Università degli studi di Roma "La Sapienza"
Corsi di Studi in Informatica e in Tecnologie Informatiche
Laboratorio di Programmazione
a.a. 2003/2004

Parte I Progetto

- **Pubbblicato il 22 Marzo 2004**
- **da consegnarsi entro le ore 24 di 14 Aprile 2004**

Indice

Introduzione.

Il progetto deve essere sviluppato in standard C sotto LINUX, usando, quando utile e possibile, le funzioni della libreria standard.

Si consiglia di mettersi al lavoro non appena noto il testo del progetto seguendo le indicazioni del docente.

Bisognerebbe anche progettare bene tutti gli aspetti del programma prima di cominciare a scrivere il codice. E' in generale molto più efficiente verificare, eventualmente correggere ed eseguire ogni programma quasi a mano, piuttosto che cominciare a scrivere codice al calcolatore senza avere una chiara idea di cosa si vuole ottenere.
Parte del codice da utilizzare sarà stato sviluppato nell'ambito dei corsi di Prigrammazione II e di Laboratorio di Programmazione.

File consegnati oltra la scadenza non saranno accettati. I nomi delle funzioni e delle costanti simboliche introdotte non devono essere modificati, altrimenti non si potrà provvedere alla correzione automatica.

Lo scopo di questo progetto è quello di far sperimentare allo studente la costruzione di un semplice sistema software, nello sviluppo del quale egli dotrà avvalersi delle tecniche apprese durante i corsi di Programmazione e di Laboratorio di Programmazione e in particolare

1. **Programma su più file,**
2. **allocazione dinamica della memoria,**
3. **lettura di file di testo,**
4. **testing e debugging di programmi**

Le funzioni da costruire in questo progetto devono

1. aprire e leggere da un file di testo un testo, restituito come una stringa di caratteri
2. trasformare una stringa di caratteri in un vettore di interi sotto particolari condizioni
3. ottenere una stringa di caratteri da un vettore di interi.

Per facilitare il riuso del codice prodotto si utilizzino i seguenti files: protPart1.h (un file header con tutti i prototipi e le costanti simboliche), funzParte1.c, (un file con le funzioni utilizzate), testParte1.h e testParte1.c (i files rispettivamente con i prototipi e le funzioni utilizzate per il test) e il file main.c (il file che contiene il main utilizzato per le verifiche). Si includino gli opportuni file di libreria.

1. Le funzioni.

Le funzioni sviluppate in questa parte del progetto costituiscono un primo passo verso la costruzione di un sistema per la cifratura e la decifratura di testi, basate su un metodo molto semplice. Tutte le funzioni devono essere corredate di precondizioni e postcondizioni, pena la loro non considerazione in fase di correzione. Inoltre, quando necessario, si devono introdurre opportune funzioni di verifica della correttezza dell'input e/o dell'output da utilizzare con assert nel corpo delle funzioni. Si deve usare l'allocazione dinamica e quindi gestire oculatamente la memoria durante l'esecuzione del programma. Si introduca una costante simbolica DIMALF (si usi 26 per le prove di testing) destinata a contenere il numero delle lettere dell'alfabeto utilizzato.

1.1 openFile

I messaggi da cifrare sono letti da un file, per prima cosa serve dunque una funzione che apre un file, con un determinato nome e in una determinata modalità di apertura.

Prototitipo della funzione: FILE* openFile(**char*** nFile, **char*** modoAcc);

La funzione ha due parametri di input: nFile, il nome del file, e modoAcc, la modalità di accesso al file e restituisce un puntatore al file aperto, se l'apertura è andata a buon fine, NULL altrimenti.

1.2. leggiTesto

Un messaggio sarà composto di parole e frasi in italiano, anche se faremo riferimento all'alfabeto internazionale. I messaggi sarano letti da file di testo e trasformati in una stringa di caratteri minuscoli, sostituendo ogni lettera accentata

con la corrispondente non accentata e ogni carattere non alfabetico con uno spazio.

Prototitipo della funzione: **char*** leggiTesto(FILE* fPtr);

La funzione ha come parametro di ingresso il puntatore al file aperto e restituisce una stringa che contiene il contenuto del file "normalizzato" secondo le regole date (caratteri minuscoli e spazi, niente accenti o altri caratteri).

Per esempio se il file di input contiene il seguente testo

Non c'è giorno più bello del sabato!

