Laboratorio di Calcolo Canale Lp-P, Gruppo B
Esercitazione Valutata del 21/12/2021

Docenti: Prof. C. De Michele e Prof. A. Paiella

Nome _____ Cognome _____

Matricola _____ ☐ Ritirato

Per lo svolgimento di tale prova avete a disposizione 3 ore. Verranno valutati eventuali commenti e l'indentazione del codice che potranno costituire un bonus ulteriore nel punteggio finale. E' ammesso l'utilizzo degli appunti, del libro di testo e dei prontuari di python e linux. All'interno di ogni codice riportare un commento con il proprio nome, cognome e numero di matricola.

► **Prima parte:** Dopo aver fatto l'accesso al computer tramite lo username "studente" e la password "informatica", create una cartella chiamata `EX10_<NOME>_<COGNOME>` che dovrà contenere tutto quello che produrrete durante questa esercitazione. Lo scopo di questa esercitazione valutata è di scrivere un programma in C, chiamato `<nome>_<cognome>.c`, che converta un numero decimale compreso tra 1.0 e 1.9 nella sua rappresentazione binaria in virgola mobile con una mantissa a M bit (si ricorda che per mantissa s'intende la parte decimale del numero). Non si richiede di memorizzare né il segno del numero, poiché il numero è positivo, né l'esponente, poiché in questo caso specifico è sempre 0. La conversione consisterà quindi nel calcolo della mantissa del numero decimale suddetto.
Il programma dovrà richiedere in input, come un intero, il numero M, verificando che sia compreso tra 12 e 23 inclusi, ed un numero di tipo `double`, verificando che sia compreso tra 1 e 1.9. Nel caso che le verifiche abbiano esito negativo si dovrà reiterare la richiesta di immissione di tali numeri. Il programma quindi dovrà calcolare la mantissa in formato binario del numero double fornito in input memorizzandola in un array di M interi e stampando infine il risultato su schermo. L'array conterrà come primo elemento il bit meno significativo della mantissa, cioè quello relativo alla potenza 2^{-M}.
In particolare il programma dovrà essere sviluppato secondo le seguenti specifiche:

1. Nel main si dovrà dichiarare un array di 23 interi chiamato `numfp`.

2. Si dovrà implementare una funzione chiamata `immettiM` che richiederà come argomento un puntatore ad intero e che dovrà richiedere in input il numero M controllando che sia nell'intervallo di valori richiesto e reiterando la richiesta di immissione se così non fosse. Il valore di M immesso deve essere restituito alla funzione chiamante tramite il puntatore passato come argomento.

3. Si dovrà implementare una funzione chiamata `immettiNumero` che richiederà come argomento un puntatore a double e che dovrà richiedere in input una numero in virgola mobile, verificando che sia compreso tra 1.0 e 1.9 (estremi inclusi) e reiterando la richiesta di immissione qualora il numero immesso non sia come richiesto.

4. Si dovrà implementare una funzione chiamata `convertiNumero` a cui si dovranno passare come argomenti il numero `double` da convertire, l'array `numfp` e il numero M. Tale funzione convertirà il numero `double` fornito in input nella sua rappresentazione in virgola mobile (che nel presente

caso si riduce alla sola mantissa), che verrà memorizzata nell'array `numfp` di interi passato come argomento. La conversione si ottiene nel seguente modo, assumendo che `i` sia una variabile intera opportunamente dichiarata e n il numero `double` da convertire decrementato di 1 (poiché si deve convertire solo la parte frazionaria):

(a) `i=M-1`

(b) `n=n*2`

(c) Se la parte intera del numero è 0, allora `numfp[i]=0`, se invece è 1, allora `numfp[i]=1` e al numero n si deve sottrarre 1.

(d) `i=i-1`

(e) Se `i < 0` termina, altrimenti torna al punto b).

Se, dopo aver calcolato tutti gli M bit della mantissa, il valore del numero n è maggiore o uguale di 0.5, si dovrà sommare 1 al numero memorizzato nella mantissa per effettuare il necessario arrotondamento. Per fare questa somma potete trattare il numero di M bit memorizzato nell'array `numfp` come un numero binario positivo a cui dovete sommare il numero binario 1.

Se il codice che avete scritto è corretto, la mantissa a 23 bit (cioè per $M = 23$) del numero 1.2344 dovrà essere uguale a 00111100000000011010010.

► **Seconda parte:** Modificare il programma scritto nella prima parte facendo in modo che se $M = 0$ allora il programma generi un numero casuale `double` compreso nell'intervallo $[1.0, 1.9)$ e lo rappresenti in virgola mobile con $M = 23$. Il programma dovrà poi salvare in un file chiamato `numfpsum.dat` (utilizzando le istruzioni `fopen`, `fprintf` e `fclose`) la somma dei primi N bit del numero in virgola mobile memorizzato nell'array `numfp` a partire da quello meno significativo per ogni $N = 0 \dots M-1$, ovvero il programma dovrà generare un file nel seguente formato:

0	`numfp[0]`
1	`numfp[0]+numfp[1]`
2	.
.	.
.	.
M-1	$\sum_{i=0}^{M-1}$ `numfp[i] = numfp[0] + ...numfp[M-1]`

► **Terza parte:** Creare uno script python chiamato `<nome>_<cognome>.py` che legga il file `numfpsum.dat` e ne grafichi il contenuto (verificate che il grafico sia monotono crescente). Fare in modo che lo script stampi i nomi degli assi, una legenda ed un titolo. Lo script dovrà anche salvare il grafico su di un file chiamato `numfpsum.png`