

Fondamenti di Informatica T1  
Prova Pratica – 10 Febbraio 2020  
Compito

**Prima di cominciare:** si scarichi il file **StartKit.zip** contenente i file di esempio.

**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**). Al termine, **consegnare tutti i file sorgente ed 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. **NON SARANNO CORRETTI** gli elaborati che presenteranno un numero "non ragionevole" di errori di compilazione.

**Consiglio:** per verificare l'assenza di *warnings*, effettuare di tanto in tanto un *Rebuild All*.

La società *Costa Armonia* si occupa di trasporto merci per via marittima, con base a Livorno. Per questo la compagnia ha sviluppato un componente software per la gestione delle partenze delle navi e dei ritardi. Gestendo infatti solo carico merci e non passeggeri, la società può permettersi sia di ritardare che di anticipare le partenze dei suoi mezzi.

La società dispone di alcuni file con salvati diversi dati, il primo file è il file delle navi ("navi.txt"), il file è organizzato nel seguente modo:

**codice\_nave tipo**

Dove **codice\_nave** è una stringa di 5 caratteri (terminatore escluso) che identifica univocamente la nave, **tipo** invece è un carattere che indica se la nave è del tipo veloce 'V' o lento 'L'.

Il file "Partenze.txt" contiene la lista delle partenze, il file è formato nel seguente modo:

**id\_viaggio partenza arrivo codice\_nave orario ritardo**

Dove **id\_viaggio** è un intero che identifica in modo univoco il viaggio, **partenza** e **arrivo** sono due stringhe di 40 caratteri (senza spazi) contenenti rispettivamente il nome della città di partenza e di arrivo, **codice\_nave** è una sequenza di 5 caratteri che identifica la nave, **orario** è un intero contenente l'orario di partenza in secondi dalla mezzanotte (es: se l'orario di partenza è 11:55:12 allora la variabile conterrà il valore  $11*60*60+55*60+12=42912$  ), **ritardo** è un intero contenente il ritardo in secondi.

### **Esercizio 1 - Lettura da file (*compagnia\_navi.h/ compagnia\_navi.c*)**

Si definisca un'opportuna struttura dati **nave** al fine di rappresentare e tenere traccia delle navi presenti.

Il candidato realizzi poi la funzione:

**nave\* leggi\_navi (char\* nomefile, int\* num\_n)**

che, ricevuto come parametro di ingresso il nome di un file, provveda ad aprirlo e ne legga il contenuto, ovvero la lista delle navi. Non è noto a priori quante navi siano presenti nel file, quindi sarà necessario allocare memoria dinamicamente nella dimensione opportuna. Tale funzione deve restituire la dimensione logica dell'array

Fondamenti di Informatica T1  
Prova Pratica – 10 Febbraio 2020  
Compito

tramite l'intero **num\_n**.

Si realizzi inoltre una procedura:

**void stampa\_nave (nave n)**

Che stampi a video (in maniera leggibile – usare spazi, separatori, fine linea, etc.) le informazioni relative ad una singola nave, preso come ingresso dalla procedura.

Si definisca la struttura **partenza** in modo da contenere tutte le informazioni relative alle tratte. Si crei una funzione:

**partenza \* leggi\_partenze(char \* nomefile, int \* num\_p)**

Si implementi la funzione in modo tale che, dato il nome del file contenente i dati delle partenze, crei dinamicamente l'array di strutture **partenza** della dimensione esatta e restituisca, oltre al vettore, il numero di partenze lette **num\_p**.

Si realizzi la procedura:

**void stampa\_partenza(partenza p)**

Che stampa tutte le informazioni relative ad una partenza passata in ingresso.

Il candidato abbia cura di realizzare nel main opportuni test al fine di verificare il corretto funzionamento delle funzioni di cui sopra, sfruttando il file di testo "*navi.txt*" e "*partenze.txt*" (presenti nello **StartKit.zip**). Una volta verificato il corretto funzionamento delle funzioni, il candidato non cancelli il codice nel main ma si limiti a commentarlo.

**Esercizio 2 – Nuova partenza(compagnia\_navi.h/ compagnia\_navi.c)**

Si definisca la funzione

**int trova\_nave(char \* codice, nave\* n, int num\_n)**

che, preso in ingresso una stringa contenente il codice di una nave, cerca all'interno del vettore delle navi **n**, se questa è presente; **num\_n** contiene la dimensione logica dell'array di navi. La funzione dovrà restituire 1 se la nave è stata trovata, 0 altrimenti.

Si definisca la funzione

**int nuova\_partenza(FILE \* fp, nave \* n, int num\_n)**

che prende in ingresso il puntatore ad un file già opportunamente aperto, un vettore **n** di strutture dati **nave** e la relativa dimensione logica; tale funzione deve inserire all'interno del file delle partenze, una nuova partenza (inserita da tastiera), avendo cura di verificare che il codice della nave della nuova partenza sia presente nel vettore delle navi (si utilizzi la funzione **trova\_nave**).

La funzione deve restituire 1 se l'inserimento è andato a buon fine, -1 se la nave non è stata trovata, 0 in caso di altri errori.

Si realizzino nel main le opportune istruzioni per verificare il corretto funzionamento della funzione, scrivendo la nuova partenza nel file "*partenze.txt*".

Fondamenti di Informatica T1  
Prova Pratica – 10 Febbraio 2020  
Compito

**Esercizio 3 – Ordinamento partenze (compagnia\_navi.h/ compagnia\_navi.c)**

Si realizzi la procedura:

**void ordina\_partenze(partenza \* p,int num\_p)**

Che preso in ingresso un vettore **p** di strutture dati **partenza**, con la sua dimensione logica, si occupi di ordinare le partenze sulla base dell'ora di partenza effettivo (**orario + ritardo**) dal minore al maggiore.

**Esercizio 4 – Anticipa partenze (compagnia\_navi.h/ compagnia\_navi.c)**

Si realizzi la procedura:

**void anticipa\_partenze(partenza \* p, int num\_p, nave\* n, int num\_n)**

che prende in ingresso l'array **p** di strutture dati **partenza**, con la sua dimensione logica, e l'array **n** di strutture dati **nave** e la relativa dimensione logica. La procedura deve calcolare se è possibile anticipare la partenza delle navi lente in ritardo.

La procedura deve considerare tutte le partenze che hanno un ritardo (campo **ritardo** > 0); se la partenza è associata ad una nave lenta (controllare il campo **tipo**), la procedura deve stampare a video tutte le tratte che hanno un orario di partenza effettivo (**orario + ritardo**) inferiore rispetto alla partenza in ritardo.

**Esercizio 5 – Main (main.c)**

Il candidato realizzi un programma in grado di:

- 1) Leggere Navi e Partenze dai rispettivi file creando due array;
- 2) Mostrare a video tutti le navi e le partenze;
- 3) Inserire una nuova partenza;
- 4) Anticipare le partenze dei treni a causa dei ritardi;
- 5) Ordinare le partenze e stampare a video le partenze ordinate;
- 6) Mostrare a video le possibili partenze che possono essere anticipate;
- 7) Deallocare (al termine del programma) tutte le strutture allocate dinamicamente.