O anche

Non c'e' giorno piu' bello del sabato!

la stringa prodotta da leggiTesto sarà

"n o n c e g i o r n o p i u b e l l o d e l s a b a t o "

Qui serve una funzione di verifica del formato dell'output della leggiTesto. Tale funzione restituisce 1 se la stringa in input contiene solo caratteri minuscoli e lo spazio, 0 altrimenti e il cui prototipo è

int verMin(**char*** testo);

Bisognerà poi costruire due funzioni di conversione da interi a caratteri e viceversa.

1.3. charToInt

Questa funzione prende in input una stringa di caratteri minuscoli e spazi e produce un vettore di interi, compresi tra 0 e DIMALF e in modo che allo spazio corrisponda lo 0, al carattere 'a' sia associato 1, a 'b' sia associato 2,…, a 'z' sia associato DIMALF.

Prototitipo della funzione: **int*** charToInt(**char** * testo)

Questa funzione riceve in input una stringa di caratteri minuscoli e spazi e produce un vettore di interi di dimensione di uno maggiore della lunghezzza della stringa. Nella prima posizione il vettore contiene la lunghezza della stringa, mentre nelle successive gli interi tra 0 e DIMALF cui corrispondono le lettere del messaggio.

Per esempio, data in input la stringa di 40 caratteri dell'esempio precedente (in cui gli spazi finali corrispondono a caratteri di punteggiatura e di andata a capo):

“n o n c e g i o r n o p i u b e l l o d e l s a b a t o ”

si dovrà produrre il vettore di interi (di lunghezza 41!):

40 14 15 14 0 3 0 5 0 0 7 9 15 18 14 15 0 16 9 21
0 0 2 5 12 12 15 0 4 5 12 0 19 1 2 1 20 15 0 0 0

Si consiglia di introdurre la verifica del corretto formato dell’input (precondizione!) utilizzando la funzione verMin con l’assert. Analogamente è utile introdurre una funzione di verifica del formato dell’output di charToInt. Tale funzione restituisce 1 se il vettore in input contiene un qualunque positivo nella prima posizione e numeri tra 0 e DIMALF nelle altre, 0 altrimenti e il cui prototipo è:

int verMessInt(**int*** testo);

1.4. intToChar.

Questa funzione prende in input un vettore di interi, la cui prima posizione contiene la sua dimensione –1 e le altre interi compresi tra 0 e DIMALF, e produce la stringa di caratteri minuscoli e spazi corrispondente.

Anche qui si utilizzi la funzione assert e le opportune funzioni su introdotte per il controllo del formato dell’input e dell’output.

Prototitipo della funzione: **char*** intToChar(**int** *vett);

Esempio

input:

40 14 15 14 0 3 0 5 0 0 7 9 15 18 14 15 0 16 9 21
0 0 2 5 12 12 15 0 4 5 12 0 19 1 2 1 20 15 0 0 0

output: “n o n c e g i o r n o p i u b e l l o d e l s a b a t o ”

2. Testing

Per cercare eventuali errori nelle funzioni fin qui definite si devono effettuare dei test. Conviene definire delle funzioni di supporto per poter poi riutilizzare gli stessi test in caso di correzioni. Queste funzioni andranno prototipate e definite nel file che contiene il main, che le utilizzerà e sono:

1. **char*** FileToC(**char*** nome);

Questa funzione riceve in input il nome di un file, lo apre in lettura (utilizzando la openFile, chiama la leggitesto per ottenere la stringa da restituire e chiude il file.

2. **int** verCtItC(**char*** mess);

Questa funzione riceve in input una stringa di caratteri, con charToInt la trasforma in un vettore di interi, con intToChar la ritrasforma in una stringa di caratteri e restituisce il risultato del confronto tra la stringa in input, e quella ottenuta con intToChar.

Nel main quindi basterà chiamare le due funzioni per ogni file costruito e costruire una printf in modo da ottenere un opportuno messaggio diagnostico in funzione del risultato di verCtItC.

Per visualizzare i risultati, potranno essere utili delle funzioni di stampa:

void stampaStringa(**char** *msg);

void stampaNum(**int** *vett);

Si consiglia, in ogni modo, di fare ampio uso di compilazione condizionata per queste visualizzazioni.

Si testino le funzioni fin qui prodotte su diversi tipi di testo:

1. testo su una sola riga con uno o più accenti
2. testo su più righe e con uno o più accenti e caratteri di spaziatura (tabulazioni,..)