Calcolo iterativo radice

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

double calcolaRadiceQuadrata(double numero, int maxIterazioni, double
precisione) {
    if (numero < 0) {
        cout << "Errore: impossibile calcolare la radice quadrata di un numero
negativo." << endl;
        return -1;
    }

    double stima = numero;
    int iterazione = 0;

    cout << setw(12) << "Iterazione" << setw(15) << "Stima" << setw(15) <<
"Differenza" << endl;
    cout << string(42, '-') << endl;

    while (iterazione < maxIterazioni) {
        double stimaPrecedente = stima;
        stima = (stima + numero / stima) / 2;
        double differenza = abs(stima - stimaPrecedente);

        cout << setw(12) << iterazione + 1
             << setw(15) << fixed << setprecision(6) << stima
             << setw(15) << scientific << setprecision(2) << differenza <<
endl;

        if (differenza < precisione) {
            break;
        }

        iterazione++;
    }

    return stima;
}
```

```
int main() {
    double numero;
    cout << "Inserisci un numero positivo per calcolare la sua radice
quadrata: ";
    cin >> numero;

    int maxIterazioni = 20;
    double precisione = 1e-6;

    double risultato = calcolaRadiceQuadrata(numero, maxIterazioni,
precisione);

    if (risultato >= 0) {
        cout << "\nLa radice quadrata di " << numero << " è
approssimativamente " << risultato << endl;
    }

    return 0;
}
```

Ricerca dicotomica

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int ricercaDicotomica(const vector<int>& vettore, int obiettivo) {
    int sinistra = 0;
    int destra = vettore.size() - 1;

    while (sinistra <= destra) {
        int medio = sinistra + (destra - sinistra) / 2;

        if (vettore[medio] == obiettivo) {
            return medio;
        }

        if (vettore[medio] < obiettivo) {
            sinistra = medio + 1;
        } else {
            destra = medio - 1;
        }
```

```
    }

    return -1;  // Elemento non trovato
}

int main() {
    vector<int> numeri = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};

    cout << "Vettore ordinato: ";
    for (int num : numeri) {
        cout << num << " ";
    }
    cout << endl;

    int obiettivo;
    cout << "Inserisci il numero da cercare: ";
    cin >> obiettivo;

    int risultato = ricercaDicotomica(numeri, obiettivo);

    if (risultato != -1) {
        cout << "Il numero " << obiettivo << " è stato trovato all'indice " <<
risultato << endl;
    } else {
        cout << "Il numero " << obiettivo << " non è presente nel vettore" <<
endl;
    }

    return 0;
}
```

Bubble sort

```
#include <iostream>
#include <vector>

using namespace std;

// Passaggio per valore
void stampaVettore(vector<int> vettore) {
    for (int num : vettore) {
        cout << num << " ";
    }
    cout << endl;
```

```
}

// Passaggio per riferimento
void bubbleSort(vector<int>& vettore) {
    int n = vettore.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (vettore[j] > vettore[j + 1]) {
                // Scambia gli elementi
                int temp = vettore[j];
                vettore[j] = vettore[j + 1];
                vettore[j + 1] = temp;
            }
        }

        // Stampa lo stato del vettore dopo ogni passata
        cout << "Passata " << i + 1 << ": ";
        stampaVettore(vettore);
    }
}

int main() {
    vector<int> numeri = {64, 34, 25, 12, 22, 11, 90};

    cout << "Vettore originale: ";
    stampaVettore(numeri);

    bubbleSort(numeri);

    cout << "Vettore ordinato: ";
    stampaVettore(numeri);

    return 0;
}
```

Esempio completo catalogo libri

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <iomanip>
#include <cmath>
```

```cpp
using namespace std;

struct Libro {
    string titolo;
    string autore;
    int annoPubblicazione;
    double prezzo;
};

// Funzione per calcolare la radice quadrata con il metodo di Newton
double calcolaRadiceQuadrata(double numero, int maxIterazioni = 20, double
precisione = 1e-6) {
    if (numero < 0) return -1;

    double stima = numero;
    for (int i = 0; i < maxIterazioni; i++) {
        double stimaPrecedente = stima;
        stima = (stima + numero / stima) / 2;
        if (abs(stima - stimaPrecedente) < precisione) break;
    }
    return stima;
}

// Funzione di confronto per l'ordinamento
bool confrontaLibri(const Libro& a, const Libro& b) {
    if (a.autore != b.autore) return a.autore < b.autore;
    if (a.annoPubblicazione != b.annoPubblicazione) return a.annoPubblicazione
< b.annoPubblicazione;
    return a.titolo < b.titolo;
}

