  }
  return x;
}
```

Codifica Java

```
import java.util.Scanner;
public class RadiceInversaNewton{

  public static int MAX = 10;
    static double inv_radice_q(double num){
    int volte = 0;
    double x = 0.002;
    for (volte = 0; volte <= MAX; volte++){
      x =  0.5 * x * (3 - num * x * x);
    }
    return x;
  }
```

Il seguente `main()` richiama questa funzione e ricava il valore della radice calcolando l'inverso del suo risultato:

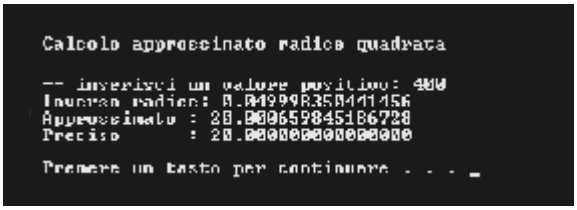
Codifica C++

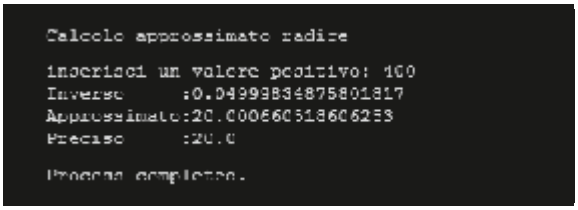
```
int main() {
    double num, radice;
    cout.precision(15); // fisso num decimali
    cout << "Calcolo approssimato radice quadrata\n";
    cout << "-- inserisci un valore positivo: ";
    cin >> num;
    radice = inv_radice_q(num);
    cout << "\nInverso radice: " << radice;
    cout << "\nApprossimato  : " << 1/radice;
    cout << "\nPreciso       : " << sqrt(num);
  }
```

Codifica Java

```
  public static void main (String []arg){
    Scanner in = new Scanner(System.in);
    double numero, radice;
    System.out.println("Calcolo approssimato radice \n");
    System.out.print("inserisci un valore positivo: ");
    numero = in.nextInt();
    radice = inv_radice_q(numero);
    System.out.println("Inverso     :"+ radice);
    System.out.println("Approssimato:"+ 1/radice);
    System.out.println("Preciso     :"+ Math.sqrt(numero));
  }
}
```

Un'esecuzione fornisce il seguente output:





Il codice è memorizzato nei file radiceInversaNewton.cpp e RadiceInversaNewton.java.

Questo metodo da il meglio di se quando il valore iniziale è abbastanza vicino al valore della radice *n*-esima, altrimenti possono essere necessarie sequenze negative che non portano alla soluzione.

Metodo della successione che utilizza l'algoritmo babilonese

Un metodo alternativo sempre proposto da Newton utilizza la seguente successione:

$$x_{n+1} = \frac{3}{2}x_n - \frac{a}{2}x_n^3 = \frac{3x_n - ax_n^3}{2}$$

che ha come limite il valore $x = \frac{1}{\sqrt{a}}$.

Possiamo quindi scrivere un algoritmo iterativo utilizzando tale formula, ma è necessario conoscere un valore stimato che funga da primo termine della successione e questo valore può essere lo stesso utilizzato nell'algoritmo babilonese prima descritto.

METTITI ALLA PROVA

• Calcolo numerico

1 Scrivi un programma che acquisisca dall'utente il valore della precisione EPS compresa tra 0,1 e 0,000001 e quindi il valore di a per il quale si deve calcolare la radice, verificando che questo sia positivo.

2 Utilizzando un valore a piacere di x_0, realizza 2 iterazioni con il metodo babilonese per calcolare x_2 quindi scrivi l'algoritmo che utilizzi il nuovo metodo iterativo che assegna come valore iniziale $1/x_2$.

VERIFICA... le competenze

ESERCIZI

1 Scrivi un programma che effettui il calcolo della radice quadrata di un numero utilizzando il seguente metodo (anch'esso attribuito agli antichi Babilonesi) approssimato alla n-esima iterazione di calcolo letta come parametro di ingresso. Ogni iterazione viene effettuata nel modo seguente:
– si definisce la radice "per eccesso" del numero considerato: eccesso1 ← num
– si definisce la radice "per difetto" del numero considerato: difetto1 ← num / eccesso1
– si prende la media aritmetica come nuovo valore di eccesso: (eccesso2 ← difetto1 + eccesso1) / 2
Esempio con num = 2:

