

Fondamenti di Informatica L-A (A.A. 2008/2009) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d’Esame di Giovedì 2 Aprile 2009 – durata 2h
Compito A

Compito Modalità Fondamenti di Informatica L-A (solo a.a. precedenti)

ESERCIZIO 1 (10 punti)

Un gestore di siti web, al fine di costruire un rudimentale motore di ricerca, è interessato a scoprire quali sono le parole più frequenti utilizzate nelle pagine web pubblicate sul suo sito. A tal scopo necessita di un programma che, dato in ingresso un file di testo contenente il testo pubblicato sul web (opportunamente filtrato dai vari codici html), ed un file di testo contenente parole non interessanti (e quindi da non considerare), restituisca in uscita un elenco di parole assieme al numero di volte che ogni parola compare in tale testo iniziale. Il gestore ha anche individuato la necessità di definire una struttura dati **termine**, che contenga spazio per una stringa ben formata (per semplicità, al più 127 caratteri), ed un intero rappresentante il numero di volte che tale parola è stata incontrata.

Dopo aver definito opportunamente la struttura dati **termine**, il candidato realizzi la seguente funzione:

```
termine * conta(FILE * text, FILE * stopWord, int * dim);
```

La funzione riceve in ingresso un puntatore ad un file di testo (variabile **text**), contenente parole separate da spazi, senza alcun carattere di punteggiatura; come secondo parametro viene passato un puntatore ad un file di testo contenente le parole da ignorare. La funzione deve restituire un array di strutture dati di tipo **termine**, contenente le parole interessanti (cioè non escluse dal file **stopWord**) assieme al numero di volte in cui compaiono. Tale array deve essere allocato dinamicamente (ogni file di testo contiene al massimo 100 parole), e la sua dimensione deve essere restituita tramite il parametro **dim**. Si rammenta al candidato l’esistenza delle funzioni di libreria **rewind(...)** per ricominciare a leggere un file dall’inizio, **strcmp(...)** per confrontare due stringhe e **strcpy(...)** per copiare una stringa in un’altra.

Il candidato realizzi quindi un programma **main** di esempio che chieda all’utente i nomi di due file (il file contenente il testo ed il file di parole da scartare), e stampi a video l’elenco delle parole incontrate (insieme al numero di volte che ogni parola compare): per ottenere tale risultato, il **main** dovrà fare uso della funzione **conta(...)** di cui sopra.

ESERCIZIO 2 (8 punti)

Si scriva una funzione ricorsiva

```
int verifica(list t);
```

che riceva in ingresso il puntatore **t** alla radice di una lista contenente stringhe di caratteri maiuscoli. In uscita la funzione restituisca il numero di vocali contenute nell’intera lista. La funzione **verifica(...)** utilizzi a sua volta una funzione iterativa:

```
int contavo(char * st);
```

che conti il numero di vocali maiuscole in una stringa. Si scriva il codice delle due funzioni nonché le strutture dati che utilizzano. Non viene data a disposizione nessuna libreria predefinita sulle liste.

ESERCIZIO 3 (5 punti)

Dati i numeri decimali $A=115$, $B=42$ e $C=-87$, si determini il minimo numero di bit necessari per rappresentare contemporaneamente i suddetti numeri in binario. Utilizzando poi lo stesso numero di bit, si esegua l'operazione:

$$D = B - C$$

e si discuta se il risultato ottenuto è o no significativo.

ESERCIZIO 4 (3 punti)

Si spieghino in modo chiaro e conciso le operazioni svolte da un compilatore e si sottolinei la differenza rispetto ad un approccio interpretato.

ESERCIZIO 5 (6 punti)

Si consideri il seguente codice:

```
#include <stdio.h>
#include <stdlib.h>

int dim = 4;

int dividi(char v[], int a, int b){
    int i=0;
    char * temp;
    for (i=0; *(v+i) != '\0'; i++);
    temp = (char*) malloc(sizeof(char) * (i+1));
    i=0;
    do {
        temp[i] = v[i];
    } while ((i=i+1) && v[i-1] != '\0');
    for (i=0; temp[i] != '\0'; i++)
        if ((i%a)%b == 0) (*v)++;
    return v[0] - temp[0];
}

int main(){
    char msg[] = "abcdefgh";
    int dim=0;
    dim = dividi(msg, 3, 2);
    printf("%s %d", msg, dim);
    return (0);
}}
```

Qualora il compilatore compili questo programma, quali sono i valori stampati a video a tempo di esecuzione? (si motivi la risposta in maniera opportuna)