// Algoritmo di ordinamento quicksort
void quicksort(vector<Libro>& libri, int inizio, int fine) {
    if (inizio >= fine) return;

    int pivot = inizio;
    int i = inizio + 1;
    int j = fine;

    while (i <= j) {
        while (i <= fine && confrontaLibri(libri[i], libri[pivot])) i++;
        while (j > inizio && !confrontaLibri(libri[j], libri[pivot])) j--;
        if (i < j) swap(libri[i], libri[j]);
    }

    swap(libri[pivot], libri[j]);
```

```
    quicksort(libri, inizio, j - 1);
    quicksort(libri, j + 1, fine);
}

// Ricerca dicotomica
int ricercaDicotomica(const vector<Libro>& libri, const string& autore) {
    int sinistra = 0;
    int destra = libri.size() - 1;

    while (sinistra <= destra) {
        int medio = sinistra + (destra - sinistra) / 2;
        if (libri[medio].autore == autore) return medio;
        if (libri[medio].autore < autore) sinistra = medio + 1;
        else destra = medio - 1;
    }

    return -1;
}

// Funzione per stampare un libro
void stampaLibro(const Libro& libro) {
    cout << setw(30) << left << libro.titolo
         << setw(20) << left << libro.autore
         << setw(10) << right << libro.annoPubblicazione
         << setw(10) << right << fixed << setprecision(2) << libro.prezzo <<
endl;
}

// Funzione per stampare il catalogo
void stampaCatalogo(const vector<Libro>& libri) {
    cout << setw(30) << left << "Titolo"
         << setw(20) << left << "Autore"
         << setw(10) << right << "Anno"
         << setw(10) << right << "Prezzo" << endl;
    cout << string(70, '-') << endl;
    for (const auto& libro : libri) {
        stampaLibro(libro);
    }
}

// Funzione per aggiungere un libro
void aggiungiLibro(vector<Libro>& libri) {
    Libro nuovoLibro;
    cout << "Inserisci il titolo: ";
    cin.ignore();
    getline(cin, nuovoLibro.titolo);
```

```
    cout << "Inserisci l'autore: ";
    getline(cin, nuovoLibro.autore);
    cout << "Inserisci l'anno di pubblicazione: ";
    cin >> nuovoLibro.annoPubblicazione;
    cout << "Inserisci il prezzo: ";
    cin >> nuovoLibro.prezzo;

    libri.push_back(nuovoLibro);
    cout << "Libro aggiunto con successo!" << endl;
}

// Funzione per calcolare statistiche
void calcolaStatistiche(const vector<Libro>& libri) {
    if (libri.empty()) {
        cout << "Il catalogo è vuoto." << endl;
        return;
    }

    double sommaPrezzo = 0;
    int annoMin = libri[0].annoPubblicazione;
    int annoMax = libri[0].annoPubblicazione;

    for (const auto& libro : libri) {
        sommaPrezzo += libro.prezzo;
        annoMin = min(annoMin, libro.annoPubblicazione);
        annoMax = max(annoMax, libro.annoPubblicazione);
    }

    double prezzoMedio = sommaPrezzo / libri.size();
    double deviazioneStandard = 0;

    for (const auto& libro : libri) {
        deviazioneStandard += pow(libro.prezzo - prezzoMedio, 2);
    }
    deviazioneStandard = calcolaRadiceQuadrata(deviazioneStandard /
libri.size());

    cout << "Statistiche del catalogo:" << endl;
    cout << "Numero di libri: " << libri.size() << endl;
    cout << "Prezzo medio: €" << fixed << setprecision(2) << prezzoMedio <<
endl;
    cout << "Deviazione standard dei prezzi: €" << fixed << setprecision(2) <<
deviazioneStandard << endl;
    cout << "Anno di pubblicazione più vecchio: " << annoMin << endl;
    cout << "Anno di pubblicazione più recente: " << annoMax << endl;
}
```

```
int main() {
    vector<Libro> catalogo = {
        {"Il nome della rosa", "Umberto Eco", 1980, 12.99},
        {"1984", "George Orwell", 1949, 10.99},
        {"Cento anni di solitudine", "Gabriel Garcia Marquez", 1967, 14.99},
        {"Il piccolo principe", "Antoine de Saint-Exupéry", 1943, 8.99},
        {"Orgoglio e pregiudizio", "Jane Austen", 1813, 9.99}
    };

    int scelta;
    do {
        cout << "\n--- Gestione Catalogo Libri ---" << endl;
        cout << "1. Visualizza catalogo" << endl;
        cout << "2. Aggiungi libro" << endl;
        cout << "3. Ordina catalogo" << endl;
        cout << "4. Cerca libro per autore" << endl;
        cout << "5. Calcola statistiche" << endl;
        cout << "0. Esci" << endl;
        cout << "Scelta: ";
        cin >> scelta;

        switch (scelta) {
            case 1:
                stampaCatalogo(catalogo);
                break;
            case 2:
                aggiungiLibro(catalogo);
                break;
            case 3:
                quicksort(catalogo, 0, catalogo.size() - 1);
                cout << "Catalogo ordinato con successo!" << endl;
                break;
            case 4: {
                string autore;
                cout << "Inserisci il nome dell'autore da cercare: ";
                cin.ignore();
                getline(cin, autore);
                int indice = ricercaDicotomica(catalogo, autore);
                if (indice != -1) {
                    cout << "Libro trovato:" << endl;
                    stampaLibro(catalogo[indice]);
                } else {
                    cout << "Nessun libro trovato per l'autore specificato."
<< endl;
                }
```

