Fondamenti di Informatica T-1, 2008/2009 – Modulo 2
Prova d'Esame 1 – 22 Dicembre 2008

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit.zip`** contenente i file necessari (*solution* di VS2005 e progetto compresi).
Avvertenze per la consegna: nominare i file sorgenti come richiesto nel testo del compito, apporre all'inizio di ogni file sorgente un commento contenente i propri dati (**cognome, nome, numero di matricola**) e il **numero** della prova d'esame. Al termine, **consegnare tutti i file sorgenti** e i file contenuti nello StartKit. Rispettare le specifiche, in particolare inserire le funzioni nei file specificati fra parentesi dopo il nome della funzione. Chi non rispetta le specifiche sarà opportunamente penalizzato.
Nota 1: **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non affrontabile" di errori di compilazione.
Nota 2: il main non è opzionale; i *test* richiesti vanno implementati.
Consiglio: per verificare l'assenza di *warning*, effettuare di tanto in tanto *"Rebuild All"*.

Un negozio d'informatica vende computer assemblati secondo le preferenze dei clienti. Si deve realizzare un programma che gestisca gli ordini richiesti, calcolando il prezzo totale di ogni ordine e le quantità di componenti utilizzati per assemblare gli elaboratori ordinati.
Le informazioni riguardanti gli ordini ricevuti e le caratteristiche dei componenti sono memorizzate in due file di testo distinti. Ogni ordine si compone di un codice identificativo (intero), un destinatario (stringa, al più di 50 caratteri) e da un insieme di, al più, 10 linee d'ordine (vettore, da dimensionarsi staticamente); ogni linea d'ordine è composta da un componente e dalla quantità (intera) di pezzi ordinati per quel componente. I componenti sono caratterizzati da un codice identificativo (intero), una descrizione (stringa, al più di 80 caratteri) e un prezzo in euro (float).

Esercizio 1 – Componenti e lettura da file di testo (file componente.h/componente.c)

Si definiscano una o più strutture dati atte a memorizzare le informazioni sui *componenti*. Si realizzi una funzione **`readComponentsFromTxt`** che prenda in ingresso il nome di un file di testo contenente le informazioni sui componenti e restituisca un *vettore di componenti* allocato dinamicamente (della dimensione strettamente necessaria). Le informazioni di un singolo componente sono memorizzate su una riga del file secondo il seguente formato:

- Codice identificativo, seguito da un ';'
- Stringa di descrizione, seguita da un ';'
- Prezzo

Per un esempio di *file* di componenti, si veda **ComponentList.txt** contenuto nello *start kit*. Si faccia l'ipotesi che i componenti siano sempre memorizzati *in ordine crescente rispetto al codice*. Sfruttando questa ipotesi, si realizzi la funzione **`findComponentByCode`** che implementi una *ricerca binaria in base al codice* sul vettore dei componenti.
Nel **`main`**, si verifichi il corretto funzionamento del codice realizzato leggendo le informazioni presenti nel *file* **`ComponentList.txt`** fornito con lo *start kit*, realizzando un ciclo di stampa di tutti i componenti letti ed effettuando la ricerca di un componente sia con un codice presente, sia con un codice assente.

Esercizio 2 – Ordini e lettura da file di testo (file ordine.h/ordine.c)

Si definiscano una o più strutture dati atte a memorizzare le informazioni sugli *ordini* e le relative *linee d'ordine*. Si realizzi una funzione **`readOrdersFromTxt`** che prenda in ingresso il nome di un *file* di testo contenente le informazioni sugli ordini e restituisca la *lista degli ordini* letti.
Le informazioni di un singolo ordine sono memorizzate su una riga del file secondo il seguente formato:

- Codice identificativo, seguito da un ';'
- Stringa relativa al destinatario, seguita da un ';'
- Sequenza di linee d'ordine, ognuna seguita da un ';' (tranne l'ultima, che è seguita direttamente dal fine linea). Ogni linea d'ordine, a sua volta, è composta di:
    - Il codice identificativo del componente a cui si riferisce, seguito da una ','
    - Il quantitativo ordinato per quel componente.

Fondamenti di Informatica T-1, 2008/2009 – Modulo 2
Prova d'Esame 1 – 22 Dicembre 2008

Per un esempio di *file* di ordini, si veda **OrderList.txt** contenuto nello start kit. Attenzione: durante il caricamento di ogni ordine è necessario riempire il corrispondente vettore delle linee d'ordine avendo cura di *tradurre il codice del componente nel componente corrispondente*.
Realizzare una funzione di nome **`printOrderList`** che consenta di stampare tutti gli ordini, ognuno dei quali deve riportare, in un formato "amichevole", anche l'elenco delle linee d'ordine, il loro prezzo e il prezzo totale dell'ordine.
Nel **`main`**, si verifichi il corretto funzionamento del codice realizzato leggendo con **`readOrdersFromTxt`** le informazioni presenti nel *file* **`OrderList.txt`** fornito con lo *start kit*, e stampando con **`printOrderList`** gli ordini letti.

Esercizio 3 – Calcolo dei quantitativi totali dei componenti utilizzati (file ordine.h/ordine.c)

Si realizzi una funzione **`printUsedComponents`** che prenda in ingresso una lista di ordini e un vettore di componenti e stampi a video, per ogni componente del vettore, il numero totale dei pezzi ordinati per quel componente, considerando tutti gli ordini contenuti nella lista. La stampa di un componente deve essere effettuata solo se ne è stato ordinato almeno un pezzo.
Nel **`main`**, si verifichi il corretto funzionamento del codice realizzato utilizzando il vettore di componenti letto all'esercizio 1 e la lista di ordini letta all'esercizio 2.

Esercizio Facoltativo – Deallocazione della memoria dinamica

Si realizzi una funzione **`destroyOrderList`** che, presa in ingresso una lista di ordini, ne deallochi l'intero spazio occupato in memoria. Si inserisca poi nel main il codice necessario alla deallocazione della lista di ordini (utilizzando la funzione precedente) e del vettore dei componenti.