Numero di vocali distinte

Gruppo N. 2

Daniele Orlando 180583

Analisi del problema

La realizzazione del codice relativo alle specifiche dettate per il progetto non ha incontrato difficoltà complessive particolari.

Attenzioni di un certo rilievo sono state concentrate sull'analisi dei file passati come argomenti avendo le specifiche progettuali dettato una discreta libertà per la loro formattazione.
La maggior parte del codice realizzato infatti è dedicato proprio al controllo di correttezza dell'input.

Una volta validata la formattazione del file in ingresso si è provveduto ad allocare dinamicamente spazio in memoria sufficiente per contenere tutti record presenti.
Questa operazione è stata ripetuta due volte in maniera uguale per entrambi i file presentando questi ultimi struttura esattamente identica. Intuibilmente, sono state create funzioni generiche evitando qualsiasi ripetizione di codice.

Ottenuti i rispettivi vettori, si è proceduto con il controllo sulla loro composizione come prescritto dalle linee guida.
Unica nota rilevante nel processo di confronto fra i vettori risiede nell'implementazione di una cache che ha lo scopo di accellerare l'esecuzione dell'algoritmo.

Struttura

Inclusioni (#*include*)*:*

- *stdio.h*: necessario per le operazioni di input output su stream (inclusi *stdout* e *stderr*);
- *stdlib.h*: necessario per l'uso delle funzioni *calloc()* ed *exit()* e della costante *EXIT_FAILURE*;
- *string.h*: la sua inclusione è necessaria per l'uso della funzione *strncmp()* in *checkArgument()*. *strncmp* viene adottata per confrontare l'header del file in ingresso che deve essere uguale a '*Prj2_08*'.

Definizioni (#*define*):

- *true*: è settato ad '1' ed emula il valore booleano true;
- *false*: è settato a '0' ed emula il valore booleano false.

Definizioni di tipo (*typedef*):

- *bool*: emula il tipo booleano assente in C89.

Prototipi:

- *void raiseError(char *, char *)*: è una interfaccia per il sollevamento di un errore critico, con conseguente terminazione del programma;
- *char ** checkArgument(char *, int *)*: è la funzione per il controllo della struttura del file in input e la relativa estrapolazione dei record;
- *int countVowels(char *)*: è una funzione per la conta delle vocali distinte presenti in una stringa;
- *bool isLessThan(int, char *, int *, int)*: verifica se il numero di vocali distinte è minore delle vocali distinte presenti nella stringa. Se il numero di vocali della stringa non è prensente nella cache viene calcolato ed inseritovi.

Funzioni

void raiseError(char *, char *)

Prende come parametri la stringa da visualizzare in output sullo stderr ed un parametro di tipo stringa da inserire nel messaggio.

Non ritorna alcun valore.

A seguito della sua invocazione il programma visualizza sullo stderr un messaggio di errore e termina la sua esecuzione con uno stato di uscita di tipo EXIT_FAILURE.

char ** checkArgument(char * arg, int * rowsNumber)

Prende come parametri il percorso al file da processare ed un puntatore ad intero in cui viene memorizzato il numero di record trovati.

Ritorna il puntatore al vettore contenente i record.

Apre uno stream sul file passato in ingresso, ne legge i record, salva il loro numero in *rowsNumber*, alloca memoria dinamicamente per il loro contenimento e li memorizza nel vettore.
Interrompe l'esecuzione del programma chiamando *raiseError()* nel caso in cui:

- il file non sia leggibile;
- il file non inizi con l'intestazione giusta;
- l'intestazione non è seguita da una formattazione secondo specifiche;
- il numero che indica le righe presenti nel file è minore di uno;
- trovi caratteri non ammessi a seguito del numero;
- rilevi spazi all'interno della stringa;
- rilevi linee vuote;
- rilevi un numero di righe differente da quanto indicato all'inizio del file.

int countVowels(char *)

Prende come paramentro la stringa da analizzare.

Ritorna il numero di vocali distinte trovato.

bool isLessThan(int, char *, int *, int)

Prende come parametri il numero di vocali da controllare, la stringa di cui calcolare le vocali, un vettore per la cache, l'indice a cui accedere nella cache.

Ritorna true se il numero di vocali distinte passato in ingresso è minore del numero di vocali distinte presenti nella stringa passata in ingresso, altrimenti false.

Per evitare di ricalcolare ripetutamente il numero di vocali distinte presenti nella stringa passata in ingresso, la funzione si appoggia ad un vettore per la cache e ad un indice associato alla stringa.
Qualora nella cache, nella posizione indicata dall'indice, sia presente un valore diverso da zero viene saltato il calcolo delle vocali distinte sulla stringa e viene usato il suddetto valore, altrimenti viene calcolato e viene inserito nella cache nella posizione indicata dall'indice.

Main

Il main richiede che vengano passati come paramentri almeno due percorsi ai file da analizzare.
Qualora i parametri siano in numero minore di due visualizza a schermo un messaggio di aiuto e termina l'esecuzione del programma con stato uguale a meno uno.
Qualora invece siano in numero maggiore, i parametri in eccesso vengono ignorati.

Una volta verificato che i parametri siano nel numero corretto chiama *checkArgument()* prima sul primo parametro e poi sul secondo.
Ottenuti i vettori di stringhe esegue due cicli iterativi innestati per eseguire i confronti dettati dalle linee guida. Appena una di queste condizione viene meno, i cicli terminano evitando di iterare inutilmente.

Lo stato finale viene visualizzato sullo stdout e ritornato come stato di uscita.

Compilazione

L'unico file richiesto per la compilazione è 'src/Vocali.c'.

Il codice rispetta tutte le direttive di compatibilità con lo standard C90 (ANSI C89) verificate mediante il compilatore gcc 4.3.0 su sistema RedHat Linux.

Le direttive di compilazione usate sono le seguenti:
gcc -Wall -Wextra -pedantic -std=c89 -o bin/Vocali src/Vocali.c

Le direttive di esecuzione sono le seguenti:
./bin/Vocali res/arg1.txt res/arg2.txt