Anna Delle Piane (5565836), Marco Zoratti (5562866).

Relazione Labo 1 ALAN

Il codice del laboratorio è suddiviso in 3 parti: funzioni ausiliare, esercizi e main.

Il file .cpp può essere visionato al seguente link: [Uni-Informatica/SecondoAnno/ALAN/labo1/labo.cpp at master · marchacio/Uni-Informatica (github.com)](https://github.com/marchacio/Uni-Informatica/blob/master/SecondoAnno/ALAN/labo1/labo.cpp)

# Funzioni ausiliarie

Le seguenti funzioni sono state sviluppate ed utilizzate per semplificare e riutilizzare più volte il codice all’interno del programma:

* ***void stampaElemTabella e void stampaDivisoreTabella***

Funzioni ausiliarie per migliorare l’output e renderlo più leggibile e facilitarne la comprensione.

* ***double fattoriale(double n)***

Funzione ausiliaria per calcolare il fattoriale del numero passato come argomento.

* ***double f (double x)***

Rinominazione della funzione esponenziale exp (ex) ottenuta dalla libreria cmath.

* ***double fNx(double n, double x)***

Funzione ricorsiva per ottenere il Polinomio di Taylor:

La ricorsività di questa funzione è dovuta alla presenza della sommatoria:

* + Il caso base della ricorsione avviene quando il grado del polinomio (n) è uguale a 0, infatti quando questa condizione è vera e , ovvero .
  + Altrimenti il calcolo della frazione è seguito dalla chiamata ricorsiva della funzione con argomenti *n-1* e *x*.

# Esercizio 1

Definiamo d0 e d1 come l’ultima e la penultima cifra del numero matricola 5565836, quindi rispettivamente 6 e 3.

Dato 0 < i <= 6:

a = (d0+1) \* 10i

b = (d1+1) \* 1020

c = -b

## Implementazione algoritmo

Una volta definito il numero di matricola come stringa, si ottiene il carattere corrispondente all’ultima e penultima cifra grazie all’operatore [].

In c++, il tipo di dato char fa riferimento alla codifica Unicode e quindi ogni carattere è definito da un numero minore di 28 che si può ottenere grazie al casting dinamico.

Inoltre in Unicode i numeri sono definiti sequenzialmente quindi per ottenere il numero intero rappresentato da un carattere basta sottrarre a quest’ultimo il numero rappresentato dal carattere ‘0’; in questo modo convertiamo i due caratteri del codice matricola in numeri interi.

Una volta ottenute le cifre, il programma esegue un ciclo for da 0 a 6 (compresi) ed esegue i calcoli.

## Output programma:

Numero matricola: 5565836; ultime due cifre: 6, 3.

|  |  |  |
| --- | --- | --- |
|  | Algoritmo 1: (a+b)+c | Algoritmo 2: a+(b+c) |
| i = 0 | 0 | 7 |
| i = 1 | 0 | 70 |
| i = 2 | 0 | 700 |
| i = 3 | 0 | 7000 |
| i = 4 | 65536 | 70000 |
| i = 5 | 720896 | 700000 |
| i = 6 | 7.01235e+06 | 7e+06 |

## Analisi algoritmica

Algoritmo 1: (

L’algoritmo è instabile quando *s ≈ -c*, ovvero *(a + b) ≈ -c* , in quanto comporterebbe una divisione per zero e quindi un incremento infinito dell’errore .

Questo caso si verifica finche *a* non diventa abbastanza grande da influenzare la somma *a + b* in modo considerevole e quindi rende questo algoritmo instabile.

Algoritmo 2:

L’algoritmo è instabile quando *s ≈ -a* ma questo caso non può accadere perché *s* è la somma di due numeri discordi e quindi la loro somma vale 0.

Dato che *a ≠ 0* per definizione, l’algoritmo è stabile.

## Considerazioni

Come si può notare dai dati prodotti in output, il primo algoritmo è instabile mentre il secondo è stabile e riporta i risultati corretti.

Questa differenza è dovuta alla precedenza delle addizioni e cancellazioni.

In un primo momento, l'algoritmo 1 esegue una somma tra due numeri di ordine molto differente, *a e b*, fintantoché il valore di *i* è piccolo. Infatti, se *i < 3*, *a* è al massimo *n^3*, mentre *b* è sempre *m^20*.

L'influenza di *a* nella somma inizia a farsi sentire quando la differenza di ordine tra le due variabili è minore o uguale a 16 gradi. Ciò è dovuto al fatto che *a* e *b* sono entrambi numeri double, con una precisione di 2^16. Pertanto, per i valori di *i* fino a 3, l'output dell'algoritmo è 0.

Quando *a* inizia a "modificare" *b*, in ogni caso è ancora troppo piccolo per renderlo significativamente diverso da *c*. Di conseguenza, si verifica una cancellazione tra *(a+b)* e *c* che fa degenerare i risultati.

A differenza del primo, l’algoritmo 2 esegue una somma tra due numeri discordi che hanno quindi risultato 0 per poi sommare *a* e ritornare il risultante corretto.

# Esercizio 3

Implementazione di un programma che determini la precisione di macchina in singola e doppia precisione.

## Implementazione algoritmo

Una volta definita la variabile *float* *eps* inizializzata ad 1*,* esegue un ciclo while che la dimezza fino a quando non saràabbastanza piccola da poter essere semplificata dal calcolatore.

Lo stesso codice è stato utilizzato per la variabile in doppia precisione *double eps2 = 1*.

## Output programma:

*Precisione di macchina (eps) in singola precisione: 5.96046e-08*

*Precisione di macchina (eps) in doppia precisione: 1.11022e-16*

## Considerazioni