LABORATORIO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO
2019–2020

INDICE

Ultima revisione: 23 febbraio 2020.

In questa lezione svilupperete una serie di esercizi per poi integrarli in un unico programma finale. Il livello complessivo dell'esercitazione non è lontano da quello richiesto da un tema d'esame tipico, anche se la quantità di lavoro richiesta è un poco minore, ed il testo contiene, in alcuni punti critici, molti più suggerimenti del solito. La gestione della memoria allocata dinamicamente vi richiederà particolare cura. Il programma finale consente alcune semplici elaborazioni su un file di testo caricato in memoria come un array di stringhe, una per ciascuna riga del file. Una possibile soluzione completa è allegata a questo documento. Pure allegato a questo documento troverete il file di testo `lorem.txt` che potrete usare per fare dei test. Cercate di completare l'esercitazione in non più di tre ore. È importante che leggiate tutto il testo prima di cominciare a scrivere il codice, per farvi un'idea di cosa richieda il programma finale.

Esercizio 1

Progettazione e test delle strutture dati.

Usando `typedef`, definite un tipo di dato che rinominerete `testo`, adatto a memorizzare un elenco di stringhe di lunghezza variabile. Usate una struttura di due campi. Il primo campo, di nome `f`, è di tipo puntatore a puntatore a `char`. Il secondo campo, di nome `n`, è un intero il cui valore rappresenterà il numero di stringhe nell'elenco. Scrivete una funzione di prototipo `void mostra(testo)` che visualizzi su terminale le stringhe dell'elenco, senza aggiungere fra una stringa e l'altra alcun carattere. Testate la funzione scrivendo una procedura `main` nella quale dichiarerete e inizializzerete con qualche stringa di prova una variabile di tipo `testo`, passandola poi a `mostra`.

Suggerimento. Il campo `f` può essere pensato come un puntatore alla prima stringa dell'elenco. Per accedere alle successive stringhe dell'elenco potrete usare, come sapete, la sintassi con le quadre e gli indici tipica degli array.

Suggerimento. 'Puntatore a puntatore a `char`' può confondere, all'inizio, ma basta ricordare quanto segue. Se `T` è un tipo qualunque, `*T` è il tipo 'puntatore a `T`'. Ora sostituiamo a `T` in `*T` il tipo 'puntatore a `char`', ossia `char *`. Conclusione: la dichiarazione di `f` richiesta dalla traccia è `char **f`.

Suggerimento. Supponiamo di aver dichiarato la variabile `t` di tipo `testo`. Per inizializzare il campo `n` di `t`, basta accedervi come al solito con `t.n`. supponiamo di aver posto il valore di `n` a 3. Allora, per inizializzare il campo `f` di `t` con delle stringhe, dichiariamo e inizializziamo prima un array di 3 stringhe, per esempio così:

```
char *p[]={"Bobarchio\n", "Potarchio\n", "Mariangiongiangela\n"};
```

Ora facciamo puntare `t.f` alla prima stringa di questo array:

```
t.f = p;
```

A questo punto possiamo testare la funzione `mostra` passandole `t` in argomento.

Esercizio 2

Caricamento dei dati da file.

Scrivete una funzione di prototipo `int carica(char *nf, testo *txt)` che riceva in ingresso una stringa `nf`, da interpretarsi come nome di un file, e un puntatore

`txt` a `testo`. La funzione apre in lettura il file di nome `nf`. La funzione assume che il file sia costituito da una prima riga contenente un intero $n > 0$, seguita da n successive righe. La funzione legge l'intero n, alloca la memoria necessaria a contenere un testo di n righe, legge e memorizza le righe, chiude il file e restituisce il controllo. La funzione restituisce i seguenti valori:

-1 : Errore nell'apertura del file.
-2 : L'intero n non è positivo, o la prima riga del file non contiene un intero.
-3 : Errore nell'allocazione del testo.
-4 : Errore nell'allocazione di una frase del testo.
0 : Nessun errore.

Scrivete una semplice funzione `main` per testare il funzionamento di `carica`, usando anche la funzione `mostra` dell'Esercizio 1 e il file di testo `lorem.txt` allegato a questo documento.

