Esercitazione 6

Classi

3-4-5 dicembre 2018

Corso di Laboratorio 1A, A.A. 18-19
Laurea triennale in Fisica, Università di Genova

Obiettivi formativi: familiarizzare con la scrittura di semplici classi di C++ e con il loro utilizzo all'interno di programmi, mediante applicazioni a problemi di fisica di base.

Nell'esercizio 1, da svolgere in laboratorio, si chiede di implementare una classe per la descrizione dei vettori geometrici in 3 dimensioni, e di verificarne la funzionalità con un programma di test.

Nell'esercizio 2, da svolgere in laboratorio, si chiede di realizzare un programma in grado di calcolare il campo gravitazionale generato da un insieme di punti materiali di massa e posizione note e lette da file.

Nell'esercizio 3, da svolgere a casa, si chiede di implementare uno degli algoritmi numerici mostrati a lezione, tra calcolo di integrali, ricerca di zeri ed inversione di matrici.

- Svolgete gli esercizi secondo la traccia riportata nel resto di questo documento
- Seguite le istruzioni riportate qui per la consegna degli esercizi di laboratorio
- IMPORTANTE: la consegna deve essere effettuata da entrambi i componenti del gruppo

Esercizio 1

Considerare i file Vettore.h e Vettore.cpp. Il primo contiene la definizione della classe, il secondo la sua parziale implementazione.

In particolare sono definiti nel file .h ed implementati nel file .cpp i seguenti metodi:

1. Costruttore di default
2. Costruttore con inizializzazione
3. Distruttore
4. “Setter” e “Getter” per leggere e scrivere le tre componenti del vettore (implementati inline)
5. operator+(const Vettore& other) per fare la somma tra due vettori
6. Print() per la stampa su terminale nel formato “(x, y, z)”
7. operator-() per ottenere il negato di un vettore

Sono definiti nel file .h ma NON implementati nel file .cpp i seguenti metodi

1. operator-(const Vettore& other) per fare la sottrazione tra due vettori
2. operator*(const Vettore& other) per fare il prodotto scalare con un altro vettore
3. operator*(double scalar) per fare il prodotto tra un vettore ed un numero

Infine il metodo Modulo() per il calcolo del modulo del vettore, non è definito nel file .h né implementato nel file .cpp

Implementare i metodi definiti ma non implementati, ed il metodo Modulo(), e verificate il funzionamento della classe Vettore con il programma test_vettore.cpp. Aggiungere i seguenti test a quelli già presenti nel programma:

1. prodotto di un vettore per un numero
2. modulo di un vettore
3. prodotto scalare con un altro vettore

Compilare il programma con il comando: g++ -o test_vettore test_vettore.cpp Vettore.cpp

Materiale da consegnare: test_vettore.cpp, Vettore.h, Vettore.cpp

Esercizio 2

Realizzare un programma che calcoli il campo gravitazionale generato in un punto P dall'insieme di N punti materiali le cui masse e posizioni sono contenute nel file PuntiMateriali.dat, che ha il formato (masse in kg, posizioni in m)

m_1	x_1	y_1	z_1
…	…	…	…
m_N	x_N	y_N	z_N

Input del programma: file PuntiMateriali.dat
Output del programma: campo gravitazionale calcolato nel punto P = (100, 0, 0) m.

Schematizzazione fisica del problema

Il campo gravitazionale totale **g** nel punto P si ottiene usando il principio di sovrapposizione: esso sarà dato dalla somma dei campi gravitazionali $\mathbf{g}_i$ generati da ciascun punto materiale. Indichiamo con $\mathbf{r}_P$ il vettore di coordinate del punto P in cui vogliamo calcolare il campo gravitazionale. Un punto materiale, di massa m_i e posto nel punto di coordinate $\mathbf{r}_i$, genera in $\mathbf{r}_P$ un campo gravitazionale $\mathbf{g}_i$ dato da

$$\vec{\mathbf{g}}_i = -\mathbf{G}\frac{m_i}{|\vec{\Delta r}_i|^3}\vec{\Delta r}_i$$

in cui $\vec{\Delta r}_i = \vec{r}_P - \vec{r}_i$ identifica il vettore che va da dall'i-esimo punto materiale al punto P e $G=6.67\cdot10^{-11}$ $m^3/(kg\ s^2)$.

