Elaborato di Algoritmi e Strutture Dati

Calcolo della derivata di una funzione

Programmatori: *Eugenio Severi - Mauro Valmori - Ludovico de Nittis*

Introduzione

Il progetto consiste in un software che, data una funzione ad una variabile in input, ne calcola la derivata prima. Abbiamo scelto di non limitarci all'uso del solo C, bensì di utilizzare il C++, principalmente a causa dell'esistenza del tipo *string* e di funzioni specifiche per le stringhe in quest'ultimo, che hanno consentito una stesura più agevole del codice. Il programma è composto da tre file sorgenti al fine di organizzare in modo più chiaro le funzioni:

- Main.cpp --> Contiene la funzione di avvio del programma e l'interfaccia grafica;
- Functions.h --> Header contenente le dichiarazioni delle funzioni personalizzate utilizzate nel programma;
- Functions.cpp --> Contiene il codice di tutte le funzioni utilizzate dal programma per gestire l'I/O su disco e il calcolo della derivata vero e proprio.

Notazione

Tutte le funzioni matematiche devono essere espresse tramite notazione prefissa secondo la sintassi qui specificata. Tutte le funzioni richiedono uno o due parametri in ingresso.

- Sen(x) => Funzione trigonometrica seno
- Cos(x) => Funzione trigonometrica coseno
- Times(x,y) => Operatore matematico moltiplicativo, corrispondente a x*y
- Pow(x,y) => Funzione potenza, corrispondente a x^y
- Plus(x,y) => Operatore matematico additivo, corrispondente a x+y
- Minus(x,y) => Operatore matematico sottrativo, corrispondente a x-y
- Div(x,y) => Operatore matematico divisivo, corrispondente a x/y

I parametri di tali funzioni possono essere espressioni semplici (costanti oppure *x*), oppure funzioni a loro volta. Con questo metodo è possibile eseguire operazioni complesse non definite tra le funzioni di base utilizzando le funzioni a cascata. Ad esempio se si desidera rappresentare la somma di tre valori x,y,z, si può supplire alla mancanza di una funzione sommatrice a tre parametri sommando i numeri a gruppi di due, grazie alla proprietà associativa (x+y+z=>Plus(Plus(x,y),z)).

Se si desidera rappresentare un valore negativo, lo si può fare utilizzando la funzione Minus con la seguente notazione: Minus(0,x), dove x è il valore assoluto di tale valore.

Input/Output

L'utente può scegliere tramite menù la modalità di input e output tra:

- file di testo *input.txt/output.txt* (nome predefinito) contenuto nella cartella di esecuzione del programma;
- file specificato dall'utente tramite percorso completo o solo con il nome (in questo caso verrà considerata implicitamente la cartella di esecuzione del programma);
- inserimento da tastiera direttamente nella finestra del programma.

Nel caso l'utente specifichi manualmente il percorso, occorre utilizzare il simbolo "/" invece di "\" per separare i nomi delle cartelle.

Della lettura del file di input si occupa la funzione *bool read_file* (tramite *ifstream*), che richiede due parametri di tipo *string* in input:

- *input_string*, passato per riferimento, all'interno del quale verrà memorizzato il contenuto del file;
- *input_file*, passato per valore, contenente il percorso completo del file di input (oppure solo il suo nome: in questo caso verrà implicitamente scelta la cartella di esecuzione del programma).

Restituisce 1 nel caso in cui si siano verificati errori nell'apertura del file, 0 altrimenti.

Della scrittura del file di output si occupa la funzione *bool write_file* (tramite *ofstream*), che richiede due parametri di tipo *string* in input:

- *output_string*, passato per valore, contenente il risultato dell'elaborazione dell'input, ovvero la derivata della funzione iniziale;
- *output_file*, passato per valore, contenente il percorso completo del file di output (oppure solo il suo nome: in questo caso verrà implicitamente scelta la cartella di esecuzione del programma).

Restituisce 1 nel caso in cui si siano verificati errori nell'apertura del file, 0 altrimenti.

Funzioni accessorie

void getNumber -> Funzione utilizzata nei menù, nei quali all'utente è richiesto di inserire il numero corrispondente alla voce desiderata. Richiede in input un intero (*choice*) passato per riferimento, in cui viene memorizzata la scelta dell'utente.

size_t find_comma -> Funzione utilizzata dalla funzione di derivazione per individuare la posizione della virgola (se presente) che separa i due parametri di una funzione in forma prefissa. Richiede in input la stringa *input_string* da esaminare. Restituisce 0 se non sono presenti virgole oppure se sono presenti più parentesi chiuse che aperte, altrimenti l'indice a cui si trova.

Calcolo della derivata

Del calcolo della derivata si occupa la funzione "divide et impera" *bool derivative* (descritta dettagliatamente nel paragrafo successivo) con struttura bottom-up, in cui si costruisce l'output finale a partire da casi base semplici. La funzione che richiede due parametri di tipo *string* in ingresso:

- *input_string*, passato per valore, contenente l'espressione matematica da derivare posta in forma prefissa;
- *output_string*, passato per riferimento, in cui verrà memorizzato il risultato dell'elaborazione della funzione di input, ovvero la sua derivata prima posta in notazione prefissa.

