# Laboratorio di Algoritmi e Strutture Dati

Esercitazioni del 16 Ottobre 2012

**Riepilogo di concetti vari del linguaggio C che possono tornare utili**

- Gli operatori logici sono `&&` per l'AND, `||` per l'OR, `!` per il NOT.
- `&&` e `||` hanno la *valutazione cortocircuitata*: Si garantisce che la valutazione del loro primo operando viene eseguita per prima. La valutazione del secondo non è eseguita affatto se la valutazione del primo è sufficiente a determinare il valore totale dell'espressione.
- L'operatore `%` fornisce il resto della divisione intera: esempio: `11 % 2 == 1`.
- Le definizioni di macro (anche quelle con parametri) non devono essere terminate da `;`
- In C non esiste un tipo `boolean`. Per memorizzare valori booleani si utilizzi il tipo `char`.

## Esercizio 4: dec2bin, dec2base

Si supponga di aver dichiarato nel `main` l'array:

```
char b[MAXDIM + 1];
```

dove `MAXDIM` è una macro definita come

```
#define MAXDIM 80
```

**dec2bin:**

Si tratta di implementare due funzioni:

- `short dec2bin(int d, char *b)`

  Converte l'intero `d` in base 10 in una stringa di caratteri $\beta_{\texttt{d}}$ che lo rappresenta in forma binaria. La stringa dove memorizzare il risultato è passata per mezzo del parametro `char *b`.

  Il valore di ritorno di tipo `short` può essere utile nel caso in cui il carattere iniziale della stringa $\beta_{\texttt{d}}$ non sia necessariamente il primo (lo zeresimo, in realtà) dell'array `b`.

  Esempio: se `d == 11` allora $\beta_{\texttt{d}}$ deve essere la stringa di caratteri `1011`, e può essere memorizzata dove più torna comodo nell'array `b`. Supponendo che il primo carattere di $\beta_{\texttt{d}}$ sia memorizzato nel 76-esimo di `b`, allora `dec2bin` dovrà restituire 76.

  N.B. Non dimenticate di gestire il carattere di terminazione nella rappresentazione delle stringhe!

- `int bin2dec(char *b)`

  Converte la stringa binaria $\beta$ il cui indirizzo (l'indirizzo del suo primo carattere) è `b` nell'intero $\texttt{d}_\beta$ in base 10 corrispondente. Restituisce $\texttt{d}_\beta$.

Esempio: se l'array `b` dichiarato nel `main` memorizza la stringa binaria `1011` in modo che l'indirizzo (del primo carattere) di `1011` sia l'indirizzo del 76-esimo carattere di `b`, allora `bin2dec` dovrà essere richiamata passandole l'indirizzo del 76-esimo carattere di `b`. Tale chiamata restituirà il valore intero 11.

Ovviamente, se si sceglie di rappresentare sempre la stringa $\beta_d$ a partire dall'indirizzo iniziale dell'array `b`, allora `dec2bin` restituirà sempre `0`, mentre l'indirizzo da passare a `bin2dec` sarà sempre `b`.

Non si devono usare variabili *globali*.

Il `main` dovrà implementare l'algoritmo seguente:

1. Stampa del messaggio: `Immetti numero in base 10 da convertire in base 2:`
2. Lettura dell'intero `d` immesso dall'utente. (Non si richiede alcun controllo sulla correttezza dell'immissione).
3. Conversione dell'intero `d` dalla notazione decimale alla notazione binaria nella stringa $\beta_{\texttt{d}}$ (rappresentata nell'array `b` dichiarato nel `main`).
4. Stampa del messaggio: `d in base 10 si scrive` $\beta_{\texttt{d}}$ `in base 2`
5. Conversione della stringa binaria $\beta_{\texttt{d}}$ nell'intero `d` in base 10 corrispondente.
6. Stampa del messaggio: $\beta_{\texttt{d}}$ `in base 2 equivale a d in base 10`

**dec2base:**

Si tratta di generalizzare le due funzioni precedenti in modo che ricevano un argomento supplementare `short base` che specifica la `base` in cui e da cui convertire gli interi in base 10.

- `short dec2base(short base, int d, char *b)`

  Converte `d` nella sua espressione in base `base`. Memorizza la stringa ottenuta nell'array di caratteri il cui indirizzo iniziale è `b`, con le stesse modalità della funzione `dec2bin`.

- `int base2dec(short base, char *b)`

  Converte la stringa di caratteri `char *b` che contiene la rappresentazione di un intero in base `base`, nella sua rappresentazione `d` come intero in base 10. Restituisce `d`.

- La funzione `main` dovrà essere modificata in questo modo. Prima del passo 1, si stampi il messaggio: `Immetti base in cui convertire:`. Si legga da tastiera il valore da assegnare alla `base`. Si modifichino inoltre i rimanenti messaggi in modo che ogni menzione della base `2` venga rimpiazzata dal valore della `base` scelta.

  Si ricorda che per leggere con `scanf` un valore da assegnare a una variabile di tipo `short`, bisogna utilizzare la specifica di formato: `%hd`.

Ricordiamo che la stringa $\beta_d$ rappresenta l'intero $d$ in base $k$ se solo se vale che:

1. $\beta_d = p_1 p_2 p_3 \cdots p_n$ per qualche $n$. I simboli $p_1, p_2, \ldots, p_n$ appartengono a un alfabeto $\{c_1, c_2, \ldots, c_k\}$ i cui $k$ elementi distinti sono dette *cifre*. Il valore $v(c_i)$ è $i - 1$.
2. $d = v(p_1) * k^{n-1} + v(p_2) * k^{n-2} + \cdots + v(p_{n-1}) * k + v(p_n)$.

Esempio, se la base è 2, allora $11 = 1 * 2^3 + 0 * 2^2 + 1 * 2 + 1$, dunque, scegliendo come alfabeto $\{c_1 = \texttt{0}, c_2 = \texttt{1}\}$, abbiamo che $v(\texttt{0}) = 0$ e $v(\texttt{1}) = 1$, inoltre $p_1 = \texttt{1}, p_2 = \texttt{0}, p_3 = \texttt{1}, p_4 = \texttt{1}$, e dunque l'espressione di 11 in binario risulta essere `1011`.

Si esegua l'esercizio dapprima limitando la scelta della base $k$ a $1 < k \leq 10$, in tal modo l'alfabeto delle cifre necessarie a codificare in base $k$ è $\{0, 1, \ldots, k-1\}$.

In un secondo tempo si estenda la scelta della base $k$ a $1 < k \leq 36$, e si utilizzino le lettere minuscole `a`,`b`,...,`z` per ottenere un alfabeto di cifre adeguato.

Ad esempio, se la base è 16, allora l'alfabeto è l'insieme di cifre $\{0, 1, \ldots, 9, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\}$, e quindi $v(\texttt{a}) = 10$, $v(\texttt{b}) = 11$, ..., $v(\texttt{f}) = 15$. Allora $255 = 15 * 16^1 + 15$ e la rappresentazione in base 16 di 255 è `ff`.