

16 Ottobre 2023

Programmazione B
Ingegneria e Scienze Informatiche - Cesena
A.A. 2023-2024

Elaborato 5

Data di sottomissione: entro le 23.59 del 22 Ottobre 2023.

Formato di sottomissione: un file compresso con nome `elaborato5.zip` contenente un unico file sorgente con nome `base_conversion.c`

Codeboard: <https://codeboard.io/projects/406829>

Specifiche:

- Sviluppare una funzione ricorsiva ed una iterativa per stampare la conversione di un numero positivo (in base 10) in una base compresa tra 2 e 16.
- I prototipi delle due funzioni sono specificati di seguito (`base_conversion.h`):

```
1  /*
2   * Procedure per stampare un numero n in base b.
3   * La base b deve essere compresa tra 2 e 16,
4   * diversamente le procedure non stampano nulla.
5   */
6
7  // Versione ricorsiva
8  void base_conversion_rc(unsigned int n, unsigned int b);
9
10 // Versione iterativa
11 void base_conversion_it(unsigned int n, unsigned int b);
```

- Le definizioni delle due funzioni devono essere commentate nei punti critici.
- Non è necessario allegare un file principale, contenente la funzione `main()`: verrà valutato unicamente il contenuto del file `base_conversion.c`.
- Per semplificare il codice è possibile evitare di gestire nella versione ricorsiva la conversione del numero zero.
- Non è necessario che la stampa sia terminata da un newline.

- Le implementazioni devono rispettare le seguenti specifiche:
 - La funzione `base_conversion_rc()` deve essere ricorsiva.
 - La funzione `base_conversion_it()` deve essere iterativa.
 - Le funzioni devono gestire solo le basi da 2 a 16. In caso contrario, non stampano nulla.
 - Per le basi maggiori di 10, le cifre da 10 a 15 devono essere rappresentate utilizzando le lettere maiuscole dell'alfabeto secondo la seguente codifica:

$10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F$

Nel dettaglio,

- * la base 11 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A
 - * la base 12 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A,B
 - * la base 13 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A,B,C
 - * la base 14 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A,B,C,D
 - * la base 15 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
 - * la base 16 usa l'alfabeto: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Entrambe le funzioni stampano esattamente la stessa stringa di testo, se invocate con gli stessi parametri. Esempi:
 - * se invocate con parametri $n = 123456$ e $b = 2$, stampano
11110001001000000
 - * se invocate con parametri $n = 123456$ e $b = 3$, stampano
20021100110
 - * se invocate con parametri $n = 123456$ e $b = 8$, stampano
361100
 - * se invocate con parametri $n = 123456$ e $b = 12$, stampano
5B540
 - * se invocate con parametri $n = 123456$ e $b = 16$, stampano
1E240

Vincoli:

- Le implementazioni devono aderire perfettamente ai prototipi e alle specifiche fornite.
- Non è possibile utilizzare tipi di dati avanzati come vettori, strutture, puntatori, ecc.
- Non è possibile utilizzare variabili globali, nemmeno se dichiarate `static`.
- La funzione `base_conversion_rc()` non può richiamare altre funzioni oltre a se stessa e la `printf()`.
- La funzione `base_conversion_it()` può richiamare altre funzioni **non ricorsive** oltre alla `printf()` ed eventualmente funzioni della libreria `math.h`.
- Le eventuali funzioni utility nella libreria devono essere "nascoste" in `base_conversion.c`.
- Le implementazioni devono essere indipendenti dalla dimensione in byte del tipo di dato `unsigned int`.

Suggerimenti:

- Partire da una implementazione del *metodo delle divisioni successive*, che permette di calcolare (e quindi stampare) le cifre della conversione a partire dalla meno significativa.
- Modificare l'implementazione in modo da poter stampare le cifre nell'ordine richiesto: dalla più significativa alla meno significativa.
 - Implementazione ricorsiva: richiede poche righe di codice ed è estremamente efficiente in termini di tempo di calcolo.
 - Implementazione iterativa: molto più complessa da sviluppare ed inefficiente in termini di tempo di calcolo (a causa dei vincoli imposti). Lo sforzo implementativo è notevolmente semplificato se si riescono ad isolare due precisi sotto-problemi e si affrontano in modo indipendente l'uno dall'altro. Difficilmente si può migliorare l'efficienza computazionale.
 - * **Sotto-problema 1:** determinare di quante cifre è composto il numero nella conversione in base b .
 - * **Sotto-problema 2:** stampare una singola cifra (indicizzata) della rappresentazione in base b .

APPENDICE

- Vogliamo convertire un numero intero $n > 0$ dalla rappresentazione in base 10 alla rappresentazione in base b . Di quante cifre sarà composta la rappresentazione in base b di n ?

- Avendo a disposizione k cifre, in base b possiamo rappresentare i numeri positivi

$$0, 1, \dots, b^k - 1$$

- Per poter rappresentare il numero n , dobbiamo trovare un k sufficientemente grande, tale per cui

$$\begin{array}{rclcl} b^k - 1 & \geq & n & \implies \\ \log_b(b^k) & \geq & \log_b(n + 1) & \implies \\ k & \geq & \log_b(n + 1) \end{array}$$

Quindi qualsiasi $k \geq \log_b(n + 1)$ ci permette di rappresentare in base b il numero n .

- Scegliamo il *più piccolo intero* k tra quelli che soddisfano la disequazione: questo è il numero esatto di cifre della rappresentazione di n in base b .

$$k = \lceil \log_b(n + 1) \rceil$$

- La formula per determinare il numero di cifre nella conversione non funziona per il numero 0, che deve essere gestito come caso speciale.
- Come convertiamo un logaritmo da base a a base b ?

$$\log_b(n) = \frac{\log_a(n)}{\log_a(b)}$$