La funzione è ricorsiva, e restituisce 0 se non si verificano errori, altrimenti 1. In questo modo è possibile per l'utente verificare in maniera immediata la presenza di alcuni errori nell'input. La funzione viene eseguita inizialmente passandole l'input completo e, ad ogni passo della ricorsione, entra in profondità nell'espressione, tanto più complessa quanto più radicate le funzioni. Il caso base viene raggiunto quando si giunge ad un livello in cui sono presenti solo espressioni semplici (costanti oppure *x*). Infatti la funzione *derivative* è in grado di eseguire solo operazioni a due operandi semplici: nel caso incontri un parametro complesso (ad esempio una funzione), l'esecuzione del livello corrente viene sospesa per semplificare tale parametro.

Funzione *derivative()*

Inizialmente vengono gestiti i casi base, cioè quelli in cui l'input non contiene funzioni e, qualora ci si trovi di fronte ad uno di questi casi (costante oppure *x*), viene calcolata direttamente la derivata. Nel caso siano presenti funzioni, viene eseguita la ricerca di una parentesi chiusa in fondo alla stringa: se non venisse trovata, ciò indicherebbe un errore nell'input. In caso di successo si procede con l'individuazione delle funzioni matematiche tramite una serie di *if* che confrontano una sottostringa dell'input di lunghezza pari a quella dei nomi delle funzioni con le funzioni supportate dal programma (i nomi delle funzioni possono essere indicati con la prima lettera maiuscola oppure con tutte le lettere minuscole). Se non viene trovata una corrispondenza significa che è stata utilizzata una funzione non supportata dal programma oppure che si è verificato un errore di input.

Le funzioni Sen() e Cos() possono essere calcolate direttamente applicando la formula di derivazione, ma abbiamo inserito due controlli aggiuntivi che consentono di semplificare l'espressione in caso la derivata assuma valori particolari (0 o 1).

La funzione Times() è invece più complessa. Essendo una funzione a due parametri, per prima cosa occorre individuare la virgola tramite la funzione *find_comma*. In questo modo è possibile isolare i due parametri della funzione per poterli trattare individualmente. Poi è necessario memorizzare in due variabili temporanee le derivate di entrambi gli operandi, in quanto richieste per la formula di derivazione della moltiplicazione di funzioni. Una volta calcolate, vengono eseguiti dei controlli sui casi semplici in cui è possibile semplificare l'espressione (una o entrambe le derivate nulle oppure uguali a 1), altrimenti su utilizza la formula completa. Analogamente si procede per la funzione Div(), all'interno della quale è previsto un ulteriore controllo per evitare divisioni per zero.

La funzione Pow() richiede anch'essa l'individuazione degli argomenti tramite *find_comma*. Inizialmente vengono controllati i casi semplici (esponente uguale a 0, 1 o 2 o base uguale a costante) e, in caso non sia possibile attuare semplificazioni, si passa all'applicazione della regola di derivazione della potenza. Poiché tale regola prevede di decrementare l'esponente, è necessario convertire temporaneamente tale numero da stringa a intero per eseguire la sottrazione.

Le funzioni Plus() e Minus() sono analoghe, poiché la sottrazione può essere considerata come parte della somma algebrica. In entrambi casi, dopo l'individuazione della virgola (e di conseguenza dei parametri) tramite *find_comma*, vengono calcolate le derivate degli argomenti. Al fine di semplificare l'output vengono eseguiti controlli nel caso in cui le derivate siano uguali a 0. Se non è possibile effettuare semplificazioni, viene applicata la regola di derivazione della somma/differenza.

Prestazioni

Le funzioni di input e di output, sia su disco che all'interno del programma, hanno un costo lineare, dipendendo in maniera diretta dalla lunghezza dell'input. La quasi totalità del costo computazionale del programma è dato dalla funzione *derivative()*, la quale opera nel seguente modo:

- il programma prende in ingresso una stringa di lunghezza *n* (con *n* uguale al numero dei caratteri che la compongono);
- viene individuata la prima funzione della stringa, il cui nome è composto da un numero di caratteri *k*, variabile da 4 a 6 a seconda della funzione (questo valore comprende la parentesi aperta che separa il nome della funzione dal primo parametro della stessa);
- per individuare l'operatore vengono eseguiti fino a 7 confronti,dal costo lineare (costante*n) dovuto alla funzione *length*;
- Al fine di recuperare la posizione della virgola (separatore degli argomenti), vengono analizzati fino a *n-k* caratteri (costo sempre lineare dovuto alla funzione length).

La procedura descritta viene ripetuta ricorsivamente su ognuno dei due parametri lunghi rispettivamente *c* e *n-k-c*, con *c* uguale alla posizione della virgola (quindi al numero di caratteri controllati).

Analisi del caso medio e ottimo, nel quale la virgola tenderà a trovarsi nella posizione *(n-k)/2*. In questo caso il tempo di esecuzione è espresso dalla formula:

$T(n) = n + (n-k) + 2T((n-k)/2)$, con k costante

Semplificando si ottiene:

$T(n) = n + 2T(n/2) = \Theta(n/2 * \log(n)) =$ **$\Theta(n*\log(n))$**

Analisi del caso pessimo, nel quale la virgola si troverà sempre nella posizione *n-c* (con $c << n$). In questo caso il tempo di esecuzione è espresso dalla formula:

$T(n) = n + n-c + T(c) + T(n-k-c)$, con *k* e *c* costanti

Semplificando si ottiene:

$T(n) = n + T(n-c) = \Theta(n * n) =$ **$\Theta(n^2)$**

È opportuno osservare che tipicamente ci si ritroverà nel caso medio, poiché è improbabile che le espressioni in input siano tali da ricadere sempre o quasi nel caso pessimo, a meno che non siano realizzate artificialmente a questo scopo.