Fondamenti di Informatica
Assegnamento 6

Istruzioni per lo svolgimento e la consegna

- La prima operazione da effettuare è modificare il file `studente.txt` (presente nella directory dove avete trovato questo `pdf`) inserendo il proprio nome e cognome e numero di matricola. Utilizzare un semplice editor di testo, salvando il file senza modificarne il nome.
- Nella stessa directory sono presenti i file necessari allo svolgimento dell'esercizio. **Per ogni esercizio dovrà essere modificato solamente il file `.c` corrispondente. Non devono essere modificati né spostati o eliminati i rimanenti file, pena la valutazione negativa dell'assegnamento**. Nel file `.c` dovranno essere mantenuti tutti e soli gli output a schermo forniti, in modo da mantenere la corrispondenza con l'output di esempio. Inoltre, è possibile definire funzioni ausiliarie all'interno del sorgente `.c`.
- Per compilare e generare l'eseguibile, da terminale entrate nella directory dove avete trovato questo `pdf` e lanciate il comando `make nome_esercizio`. Il `nome_esercizio` corrisponde al nome del sorgente, privato dell'estensione `.c`. Verrà generato l'eseguibile `nome_esercizio` che, lanciato da terminale (`./nome_esercizio`), vi permetterà di provare il vostro programma.
- Lanciando invece il comando (`./self_evaluation nome_esercizio`) eseguirete in maniera automatica alcuni test per verificare le soluzioni che avete implementato. I test sono studiati per verificare anche i casi particolari, in modo da gestire quelli che possono essere errori comuni in fase di implementazione. **Tenete presente che il correttore funziona solo all'interno di una distribuzione Linux a 64 bit (ad esempio, le macchine messe a disposizione nel laboratorio)**.
- La procedura di consegna dovrà iniziare lanciando il programma `./prepara_consegna.sh` presente nella directory dove avete trovato il presente `pdf`. Una volta lanciato, esso genererà un archivio di nome `consegna.tar.gz`: tale file sarà **l'unico che dovrà essere inviato** attraverso il sito `https://stem.elearning.unipd.it` per consegnare il vostro elaborato, seguendo anche le istruzioni che saranno fornite in aula dai docenti.

Considerate 2 aspetti:

- Se ci sono errori in compilazione/esecuzione, c'è qualcosa che rende errata/incompleta la vostra implementazione;
- Se non ci sono errori in compilazione/esecuzione, verificate che i risultati siano corretti (in alcuni casi è molto semplice fare il calcolo anche a mente).

1 Parole chiave (`keywords.c`)

Scrivere un programma C che estragga delle parole chiave da un testo formattato, allocando dinamicamente la memoria necessaria.
Il testo formattato è contenuto in un file di testo di cui si conosce la formattazione, e di cui si può dare per scontata la correttezza.
Il file è strutturato come segue (dal `#` in poi sono commenti e non saranno presenti nei file):

```
5   # Numero di righe e di parole chiave (una per riga) presenti nel file (questa riga non viene contata)
3 word_0 word_1 word_2 keyword_0 ... word_n0   # il numero all'inizio di ogni riga indica la posizione della keyword nella riga
2 word_0 word_1 keyword_1 word_3 ... word_n1   # il numero di parole nella riga varia di caso in caso (max 256 caratteri)
5 word_0 word_1 word_2 word_3 word_4 keyword_2 # ci sono sempre parole sufficienti per raggiungere la posizione della keyword
0 keyword_3 word_1 word_2 word_3 ... word_n3   # gli indici partono da 0, le parole sono separate esclusivamente da spazi (1 o +)
1 word_0 keyword_4 word_2 word_3 ... word_n4   # la lunghezza delle singole parole varia anch'essa di caso in caso
```

Implementare la seguente funzione che riceve in input il path del file di testo e restituisce un puntatore all'array di parole chiave e il loro numero:

```
// Aprire in lettura il file di cui si riceve il path in input
// Analizzare il file per estrarre le parole chiave dal testo
// Restituire le parole usando un array di stringhe allocato dinamicamente
// Usare num_rows_ptr per fornire in output il numero di parole chiave trovate
// Nel caso la lettura del file o l'allocazione di memoria non vadano a buon fine,
// azzerare il valore di num_rows_ptr, liberare la memoria allocata e restituire NULL
string* get_keyworks(char filepath[], int *num_rows_ptr);
```

Il file header `keywords.h` contiene le seguenti informazioni:

```
// Numero massimo di caratteri per riga (compresa la posizione della parola chiave)
#define MAX_CHARS_PER_ROW 256

// Definizione del tipo string
typedef char* string;
```

In base al metodo scelto per risolvere il problema potreste trovare utile sapere che è possibile:

```
// Andare alla riga successiva di un file ignorandone il contenuto
FILE* f;
...
fscanf(f, "%*[^\n]\n");

// Convertire una stringa in un long tramite la funzione di libreria:
long strtol(const char *restrict str, char **restrict str_end, int base);
// Per informazioni sull'utilizzo andare alla pagina:
// https://en.cppreference.com/w/c/string/byte/strtol
// Esempio
char p[] = "10";
char *end;
long i = strtol(p, &end, 10);
if (p == end) {
    printf("Non sono stati trovati numeri da convertire\n");
} else {
    printf("Numero convertito: %ld\n", i);
}

// Separare un array di caratteri in base a dei caratteri di delimitazione
// tramite la funzione di libreria:
char *strtok(char *restrict str, const char *restrict delim);
// Per informazioni sull'utilizzo andare alla pagina:
// https://en.cppreference.com/w/c/string/byte/strtok
// Esempio
char input[] = "A bird came down the walk";
char *token = strtok(input, " ");
while(token) {
    printf("%s\n", token);
    token = strtok(NULL, " ");
}
```

Il programma di test è già implementato e compilato, fornito in forma di file oggetto `keywords_main.obj`.
Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o keywords keywords.c keywords_main.obj
```

Esempio 1

File formattato:

```
5
3 word_0 word_1 word_2 keyword_0 ... word_n0
2 word_0 word_1 keyword_1 word_3 ... word_n1
5 word_0 word_1 word_2 word_3 word_4 keyword_2
0 keyword_3 word_1 word_2 word_3 ... word_n3
1 word_0 keyword_4 word_2 word_3 ... word_n4
```

Output:

```
File path:
keywords_files/words.txt
KEYWORDS:
0. keyword_0
1. keyword_1
2. keyword_2
3. keyword_3
4. keyword_4
```

Esempio 2

File formattato:

```
9
0 The morning had dawned clear and cold, with a
     crispness that hinted at the end of summer.
10 They set forth at daybreak to see a man
     beheaded, twenty in all, and Bran rode
     among them, nervous with excitement.
19 This was the first time he had been deemed
     old enough to go with his lord father and
     his brothers to see the king's justice done.
5 It was the ninth year of summer, and the
     seventh of Bran's life.
10 The man had been taken outside a small
     holdfast in the hills.
7 Robb thought he was a wildling, his sword
     sworn to Mance Rayder, the
     King-beyond-the-Wall.
10 It made Bran's skin prickle to think of it.
     He remembered the hearth tales Old Nan told
     them.
3 The wildlings were cruel men, she said,
     slavers and slayers and thieves.
3 They consorted with giants and ghouls, stole
     girl children in the dead of night, and
     drank blood from polished horns.
```

Output:

```
File path:
keywords_files/a_game_of_thrones.txt
KEYWORDS:
0. The
1. twenty
2. brothers
3. of
4. the
5. sword
6. remembered
7. cruel
8. giants
```