Suggerimento. Notate che è necessario allocare separatamente memoria per il testo (cioè per la variabile di tipo puntatore a `testo`), e per le sue varie righe. Fate un disegno su carta di come dovrebbe essere allocata la memoria in un esempio semplice! Per ciascuna riga letta dovrete allocare memoria opportunamente. A questo fine dovrete calcolare la lunghezza della stringa letta, che sarà inizialmente memorizzata in un array di `char` ausiliario. Dovrete poi copiare la stringa nella memoria allocata allo scopo. Per la lunghezza di una stringa, usate `size_t strlen(char *)`. (Potete usare il tipo `int` in luogo di `size_t` ai fini di questo esercizio.) Per copiare la stringa `s` sulla stringa `d`, usate `char * strcpy(char *d, char *s)`. Entrambe le funzioni sono dichiarate nel file di intestazione `string.h`.

Esercizio 3

Funzione `aggiungi`.

Scrivete una funzione di prototipo `int aggiungi(testo *txt, char *s)` che riceva in ingresso una stringa `s` ed un puntatore a testo `txt` e accodi una copia di `s` come ultima riga del testo `txt`. Se si verifica un errore nella riallocazione del testo o della frase la funzione restituisce -1 o -2, rispettivamente. Se non vi sono errori la funzione restituisce 0.

Suggerimento. Dovrete usare sia `realloc` che `malloc`.

Esercizio 4

Scrittura del testo su file.

Scrivete una funzione di nome `scrivisuf` e di prototipo appropriato che riceva in ingresso una stringa, da interpretarsi come nome di un file, e un testo. La funzione apre il file in scrittura e copia sul file, una per volta e nell'ordine, le righe del testo. Terminata la copia, la funzione restituisce il controllo al chiamante. La funzione restituisce 0 se non vi sono errori, e -1 se vi è un errore di apertura del file.

```
Vincenzos-MacBook-Pro:Lab11 enzo$ ./a.out lorem.txt
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>
```

FIGURA 1. Il menu del programma.

```
Vincenzos-MacBook-Pro:Lab11 enzo$ ./a.out lorem.txt
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>1
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nunc lobortis mattis orci, in fringilla elit volutpat at.
Praesent erat mi, egestas quis tempor ac, convallis in dolor.
Curabitur a justo ac nisl mollis blandit.
Phasellus non rhoncus lectus.
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>2
Digita frase da aggiungere.
>Ma quando finisce 'sto laboratorio?
Frase aggiunta.
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>1
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nunc lobortis mattis orci, in fringilla elit volutpat at.
Praesent erat mi, egestas quis tempor ac, convallis in dolor.
Curabitur a justo ac nisl mollis blandit.
Phasellus non rhoncus lectus.
Ma quando finisce 'sto laboratorio?
```

FIGURA 2. Esempio di esecuzione.

ESERCIZIO 5

Funzione `main` *e programma completo*.

Nota bene. Dalla riga di comando, e non acquisito dall'utente dopo l'avvio del programma.

Scrivete una funzione `main` che riceva dalla riga di comando il nome di un file. Se non vi sono argomenti sulla riga di comando il programma termina con errore. Altrimenti, il programma apre il programma carica in una variabile di tipo `testo` il testo contenuto nel file; in caso di errori, il programma visualizza un messaggio appropriato e termina. Se il caricamento va a buon fine, il programma visualizza il menu in Fig. 1. Se l'utente digita 4 il programma termina. Se l'utente digita 1 il programma visualizza il testo. Se l'utente digita 2 il programma chiede all'utente di inserire una frase e la accoda al testo corrente. Se l'utente digita 3 il programma chiede all'utente di inserire un nome di file e scrive il testo corrente su quel file. Gestite il caso in cui l'utente digiti una scelta inesistente. Si vedano le Figg. 2 e 3 per due esempi di esecuzione.

Suggerimento. Avete deallocato la memoria? Anche se alla terminazione del programma il sistema operativo compie automaticamente questa azione, è meglio abituarsi a liberare sempre la memoria allocata quando essa non serve più.

```
Vincenzos-MacBook-Pro:Lab11 enzo$ ./a.out lorem.txt
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>1
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nunc lobortis mattis orci, in fringilla elit volutpat at.
Praesent erat mi, egestas quis tempor ac, convallis in dolor.
Curabitur a justo ac nisl mollis blandit.
Phasellus non rhoncus lectus.
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>3
Digita il nome del file.
>unnomeacaso.txt
File scritto.
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>234asdfgwst89
Scelta inesistente.
1.      Mostra.
2.      Aggiungi frase.
3.      Scrivi su file.
4.      Esci.
>
```

FIGURA 3. Esempio di esecuzione.

DIPARTIMENTO DI MATEMATICA *Federigo Enriques*, UNIVERSITÀ DEGLI STUDI DI MILANO, VIA CESARE SALDINI, 50, I-20133 MILANO