Schematizzazione del programma

I punti materiali possono essere rappresentati mediante una classe PuntoMateriale, definita e parzialmente implementata nei file PuntoMateriale.h e PuntoMateriale.cpp. Questa classe ha come data member un double per rappresentare la massa, ed un Vettore (lo stesso dell'esercizio 1) per la rappresentazione della posizione.
All'interno della classe PuntoMateriale, implementare il metodo CalcolaCampo(), che calcoli il campo gravitazionale generato dal punto materiale in un generico punto P, secondo la formula discussa in precedenza. Le coordinate del punto P, passate come parametro al metodo, sono rappresentate da un oggetto della classe Vettore. Il metodo CalcolaCampo() ritorna anch'esso un Vettore, contenente le coordinate del campo in N/kg.
All'interno del programma prinicipale, test_campo.cpp, si dovra' leggere il file PuntiMateriali.dat, e per ogni punto trovato al suo interno si dovra' creare un oggetto della classe PuntoMateriale. Per ciascuno di questi oggetti, si procede quindi a calcolare il campo nel punto P mediante chiamata al metodo CalcolaCampo() e si sommano i vettori ottenuti fino ad ottenere il valore totale del campo nel punto P richiesto.

Controlli

- Usando il programma per calcolare il campo nel punto P = (100, 0, 0) m, verificare di ottenere il valore $\mathbf{g}_{TOT}$ = (-1.127e-08, -3.228e-11, -7.789e-11).
- Usare il metodo Print() della classe Vettore, stampare la posizione dei punti materiali letti da file, e verificare che sono tutti in posizioni vicine all'origine rispetto alla posizione del punto P. In altre parole, $|\mathbf{r}_i|/|\mathbf{r}_P| << 1$, per tutti gli $\mathbf{r}_i$. Data questa condizione, ci si aspetta che il campo calcolato dal programma sia simile a quello che si otterrebbe se tutti i punti materiali fossero concentrati nell'origine. Verificare questa affermazione.
- Verificare che, dato un punto materiale con m= M_T, il valore del campo calcolato in un punto che disti r=R_T dal punto materiale stesso, valga in modulo g=9.81 m/s (M_T=5.97e24 kg e R_T=6370 km sono rispettivamente la massa e il raggio della Terra).

Compilare il programma con il comando: g++ -o test_campo test_campo.cpp Vettore.cpp PuntoMateriale.cpp

Materiale da consegnare: test_campo.cpp, PuntoMateriale.cpp, PuntoMateriale.h

Esercizio 3 (per casa)

Scrivete un programma che risolva uno a scelta dei seguenti problemi presentati a lezione:

1. Integrale numerico
2. ricerca di zeri di una funzione
3. soluzione di sistemi lineari

Integrale numerico

Scrivete un programma che, data una funzione f(x) e un intervallo di integrazione $[x_{MIN}, x_{MAX}]$, sia in grado di calcolare l'integrale di f(x) tra i due estremi con una certa precisione.
Argomenti in ingresso al programma (da riga di comando, letti da tastiera, etc):

1. funzione in formato di stringa (es. "2*exp(x*x/4)")
2. estremi di integrazione
3. precisione con cui si vuole che l'integrale venga calcolato

Materiale da consegnare: main_integrale.cpp, ed eventuali file con classi

Ricerca di zeri

Scrivere un programma che, utilizzando il metodo di bisezione, calcoli il valore di x=x* per cui f(x*) = 0, con $x_{MIN} < x < x_{MAX}$ (assumere che la funzione abbia uno e un solo zero tra x_{MIN} e x_{MAX}).
Argomenti in ingresso al programma (da riga di comando, letti da tastiera, etc):

1. funzione in formato di stringa (es: "1/x - 2/(x*x)")
2. estremi tra i quali cercare gli zeri
3. precisione con cui si vogliono identificare gli zeri

Materiale da consegnare: main_zeri.cpp, ed eventuali file con classi

Soluzione di sistemi lineari / calcolo inversa di una matrice (piu' difficile)

Partite da queste tracce: Matrice.h, Matrice.cpp e test_matrice.cpp. Usate come test i file sistlin10.dat e magic9.dat.
Implementare i metodi vuoti in fondo a Matrice.cpp e poi implementare il programma di test test_matrice.cpp per leggere uno dei due file e calcolare la soluzione di un sistema lineare (file sistlin10.dat) o l'inversa di una matrice (file magic9.dat)

Materiale da consegnare: test_matrice.cpp, Matrice.h, Matrice.cpp

Scadenza per la consegna (tramite AulaWeb: 2018-12-14)