Soluzioni

ESERCIZIO 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
    char parola[128];
    int count;
} termine;

termine * conta(FILE * text, FILE * stopWord, int * dim) {
    int pos, trovato;
    char tempWord[128];
    char stop[128];
    termine * result;

    result = (termine *) malloc(sizeof(termine) * 100);
    *dim = 0;
    while (fscanf(text, "%s", tempWord)==1) {
        trovato = 0;
        rewind(stopWord);
        while (fscanf(stopWord, "%s", stop)==1 && !trovato) {
            if (strcmp(stop, tempWord) == 0)
                trovato = 1;
        }
        if (!trovato) {
            for (pos=0; pos < *dim && strcmp(tempWord, result[pos].parola); pos++);
            if (pos == *dim) {
                strcpy(result[pos].parola, tempWord);
                result[pos].count = 0;
                (*dim)++;
            }
            result[pos].count = result[pos].count + 1;
        }
    }
    return result;
}

int main(void) {
    char f1[100];
    char f2[100];
    FILE * fp1;
    FILE * fp2;
    int dim;
    termine * result;
    int i;

    printf("Inserire i nomi dei due file: ");
    scanf("%s %s", f1, f2);
    fp1 = fopen(f1, "r");
    fp2 = fopen(f2, "r");
    result = conta(fp1, fp2, &dim);
    for (i=0; i<dim; i++)
        printf("%s %d\n", result[i].parola, result[i].count);
    fclose(fp1);
    fclose(fp2);
    return (0);
}
```

ESERCIZIO 2

```
int contavo(char * st) {
    int result = 0;
    while (*st != '\0') {
        if (*st=='A' || *st=='E' || *st=='I' || *st=='O' || *st=='U')
            result++;
    }
    return result;
}
```

```

int verifica(list t) {
    if (empty(t)) {
        return 0;
    }
    else {
        return contavo(head(t)) + verifica(tail(t));
    }
}

```

Per le primitive sulle liste, si faccia riferimento direttamente alle slides del corso.

ESERCIZIO 3

Il numero maggiore da rappresentare è A, che vale 115. Sono necessari almeno 115 simboli differenti per rappresentare tutti i numeri da 0 a 115 almeno... la più piccola potenza di 2, maggiore di 115, è 128, cioè 2^7 . Sembra quindi che possano bastare 7 bit. Però bisogna considerare anche il numero C, che vale -87. Siccome è necessario rappresentare anche tale valore, 7 bit forse non sono più sufficienti. Dunque è necessario rappresentare tutti i numeri compresi tra -87 e 115, cioè 87 numeri negativi, più lo 0, più 115 numeri positivi per un totale di 203 numeri. Saranno quindi necessari almeno 203 simboli differenti. La più piccola potenza di 2, maggiore di 203 è 256, cioè 2^8 . Per rappresentare i numeri A, B e C sono necessari almeno 8 bit.

Utilizzando la notazione complemento a due:

B 42: 00101010

C -87:

```

      87:  01010111
           10101000
                1
           -----

```

C -87 10101001

```

      B-C:           1      1
                  00101010 -
                  10101001
                  -----
                  10000001   (-127 in decimale)

```

Si noti che nelle due colonne più significative, è stato fatto un solo riporto/prestito: ciò comporta che si sia verificato un errore di overflow nella rappresentazione del risultato. Infatti si è ottenuto il valore -127, mentre svolgendo il conto a mente ci saremmo aspettati il valore 129. Del resto ci si poteva rendere conto, ancor prima di eseguire l'operazione in binario, che il risultato non sarebbe stato significativo poiché con 8 bit e la notazione a complemento a due si rappresentano i numeri da -128 a +127. Ed il risultato atteso, 129, non può quindi essere rappresentato con tale notazione e soli 8 bit.

ESERCIZIO 5

Il programma è corretto, compila correttamente e stampa:

fbcddefgh 5

Il main non fa altro che invocare la funzione `dividi(...)`, e salvare il risultato restituito nella variabile `dim`. Quindi il main stampa la stringa `msg` ed il valore di `dim`.

La funzione `dividi(...)` per prima cosa alloca memoria dinamica a sufficienza e vi copia la stringa passata come parametro di ingresso. Quindi scorre tale stringa copiata tramite la variabile `i` usata come indice. Tutte le volte che per il valore di `i` vale una certa condizione, viene incrementato il primo carattere dell'array passato per riferimento. La condizione è che $(i \% 3) \% 2$ sia pari a 0. Ciò vale per i valori di `i` pari a 0, 2, 3, 4, 6. Alla fine la funzione restituisce il numero di incrementi effettuati, cioè 5.