Università di Roma – Consorzio NETTUNO – Corso di Laurea in Ingegneria
Esame di Fondamenti di Informatica 1 – Prof. Andrea Sterbini – 13-5-2006

Esercizio 1

Si supponga di dover calcolare e stampare la media scorrevole degli ultimi ***k*** valori dispari letti da input.

Si scriva la funzione C **mediascorrevole**, che riceve come argomento il valore ***k*** (non maggiore di 31) e che legge una successione di valori interi positivi da input fino a quando non viene inserito il valore 0 (zero) che indica che non ce ne sono altri. Per ogni valore dispari letto in input (dopo i primi ***k***) si calcoli la media degli ultimi ***k*** valori e la si stampi con una cifra dopo la virgola.

Esempio: se k=4 e i numeri inseriti sono: **1** 4 6 **7 31** 22 2 **11 9 43** 2 4 8 4 **1** 0

l'output sarà: 12,5 14,5 23,5 16

Esercizio 2

Si supponga di dover realizzare uno spelling-checker, ovvero di dover controllare tutte le parole di un testo per verificare se sono in un dizionario. Si scriva la funzione C **correttoreOrtografico** che riceve come argomenti:

- il testo da controllare (una stringa)
- il dizionario, un vettore di stringhe che contiene le parole conosciute, disposte in ordine alfabetico
- un intero che indica quante parole sono contenute nel dizionario

Le parole del testo da controllare sono solo quelle formate da almeno 3 caratteri e non più di 15 caratteri alfabetici (si può usare la funzione **isalfa**). Si considerino tutti i caratteri non alfabetici come separatori delle parole. Si supponga di avere a disposizione una funzione **cercaparola** (vedi esercizio 3) che permette di cercare nel dizionario se la parola è presente o assente e che torna vero (1) o falso (0).

Si stampino tutte le parole del testo (nell'ordine) non trovate nel dizionario.

Esercizio 3

Si supponga di dover cercare in un dizionario se una parola è presente o assente. Il dizionario è composto da un vettore di stringhe, ciascuna contenente una parola, in ordine alfabetico.

Si scriva la funzione C **cercaparola** che riceve come argomenti:

- la parola da cercare (una stringa)
- il dizionario in cui cercare (un vettore di stringhe in ordine alfabetico)
- il numero di elementi del dizionario

E che torna 1 (vero) se la parola è presente, altrimenti torna 0 (falso).

Si implementi la funzione iterativamente (senza usare ricorsione) usando la strategia della ricerca binaria, ovvero si confronti la parola da cercare con quella centrale e si continui a cercare nella metà sinistra o destra a seconda del risultato dimezzando ogni volta la zona in cui cercare e così via. Per confrontare due stringhe si può usare la funzione **strcmp**, che torna -1, 0 o +1 a seconda che la prima stringa venga prima, sia uguale o venga dopo dell'altra.

Esercizio 4

Si implementi la funzione precedente come funzione ricorsiva, ovvero:

- se il dizionario è vuoto la parola non è stata trovata
- altrimenti si confronti la parola con l'elemento centrale del dizionario e, a seconda del risultato
  - la parola è stata trovata se è uguale
  - si cerchi nella metà sinistra se la parola cercata è minore di quella centrale
  - altrimenti si cerchi nella metà destra

Svolgimento

Nota: non ho provato a compilare ed eseguire il codice qui sotto, fatemi sapere se non funziona qualcosa

Esercizio 1

Per calcolare la media scorrevole su k elementi bisogna ricordare gli ultimi k elementi, usiamo quindi un vettore sufficientemente capiente. Dato che k è sempre minore di 32 il vettore avrà 31 elementi.
Usiamo il vettore come una coda circolare, ovvero memorizziamo gli ultimi k numeri dispari letti nei primi k elementi del vettore, e per ogni nuovo elemento da aggiungere sovrascriviamo quello letto k+1 volte prima. La coda circolare si implementa calcolando l'indice ***i*** in cui memorizzare l'elemento ***n***-esimo con la formula

