TECNICHE DI PROGRAMMAZIONE

Esercitazione di Laboratorio 4

Obiettivi

- Risolvere problemi gestendo problemi iterativi su dati scalari (*Dal problema al programma: Cap. 4*), di codifica (4.2), di elaborazione testi (4.3) e di verifica (4.4)

Contenuti tecnici

- Basi di Input Output
- Utilizzo di funzioni
- Costrutti condizionali e iterativi
- Manipolazioni elementari di numeri (int e float) e caratteri (char)

Da risolvere durante il laboratorio oppure prima/dopo il laboratorio stesso

Esercizio 1.

Competenze: IO su file, manipolazioni di caratteri;
Categoria: problemi di codifica applicati a testi/caratteri (4.2.2)

Un file (`sorgente.txt`) contiene un testo composto da un numero indefinito di righe.
Notare che il testo NON contiene il carattere '$'.

Lo scopo del programma è di ricodificare il testo sostituendo sequenze di caratteri ripetuti da 2 a 9 volte (ATTENZIONE: il numero non comprende il primo carattere, cioè AA contiene una ripetizione, BBB due ripetizioni, ecc.) con la terna di caratteri <carattere ripetuto>$<numero di ripetizioni> (ATTENZIONE: per il <numero di ripetizioni> è sufficiente un carattere). Se un carattere fosse ripetuto più di 9 volte, le ripetizioni sarebbero spezzate in più intervalli.

Ad esempio, la sequenza di caratteri "il numero 1000000 e' grande", viene compressa in "il numero 10$5 è grande". La sequenza, "ci sono 15 = ripetuti: ========== e 4 punti…." viene ricodificata in: "ci sono 15 = ripetuti: =$9=$4 e 4 punti.$3"

Il risultato della ricodifica sia salvato su un secondo file (`compresso.txt`).

Esempio:
Il contenuto del file `sorgente.txt` è:

```
 Partenza      Destinazione    Costo
   Parigi          New York     1000
     Roma            Londra      700
   Sidney       Los Angeles     2222
```

Il file di uscita `ricodificato.txt` conterrà:

```
 Partenza $5Destinazione $3Costo
 $2Parigi $9New York $410$2
 $4Roma $9  Londra $5700
 $2Sidney $6Los Angeles $42$3
```

Si scrivano due funzioni, in grado di effettuare, rispettivamente, la compressione e la decompressione. I prototipi delle funzioni siano
int comprimi(FILE *fin, FILE *fout);
int decomprimi(FILE *fin, FILE *fout);

In caso di errore le funzioni ritornano come risultato 0, diversamente ritornano il numero di caratteri scritto nel file in uscita. Il main permette all'utente di selezionare (in base a un opportune input) codifica oppure decodifica, apre i file quello in input e quello in output. chiama la funzione selezionata. Si consiglia di usare un terzo file per il testo de-compresso (evitando così di sovrascrivere, perdendolo, il file originale `sorgente.txt`)

Esercizio 2.

Competenze: IO su file, manipolazioni di caratteri;
Categoria: problemi di codifica applicati a testi/caratteri (4.2.2)

Un file contiene un testo composto da un numero non noto di caratteri.

Lo scopo del programma è di ricodificare il testo, generando un secondo file, nel quale i caratteri sono stati ricodificati secondo le regole seguenti:

I caratteri numerici sono ricodificati in questo modo:

- I caratteri numerici ('0'..'9') sono ricodificati nel carattere numerico posto k posizioni più avanti, con k che, partendo da 0, viene incrementato ad ogni carattere numerico ricodificato (ATTENZIONE: gli incrementi sono effettuati MODULO 10, cioè, arrivati a 9 si riparte da 0). Se ad esempio il file iniziasse con la riga: "Il numero 248 è pari", il '2' verrebbe ricodificato (k parte da 0) in '2'+0 = '2' (k diventa 1), il '4' in '4'+1 = '5' (k diventa 2) e '8' viene ricodificato in '8'+2 = '0' (perché arrivati a '9' si riparte da '0').
  I caratteri alfabetici sono invece ricodificati in questo modo:
- Se un carattere alfabetico è preceduto da un carattere non alfabetico, resta inalterato
- Se è preceduto da un carattere alfabetico (sia c0 il carattere precedente), il suo codice si sposta di h posizioni in avanti, nell'insieme dei caratteri alfabetici (con h = c0-'A', se c0 è maiuscolo, h=c0-'a', se c**0** è minuscolo. L'incremento di h è modulo 26, cioè arrivati alla 'z' o 'Z' (a seconda che il carattere ricodificato sia maiuscolo o minuscolo) si riparte da 'a' o 'A'.

