**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**) e il **numero** della prova d’esame. 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*.

I gestori del chiosco di piadina romagnola *iPiada* hanno deciso di avvalersi di un sistema informatizzato per facilitare l'organizzazione della propria attività. In particolare sfruttano questo sistema per raccogliere le ordinazioni e calcolare successivamente il conto per ogni cliente.

I prodotti che si possono acquistare nel chiosco rientrano in tre diverse categorie, cioè piadine, crescioni o bevande. I clienti del chiosco sono anch'essi divisi in due tipologie, i clienti occasionali e i clienti abituali, la cui fedeltà è ricompensata non facendo loro pagare le bevande ordinate.

In un file di testo, denominato “*prodotti.txt*”, viene salvato l'elenco dei prodotti in vendita. Ogni riga corrisponde ad un prodotto ed è formata dai seguenti campi:

**codiceProdotto nomeProdotto tipoProdotto prezzo**

dove il codiceProdotto è una stringa di 4 caratteri alfanumerici che identifica il prodotto in vendita, il nome è una stringa (senza spazi) di al massimo 50 caratteri, il tipo del prodotto è un carattere che può assumere tre valori (uno per ogni categoria di prodotto, quindi '*p*', '*c*' e '*b*'), il prezzo è un float che rappresenta il costo in euro per ogni unità di prodotto.

In un secondo file “*ordinazioni.txt*” sono memorizzate le ordinazioni fatte dai clienti del chiosco. In particolare in ogni riga del file sono memorizzati:

**codiceCliente nomeCliente codiceProdotto quantità**

dove il codice è una stringa assegnata ad ogni cliente per renderne più semplice l'identificazione (al massimo 10 caratteri senza spazi), il nome del cliente è una stringa senza spazi, il codice del prodotto individua il prodotto ordinato e la quantità è un intero che indica le unità di prodotto desiderate.

Il primo carattere del codice del cliente è necessariamente una lettera che identifica la sua classe e permette di distinguere i clienti abituali (carattere '*A*') da quelli occasionali (carattere '*O*')

Nel file possono essere presenti più ordinazioni relative allo stesso cliente e riguardanti prodotti diversi, ma si assume che non siano presenti più righe con la stessa coppia di valori *<codiceCliente,codiceProdotto>*.

#### Esercizio 1 - Lettura da file (gestore\_chiosco.h/gestore\_chiosco.c)

Si definisca un’opportuna struttura dati **Prodotto** al fine di rappresentare i prodotti in vendita, in particolare tenendo traccia di codice identificativo del prodotto, nome, tipologia e prezzo.

Il candidato realizzi poi la funzione:

**Prodotto \* leggiProdotti (char \* nomefile, int \* dimProd);**

che, ricevuto come parametro di ingresso il nome di un file, provveda ad aprirlo e ne legga il contenuto, ovvero i prodotti messi in vendita dal chiosco. Si noti che non è noto a priori quanti prodotti siano registrati nel file: sarà quindi necessario determinare quanti prodotti vi siano memorizzati, e poi allocare memoria dinamicamente nella dimensione opportuna. Tale funzione deve restituire la dimensione logica dell’array in **dimProd.**

Procedendo in maniera simile al punto precedente, definire un’opportuna struttura dati **Ordinazione** al fine di rappresentare le ordinazioni fatte dai clienti, in particolare considerando codice identificativo e nome del cliente, codice e quantità di prodotto ordinato.

Realizzare quindi la funzione:

**Ordinazione \* leggiOrdinazioni (char \* nomefile, int \* dimOrd);**

che provveda a leggere da un file di testo (il cui nome è passato come parametro di ingresso) le ordinazioni ricevute; anche in questo caso non è dato sapere a priori il numero di righe presenti nel file di testo e quindi sarà necessario allocare dinamicamente l'array di strutture **Ordinazione**. Nuovamente, restituire la dimensione logica del vettore tramite il parametro **dimOrd**.

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 “*prodotti.txt*”, contenente un elenco di prodotti, e “*ordinazioni.txt*”, con le ordinazioni (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 – Inserimento nuova ordinazione (gestore\_chiosco.h/gestore\_chiosco .c)**

Si realizzi una funzione:

**int inserisci\_ordinazione (char \* nomefile);**

che provveda creare una nuova ordinazione prendendo in input i dati forniti da tastiera dall'utente e successivamente scrivere le informazioni relative a questa ordinazione nel file indicato dal parametro di ingresso (mantenendo il formato visto nell'esercizio precedente, cioè codice cliente, nome cliente, codice prodotto e quantità); la funzione dovrà ritornare un intero che può assumere due valori, '0' nel caso in cui la scrittura sia terminata con successo e '-1' in caso contrario.

Questa funzione dovrà anche controllare che le informazioni introdotte dall'utente siano valide, in particolare verificando che il primo carattere del codice del cliente sia 'A' oppure 'O' e che la quantità specificata sia un valore positivo; la funzione deve anche controllare che l'ordinazione non sia già presente tra quelle memorizzate nel file di testo (la coppia *<codiceCliente,codiceProdotto>* è univoca), in qual caso deve stampare a video un messaggio di errore senza scrivere nel file.

Il candidato realizzi nel main opportuni test per verificare il corretto funzionamento delle funzioni implementate, scrivendo la nuova ordinazione nel file di testo “*ordinazioni.txt*” (quello usato anche in fase di lettura).

#### **Esercizio 3 – Calcolo conti (gestore\_chiosco.h/gestore\_chiosco .c)**

Il candidato definisca una procedura:

**void calcola\_conti(Ordinazione \* o, int dimOrd,Prodotto \* p, int dimProd);**

che preso in ingresso un vettore **o** di strutture dati **Ordinazione** e un vettore **p** di strutture dati **Prodotto** (con le rispettive dimensioni logiche), calcoli il conto totale per ogni cliente, sommando i prezzi tutti i prodotti ordinati, tenendo ovviamente conto anche le relative quantità – il candidato si rammenti inoltre del fatto che i clienti possono appartenere a due classi distinte, quelli abituali che non dovranno pagare le bevande ordinate e quelli occasionali che non godono di tale sconto; questa procedura deve inoltre stampare a video le informazioni così calcolate, ovvero il codice ed il nome del cliente e il conto da pagare.

Si realizzino nel main le opportune istruzioni per verificare il corretto funzionamento delle funzioni implementate.

#### Esercizio 4 – Main (main.c)

Il candidato realizzi un programma in grado di:

1. Leggere i prodotti in vendita e le ordinazioni dei clienti dai rispettivi file creando due array.
2. Inserire una nuova ordinazione, scrivendola anche nel file delle ordinazioni.
3. Calcolare e mostrare a video i conti di tutti i clienti.
4. Deallocare (al termine del programma) tutte le strutture allocate dinamicamente.