$$i = n \bmod k$$

```
void mediascorrevole(int k) {
     int coda[31], n = 0, valore, i;
     scanf("%d", &valore);                    // leggo il primo valore
     while (valore) {                         // continuo se è diverso da 0
          if (valore % 2) {                   // se il valore è dispari
               coda[ n % k ] = valore;        // lo inserisco nella coda
               n++;                           // ed aumento n
               if (n>=k) {                    // se ne ho letto k o più
                    for (i=0 ; i<k ; i++)              // li sommo
                         somma += valore[i];
                    printf("%.1f", (float)somma/k);    // e stampo la media
               }
          }
          scanf("%d", &valore);               // leggo il prossimo
     };
```

Esercizio 2

Nel leggere i caratteri dal testo bisogna stare attenti ad ignorare le parole più lunghe di 15 caratteri o minori di 3. Uso un vettore di appoggio per copiarci i caratteri e cerco la parola non appena scopro che è finita (carattere non alfabetico o '\0' finale).

```
int correttoreOrtografico( char * testo, char * dizionario[], int dim) {
     char parola[16];                         // buffer per la parola
     int i, lun;
     for (i=0, lun=0 ; testo[i] ; i++) {      // scandisco tutto il testo
          // se il carattere è alfabetico e ancora non siamo al 15°
          if (isalpha(testo[i] && lun < 15) {
               parola[lun++] = testo[i];     // lo aggiungo alla parola
          } else {                       // altrimenti la parola è finita
               parola[lun] = '\0';           // termino la parola con '\0'
               // cerco la parola solo se non l'ho ancora mai stampata
               if (lun>3 && !cercaparola(parola, dizionario, dim))
                    printf("%s\n", parola);  // se non c'e' la stampo
               lun = 0;                      // azzero la parola
          }
     }
     // se c'e' un'ultima parola la controllo
     if (lun>3 && !cercaparola(parola, dizionario, dim))
          printf("%s\n", parola);  // se non c'e' la stampo
}
```

Esercizio 3

La versione iterativa della ricerca binaria ricorda quale pezzo del vettore va analizzata usando due indici: l'estremo sinistro e l'estremo destro, quando non ci sono elementi vuol dire che la parola non è stata trovata.

```
int cercaparola(char * parola, char * dizionario, int dim) {
     int sx = 0;                    // all'inizio sx è al primo elemento
     int dx = dim-1;                // e dx all'ultimo
     while (sx <= dx) {             // ripeto finchè ci sono elementi
          int medio = (sx+dx)/2;    // calcolo la pos. dell'elemento medio
          // confronto la parola con l'elemento medio
          int confronto = strcmp(parola, dizionario[medio]);
          if (confronto==0)
               return TRUE;         // se sono uguali torno vero
          if (confronto<0) {        // se la parola è a sinistra
               dx = medio - 1;      // mi limito alla parte a sx del medio
          else                      // altrimenti è a destra
               sx = medio + 1;      // mi limito alla parte a dx del medio
     }
     // se sono qui vuol dire che non l'ho trovato
     return FALSE;
}
```

Esercizio 4

La versione ricorsiva della ricerca usa due parametri per indicare l'estremo sinistro e quello destro del pezzo di vettore in cui cercare: ovvero il puntatore al primo elemento del pezzo di vettore ed il numero di elementi del vettore.

Per determinare se il vettore è vuoto basta esaminare il numero di elementi

```
int cercaparolaR(char * parola, char * dizionario, int dim) {
     if (dim == 0) return FALSE;   // se il vettore è vuoto torno falso
     int medio = dim/2;            // calcolo la pos dell'elemento medio
     // lo confronto con la parola
     int confronto = strcmp(parola, dizionario[medio]);
     if (confronto==0)
          return TRUE;             // se sono uguali torno vero
     if (confronto < 0)            // se la parola è minore
          // cerco nella prima metà (ignoro il medio che ho già visto)
          return cercaparolaR(parola, dizionario, medio-1);
     else
          // sennò nella seconda metà, inizia con l'el. succ. al medio
          return cercaparolaR(parola, dizionario+medio+1, dim-medio);
}
```