Il risultato della ricodifica sia salvato su un secondo file (I nomi dei file siano opportunamente acquisiti da tastiera).

Esempio:
Se il contenuto del file è:

```
Apelle figlio di Apollo
fece una palla di pelle di pollo
tutti i pesci vennero a galla
per vedere la palla di pelle di pollo
fatta da Apelle figlio di Apollo.
```

Il file di uscita `ricodificato.txt` conterrà:

```
Aptept fntema dl Apdozn
fjlp uhh ppall dl ptept dl pdozn
```

```
tngzh i ptlnv vzmzdui a ggrcc
ptk vzcgxb ll ppall dl ptept dl pdozn
ffyrr dd Aptept fntema dl Apdozn.
```

Si scrivano due funzioni, in grado di effettuare, rispettivamente, la codifica e la decodifica.
I prototipi delle funzioni siano:

```
int codifica(FILE *fin, FILE *fout);
int decodifica(FILE *fin, FILE *fout);
```

In caso di errore le funzioni ritornano come risultato 0, diversamente ritornano il numero di caratteri scritto nel file in uscita. Il main permette all'utente di selezionare (in base a un opportuno input) codifica oppure decodifica, apre i file quello in input e quello in output. chiama la funzione selezionata.

Esercizio 3.
Competenze: IO su file, manipolazioni di caratteri
Categoria: problemi di elaborazione testi (Dal problema al programma: 3.3)

Elaborazione testi a livello di singoli caratteri

E' dato un file testo, contenente solo caratteri alfabetici, numerici, segni di punteggiatura (. , ; : ! ?), spazi e a-capo. Si vuole generare un secondo file contenente il testo, letto dal primo file e trasformato nel modo seguente:

- Sostituire le cifre numeriche con il carattere '*'.
- Inserire uno spazio dopo ogni segno di punteggiatura, a meno che questo non sia già seguito da uno spazio o un a-capo.
- Il primo carattere alfabetico successivo a un punto, punto esclamativo o interrogativo deve essere maiuscolo; se non lo è, trasformarlo in maiuscolo (anche se tra la punteggiatura e il carattere alfabetico ci sono uno o più spazi o a capo).
- Le righe del testo devono essere al massimo di 25 caratteri (a-capo escluso). Se sono più lunghe vanno troncate (anche a metà di una parola) aggiungendo un a-capo esattamente dopo il venticinquesimo carattere.
- Al termine di ogni riga vanno aggiunti, prima dell'a-capo, eventuali spazi per raggiungere i 25 caratteri.
- Ogni riga deve terminare con il numero di caratteri del file originale trascritti nella riga, nel formato: `" | c:%d \n"`.

I nomi dei file sono costanti (il primo file è input.txt, il secondo testo.txt) e definiti con #define (es. `#define filein "input.txt"`).

Esempio:

Contenuto di input.txt	Contenuto di testo.txt:
Caratterizzata da un passato turbolento, in epoca medievale Rouen fu devastata piu' volte da incendi ed epidemie e durante la Guerra dei Cent'Anni fu occupata dagli inglesi. nel 1431 nella sua piazza centrale la giovane Giovanna d'Arco (Jeanne d'Arc) fu processata per eresia e arsa sul rogo!durante la seconda	Caratterizzata da un pass| c:25 ato turbolento, in | c:19 epoca medievale Rouen fu | c:25 devastata piu' volte | c:21 da incendi ed epidemie e | c:25 durante la Guerra | c:18 dei Cent' Anni fu occupat| c:24

guerra mondiale gli Alleati bombardarono ampie zone della citta', soprattutto il quartiere che si estende a sud della cattedrale.	a dagli inglesi. | c:17 Nel **** nella sua piazza| c:25 centrale la giovane | c:21 Giovanna d' Arco (Jeanne| c:23 d' Arc) fu processata | c:22 per eresia e arsa sul rog| c:25 o! Durante la seconda | c:21 guerra mondiale gli Allea| c:25 ti bombardarono | c:16 ampie zone della citta', | c:25 soprattutto il quartiere | c:25 che si estende a sud dell| c:25 a cattedrale. | c:13

Esercizio 4.
Competenze: IO su file, manipolazioni di caratteri
Categoria: problemi di verifica applicati a un testo (Dal problema al programma: 3.4)

