Laboratorio Errori

Samuele Barrago [5703117], Daniele Sacco [5616921], Lorenzo Livio Vaccarecci [5462843]

1 Primo esercizio

In questo esercizio vogliamo calcolare tramite la matricola del primo componente in ordine alfabetico, due semplici operazioni aritmetiche: $a + (b + c)$ e $(a + b) + c$ utilizzando variabili di tipo double.

1.1 Calcolo di a, b, c

Usando la matricola 5703117, indichiamo con d_0 l'ultima cifra della matricola e con d_1 la penultima.

$$a = (d_0 + 1) \cdot 10^i \text{ con } i = 0, 1, \ldots, 6$$
$$b = (d_1 + 1) \cdot 10^{20}$$
$$c = -b$$

1.2 Analisi dei risultati

i	**a**	**b**	**c**	**(a+b)+c**	**a+(b+c)**
0	$6 \cdot 10^0$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	0	6
1	$6 \cdot 10^1$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	0	60
2	$6 \cdot 10^2$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	0	600
3	$6 \cdot 10^3$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	0	6000
4	$6 \cdot 10^4$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	0	60000
5	$6 \cdot 10^5$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	655360	600000
6	$6 \cdot 10^6$	$8 \cdot 10^{20}$	$-8 \cdot 10^{20}$	$6.03 \cdot 10^6$	$6 \cdot 10^6$

Sappiamo che per il `double` di C/C++ la precisione è

$$u = \frac{1}{2} \cdot \left(B^{-t+1}\right) = \frac{1}{2} \cdot \left(2^{-51}\right) \simeq 2.22044605 \cdot 10^{-16}$$

Quindi possiamo dire che a per essere "sentita" nelle operazioni deve essere **almeno** $u \cdot b \simeq 177635.684$ infatti fino all'iterazione $i = 4$ a non viene usata nel calcolo restituendo quindi come risultato 0. Per quanto riguarda l'iterazione $i = 5$ e $i = 6$, il risultato che ci aspettiamo è quello che viene restituito da $a + (b + c)$ però i risultati dati da $(a + b) + c$ sono diversi e meno precisi in quanto a viene "sentita" dall'operazione ma non è abbastanza grande causando l'effetto di **cancellazione** che amplifica il coefficiente di amplificazione degli errori che amplifica, a sua volta, l'errore.

2 Secondo esercizio

Consideriamo come valore "corretto" il valore restituito dalla funzione `exp()` della libreria standard di C++.

- Errore assoluto: $|Taylor - exp|$
- Errore relativo: $\left|\frac{Taylor-exp}{exp}\right|$

Alg	x	N	**Taylor**	**exp()**	**Errore assoluto**	**Errore relativo**
1	0.5	3	1.64583	1.64872	0.00289	0.00175
1	0.5	10	1.64872	1.64872	0	0
1	0.5	50	1.64872	1.64872	0	0
1	0.5	100	1.64872	1.64872	0	0
1	0.5	150	1.64872	1.64872	0	0
1	30	3	4981	$1.06865 \cdot 10^{13}$	$1.06865 \cdot 10^{13}$	0.99999
1	30	10	$2.3883 \cdot 10^{8}$	$1.06865 \cdot 10^{13}$	$1.06863 \cdot 10^{13}$	0.99998
1	30	50	$1.06833 \cdot 10^{13}$	$1.06865 \cdot 10^{13}$	$3.2 \cdot 10^{9}$	$2.99443 \cdot 10^{-4}$
1	30	100	$1.06865 \cdot 10^{13}$	$1.06865 \cdot 10^{13}$	0	0
1	30	150	$1.06865 \cdot 10^{13}$	$1.06865 \cdot 10^{13}$	0	0
1	-0.5	3	0.604167	0.606531	0.00236	0.00390
1	-0.5	10	0.606531	0.606531	0	0
1	-0.5	50	0.606531	0.606531	0	0
1	-0.5	100	0.606531	0.606531	0	0
1	-0.5	150	0.606531	0.606531	0	0
1	-30	3	-4079	$9.35762 \cdot 10^{-14}$	4079	$4.35901 \cdot 10^{16}$
1	-30	10	$1.21255 \cdot 10^{8}$	$9.35762 \cdot 10^{-14}$	121255000	$1.29579 \cdot 10^{21}$
1	-30	50	$8.78229 \cdot 10^{8}$	$9.35762 \cdot 10^{-14}$	878229000	$9.38517 \cdot 10^{21}$
1	-30	100	$-3.42134 \cdot 10^{-5}$	$9.35762 \cdot 10^{-14}$	$3.42134 \cdot 10^{-5}$	365620746.4
1	-30	150	$-3.42134 \cdot 10^{-5}$	$9.35762 \cdot 10^{-14}$	$3.42134 \cdot 10^{-5}$	365620746.4
2	-0.5	3	0.607595	0.606531	0.00106	0.00175
2	-0.5	10	0.606531	0.606531	0	0
2	-0.5	50	0.606531	0.606531	0	0
2	-0.5	100	0.606531	0.606531	0	0
2	-0.5	150	0.606531	0.606531	0	0
2	-30	3	0.000201	$9.35762 \cdot 10^{-14}$	$2.01 \cdot 10^{-4}$	$2.14798 \cdot 10^{9}$
2	-30	10	$4.187091 \cdot 10^{-9}$	$9.35762 \cdot 10^{-14}$	$4.187 \cdot 10^{-9}$	44744
2	-30	50	$9.36041 \cdot 10^{-14}$	$9.35762 \cdot 10^{-14}$	$2.79 \cdot 10^{-17}$	$2.98153 \cdot 10^{-4}$
2	-30	100	$9.35762 \cdot 10^{-14}$	$9.35762 \cdot 10^{-14}$	0	0
2	-30	150	$9.35762 \cdot 10^{-14}$	$9.35762 \cdot 10^{-14}$	0	0

I valori degli errori sono stati arrotondati a 5 cifre decimali

2.1 Analisi dei risultati

Possiamo notare che per x piccole e tutte le N per l'algoritmo 1, gli errori sono molto vicini allo zero (se non zero stesso) mentre per x grandi e N piccole, gli errori sono elevati. Per quanto riguarda l'algoritmo 2 possiamo notare che gli errori sono vicini allo zero per tutte le N e x piccole mentre per x grandi e N piccole, gli errori sono elevati. Il secondo algoritmo risulta più preciso del primo per le x negative.

3 Terzo esercizio

Il più grande numero intero positivo d tale che

$$1 + 2^{-d} > 1$$

è:

- Doppia precisione (`double`): $d = 53$
- Singola precisione (`float`): $d = 24$

Sostituendo d in 2^{-d} avremo la precisione di macchina per:

- Doppia precisione: $2^{-53} \simeq 1.11022 \cdot 10^{-16}$
- Singola precisione: $2^{-24} \simeq 5.96046 \cdot 10^{-8}$

Questo conferma che `double` è più precisa di `float` perchè utilizza più bit per rappresentare i numeri.