```
prima iterazione      eccesso1 = num
                      difetto1 = num / eccesso1 = 2 / 2          = 1
seconda iterazione    eccesso2 = (eccesso1 + difetto1) / 2       = 1,5
                      difetto2 = num / eccesso2 = 2 / 1,5        = 1,33333
terza iterazione      eccesso3 = (eccesso2 + difetto2)/2         = 1,41667
                      difetto3 = num / eccesso3 = 2 / 1,41667    = 1,411765
quarta iterazione     eccesso4 = (eccesso3 + difetto3) / 2       = 1,414216
                      difetto4 = ...
```

2 Scrivi un programma che effettui il calcolo della radice quadrata di un numero utilizzando il metodo di Cataldi approssimato alla *n*-esima iterazione di calcolo letta come parametro di ingresso.
Per esempio, per il numero 3 ogni iterazione viene effettuata nel modo seguente:
– si pone x da determinare come $x + 1 = \text{sqrt}(3)$ e, risolvendo, si ottiene:
$(x + 1)^2 = 3$, da cui $x^2 + 2x + 1 = 3$, e infine $x = 2 / (x + 2)$
– la sequenza $x_0 = 0$, $x_1 = 2 / (x_0 + 2)$, $x_2 = 2 / (x_1 + 2)$ viene utilizzata per approssimare il calcolo.

3 Modifica l'algoritmo di Erone in modo che l'utente scelga il numero massimo di iterazioni e la precisione desiderata: quindi visualizza a ogni iterazione l'errore che viene commesso. Introduci anche un controllo sui numeri inseriti che devono essere positivi.

4 Scrivi un programma che acquisisca dall'utente il valore della precisione EPS desiderata compresa tra 0,1 e 0,0001 (verificando la correttezza dell'inserimento) e il valore del numero a del quale si vuole calcolare la radice, verificando che sia positivo. Esegui il calcolo della radice mediante l'algoritmo babilonese terminando l'elaborazione quando si raggiunge il valore dell'errore e visualizzando sullo schermo il numero delle iterazioni che sono state fatte.

5 Scrivi un programma che acquisisca dall'utente il valore della precisione EPS desiderata compresa tra 0,1 e 0,0001 (verificando la correttezza dell'inserimento) e il valore del numero a $>10^6$ del quale si vuole calcolare la radice, verificando che sia positivo e rispetti la condizione desiderata. Esegui il calcolo della radice mediante entrambi gli algoritmi confrontando i valori per ogni ordine di grandezza dell'errore.

6 Scrivi un programma che effettui il calcolo della radice quadrata di un numero utilizzando un ulteriore metodo proposto da Newton approssimato alla *n*-esima iterazione (ricordando che $x_{n+1} = (1/2(x_n + \text{num}/x_n))$.

LA SFIDA

CALCOLO DELLA RADICE QUADRATA DI UN NUMERO INTERO CON L'ALGORITMO DI BOMBELLI
Scrivi un algoritmo che implementi il metodo di Bombelli di seguito descritto che opera utilizzando tre variabili intere *x, r* e *d*; in *x* si forma il risultato, in *r* è presente un resto e in *d* una cifra da accodare alla precedente scrittura di *x*. Inizialmente *x* e *r* sono poste a 0 e si procede iterando le seguenti istruzioni:
– modifica la scrittura del resto *r* accodandole il gruppo di cifre più significativo (quelle più a sinistra) non ancora usato; chiamiamo *c* il numero così ottenuto (valore corrente);
– trova la più grande *d* tale che $y = d\,(20x + d)$ non superi *c*: accoda questa nuova cifra alla scrittura del risultato *x*;
– sottrai *y* da *r* e ottieni così il nuovo resto.