Verifica di espressioni

E' dato un file testo expr.txt in cui ogni riga contiene una espressione, composta unicamente da numeri interi non negativi (quindi senza segno), simboli di operazioni aritmetiche (+, -, *, / e %) e parentesi tonde aperte o chiuse.
Si scriva un programma che verifichi la correttezza sintattica delle espressioni scritte nel file, secondo le regole seguenti:

1. Bilanciamento parentesi: le parentesi devono essere bilanciate, cioè il numero totale di parentesi aperte deve coincidere con quello delle parentesi chiuse, e ogni parentesi chiusa deve corrispondere a una precedente parentesi aperta.
2. Non ci possono essere spazi all'interno dei numeri, mentre sono ammessi spazi (da ignorare) in qualunque altra posizione all'interno dell'espressione
3. Successione corretta tra operandi e operatori: ogni operazione è costituita da una sequenza di almeno due operandi intercalati da operatori (simboli di operazione aritmetica) infissi. Un operando può essere rappresentato da un numero oppure da una espressione (eventualmente un numero) racchiusa tra parentesi.

In pratica, la regola 3 può essere verificata controllando che:

- Dopo una parentesi aperta può esserci solo un operando (una parentesi aperta oppure un numero), quindi non può esserci un operatore.
- Dopo un operando (quindi dopo una parentesi chiusa o un numero) ci può essere solo un'altra parentesi chiusa (se rispetta la regola 1), un operatore o la fine dell'espressione
- Dopo un operatore ci deve essere un operando (una parentesi aperta oppure un numero)

O in alternativa, la regola 3 può essere verificata controllando che:

- Prima di un operando (parentesi aperta o numero) può esserci solo una parentesi aperta, un operatore, oppure l'inizio dell'espressione
- Prima di una parentesi chiusa ci può solo essere un operando (quindi una parentesi chiusa o un numero)
- Prima di un operatore ci può solo essere un operando (una parentesi chiusa o un numero)

Quando incontra un errore in una espressione, il programma deve stampare un messaggio di errore (che deve includere il numero di espressione contenente l'errore), ignorare il resto della riga e passare

all'espressione successiva.

Esempio:

expr.txt	Messaggi stampati a video (espressioni numerate a partire da 1):

`(3 + 2) / 7`
`((3 - 7) 4)`
`(3 - 2) +`
`(8 * 9 1) * 4`
`((4*(2+23)) / (7-3))`
	`Errore nella lettura di un numero (espressione 2)`
	`Errore nella lettura di un operando (espressione 3)`
	`Errore nella lettura di un numero (espressione 4)`

***Suggerimenti**: utilizzare un contatore delle parentesi attualmente aperte, usare una variabile che "ricorda" l'ultimo tipo di carattere letto (o gli ultimi due), …*

Esercizio 5.
Competenze: IO formattato da file, manipolazione di numeri, costrutti condizionali e iterativi.
Categoria: problemi di verifica su una sequenza di numeri (Dal problema al programma: 3.4)

Verifica di sequenza di numeri

E' data una sequenza di numeri interi, contenuta in un file testo (numeri.txt, definito con #define), nel quale gli interi sono separati da spazio o a-capo. Occorre verificare che l'i-esimo numero x_i (con $i \geq 2$) sia, rispetto ai due numeri precedenti (x_{i-1} e x_{i-2}), pari alla loro somma ($x_{i-2} + x_{i-1}$), differenza ($x_{i-2} - x_{i-1}$), prodotto ($x_{i-2} * x_{i-1}$) o quoziente (x_{i-2} / x_{i-1}) (attenzione a evitare le divisioni per 0!). Se un dato non è corretto, va scartato e si passa a verificare il successivo (ignorando il dato scartato). Occorre inoltre individuare il massimo e il minimo tra i dati della sequenza (ignorando i dati eventualmente scartati).

Al termine stampare a video:
1. il risultato della verifica, cioè se tutti i dati rispettano la regola o se qualcuno (indicare quanti) sia stato scartato
2. I valori massimo e minimo tra i numeri in sequenza, escludendo quelli eventualmente scartati.

Contenuto di numeri.txt:	Messaggi stampati a video:	Motivazione:

`12`
`3`
`4`
`7 -3`
`0`
`4`
`1`
`3`
`3 9`
`11`
	`Numero massimo: 12`
	`Numero minimo: -3`
	`Numeri scartati: 2`
		Vengono scartati:
		• 0 perchè non è risultato di alcuna operazione su 7 e -3
		• 11 perchè non è risultato di alcuna operazione su 3 e 9