```
                break;
            }
            case 5:
                calcolaStatistiche(catalogo);
                break;
            case 0:
                cout << "Grazie per aver usato il sistema di gestione del
catalogo libri!" << endl;
                break;
            default:
                cout << "Scelta non valida. Riprova." << endl;
        }
    } while (scelta != 0);

    return 0;
}
```

Operazioni con vettori

Questo programma dimostra diverse operazioni e algoritmi comuni sui vettori:

1. Generazione di un vettore casuale
2. Stampa di un vettore
3. Ricerca del massimo e del minimo
4. Calcolo della somma e della media
5. Rimozione dei duplicati
6. Calcolo della moda
7. Calcolo della mediana
8. Inversione di un vettore
9. Rotazione di un vettore
10. Ricerca del secondo elemento più grande

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>
#include <random>
#include <iomanip>

using namespace std;

// Funzione per stampare un vettore
```

```cpp
void stampaVettore(const vector<int>& v, const string& nome) {
    cout << nome << ": ";
    for (int num : v) {
        cout << num << " ";
    }
    cout << endl;
}

// Funzione per generare un vettore di numeri casuali
vector<int> generaVettoreCasuale(int dimensione, int min, int max) {
    vector<int> v(dimensione);
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> distrib(min, max);

    generate(v.begin(), v.end(), [&]() { return distrib(gen); });
    return v;
}

// Funzione per trovare il massimo e il minimo
pair<int, int> trovaMaxMin(const vector<int>& v) {
    auto [min, max] = minmax_element(v.begin(), v.end());
    return {*min, *max};
}

// Funzione per calcolare la somma e la media
pair<int, double> calcolaSommaMedia(const vector<int>& v) {
    int somma = accumulate(v.begin(), v.end(), 0);
    double media = static_cast<double>(somma) / v.size();
    return {somma, media};
}

// Funzione per rimuovere i duplicati
vector<int> rimuoviDuplicati(vector<int>& v) {
    sort(v.begin(), v.end());
    auto last = unique(v.begin(), v.end());
    v.erase(last, v.end());
    return v;
}

// Funzione per trovare la moda
int trovaModa(const vector<int>& v) {
    vector<int> copia = v;
    sort(copia.begin(), copia.end());
    int numero_corrente = copia[0];
    int conteggio_corrente = 1;
```

```
    int moda = copia[0];
    int conteggio_max = 1;

    for (size_t i = 1; i < copia.size(); ++i) {
        if (copia[i] == numero_corrente) {
            conteggio_corrente++;
        } else {
            if (conteggio_corrente > conteggio_max) {
                conteggio_max = conteggio_corrente;
                moda = numero_corrente;
            }
            numero_corrente = copia[i];
            conteggio_corrente = 1;
        }
    }

    if (conteggio_corrente > conteggio_max) {
        moda = numero_corrente;
    }

    return moda;
}

// Funzione per calcolare la mediana
double trovaMediana(vector<int> v) {
    sort(v.begin(), v.end());
    size_t size = v.size();
    if (size % 2 == 0) {
        return (v[size/2 - 1] + v[size/2]) / 2.0;
    } else {
        return v[size/2];
    }
}

// Funzione per invertire il vettore
void invertiVettore(vector<int>& v) {
    reverse(v.begin(), v.end());
}

// Funzione per ruotare il vettore
void ruotaVettore(vector<int>& v, int k) {
    k = k % v.size();
    rotate(v.begin(), v.begin() + k, v.end());
}

// Funzione per trovare il secondo elemento più grande
```

```cpp
int trovaSecondoMassimo(const vector<int>& v) {
    if (v.size() < 2) return -1; // o un altro valore che indica errore
    vector<int> copia = v;
    sort(copia.begin(), copia.end(), greater<int>());
    auto it = unique(copia.begin(), copia.end());
    if (distance(copia.begin(), it) < 2) return -1; // non c'è un secondo
elemento unico
    return copia[1];
}

// Funzione principale per testare tutte le operazioni
int main() {
    vector<int> numeri = generaVettoreCasuale(20, 1, 100);
    stampaVettore(numeri, "Vettore originale");

    cout << "\nOperazioni sul vettore:\n";

    auto [min, max] = trovaMaxMin(numeri);
    cout << "Minimo: " << min << ", Massimo: " << max << endl;

    auto [somma, media] = calcolaSommaMedia(numeri);
    cout << "Somma: " << somma << ", Media: " << fixed << setprecision(2) <<
media << endl;

    vector<int> senzaDuplicati = rimuoviDuplicati(numeri);
    stampaVettore(senzaDuplicati, "Vettore senza duplicati");

    int moda = trovaModa(numeri);
    cout << "Moda: " << moda << endl;

    double mediana = trovaMediana(numeri);
    cout << "Mediana: " << mediana << endl;

    invertiVettore(numeri);
    stampaVettore(numeri, "Vettore invertito");

    ruotaVettore(numeri, 3);
    stampaVettore(numeri, "Vettore ruotato di 3 posizioni");

    int secondoMax = trovaSecondoMassimo(numeri);
    cout << "Secondo massimo: " << secondoMax << endl;

    return 0;
}
```