RELAZIONE ALGORITMI E PROGRAMMAZIONE

Appello del 21/02/2020 – Prova di programmazione (18 punti)

Salvatore Mallemaci s261358

STRUTTURE DATI

- Utilizzo un vettore di *struct (*piatti) ElencoPiatti*, in modo da poter memorizzare l'intero file in un'unica struttura dati, organizzata in base al formato del file.
- Utilizzo un'ulteriore *struct MENU*, in modo da poter gestire in output il menù in questione, riportandone piatti col relativo prezzo e costo totale del menù. Essa è definita da un vettore di interi *menu, contenente gli indici dei piatti del vettore *ElencoPiatti,* e da un float *prezzo* del menù.
- Utilizzo un quasi ADT per gestire tutto ciò che ha a che fare col menù, implementando funzioni di inizializzazione, stampa e calcolo delle combinazioni.

STRATEGIE RISOLUTIVE

- **Acquisizione dell'elenco da file:** Dopo aver letto la prima riga del file contenente il numero di piatti (*N*) proposti dal ristorante tramite una prima chiamata alla funzione *fscanf*, segue la dichiarazione e l'allocazione di un vettore di *struct ElencoPiatti*, grande appunto N. Tramite un ciclo for registro dunque tutti i dati in questione, riempendo il vettore di *struct*, ordinato come da lettura, definendo la gerarchia tra piatti utilizzata nel resto del codice.

- **Modello del calcolo combinatorio utilizzato**: *COMBINAZIONI RIPETUTE.*
  Ho scelto tale modello poiché la struttura dati in questione è costituita da elementi distinti, dunque ho a che fare con un SET. Da testo, so che un elemento può essere ripetuto all'interno del menù per un massimo di due volte, perciò ho utilizzato un vettore di occorrenze *mark*, settato a due per ogni piatto a disposizione. Sono dunque ammesse ripetizioni.
  Infine, l'ordine dei piatti non conta. Si tratta, di conseguenza, di calcolare tutte le possibili combinazioni ripetute, organizzando N elementi a P a P (N= numero dei piatti offerti dal ristorante, P= numero di piatti costituente ogni singolo menù generato).
  Utilizzo una funzione *COMB_RIP_WRAPPER* per comodità, dichiarando, allocando e inizializzando *pos*, *start*, il vettore *sol* e vettore *mark*, la quale non fa altro che chiamare la funzione ricorsiva vera e propria *COMB_RIP*, partendo da *pos*=0 e *start*=0.
  In *COMB_RIP*:
  Nel corpo della funzione, segue un ciclo for per i che va da start (inizialmente zero) a N (numero dei piatti complessivi presenti all'interno del file ricevuto in input).
  Dopo aver appurato la disponibilità del piatto in questione *(if (mark[i]) > 0)*, decremento proprio *mark[i]*, segnalando il fatto che ho selezionato un piatto per il menù corrente e memorizzo in *sol[pos]* l'indice corrente, in modo tale da poter risalire successivamente al piatto sfruttando quest'ultimo come indice del vettore *ElencoPiatti*.
  Infine, richiamo la funzione ricorsiva, ricorrendo solo su pos+1, per poi effettuare il cosiddetto "backtracking", incrementando *mark[i]*, in modo da rimettere a disposizione il piatto in questione per il prossimo menù e incrementando start, così da "riprendere il for da questa posizione".
  Giunti alla condizione di terminazione, andrebbe memorizzata la singola soluzione generata in un nuovo nodo, costituente un nodo appunto del BST, mantenendone le proprietà grazie alla *MENUcompare*, all'interno della quale struttura si inserisce il menù in questione. Tale parte è stata omessa all'interno del codice, poiché in sede d'esame non sono arrivato a trattarla in maniera sufficiente. Infine, seguirebbe la stampa del menù, tramite visita in-order ad hoc.

- **Funzione MENUcompare:** Tale funzione richiede che, dati due menù, si restituisca il più conveniente tra i due, in base alla gerarchia tra i piatti. Perciò prima verifico che i due menù abbiano prezzi diversi e in tal caso si "ritorna" il menù col prezzo più basso.
  In caso di parità tra i prezzi, si confrontano gli indici dei piatti costituenti i due menù e dunque, nel caso del codice da me proposto, confronto *menu1[i]* e *menu2[i]*, in quanto tali vettori contengono gli indici dei piatti. Ho impostato la funzione come tipo int, restituendo 1 nel caso in cui il primo menù sia più conveniente, 2 nel caso in cui il secondo menù sia più conveniente. Il caso di parità non è contemplato in quanto, durante la generazione delle singole combinazioni ripetute, non si potrà generare un menù esattamente uguale.

DIFFERENZE PIÙ SIGNIFICATIVE TRA COMPITO CARTACEO E CODICE ALLEGATO ALLA RELAZIONE

- Ho aggiunto un semplice controllo inerente ai parametri su linea di comando.
- Ho incluso la libreria *"string.h"*, in modo da poter utilizzare senza problemi la *strdup.*
- Ho spostato la *typedef struct piatti* all'interno del file header "menu.h", in quanto quasi ADT, in modo che potesse essere visibile a tutti i moduli costituente il mio programma. (Nel compito cartaceo tale definizione era stata effettuata nel main.c e dunque il modulo menu non era in grado di vedere tale struttura).
- Per comodità ho creato un'ulteriore struct (MENU) per gestire i menù, come segue:
  *typedef struct{int *menu; float prezzo;}MENU;*
  Di conseguenza, parti di codice in cui utilizzavo il vettore di interi e di indici *menu, vedono adesso la presenza di ElencoMenu->menu, dove appunto *ElencoMenu è un puntatore alla struct citato poco sopra.
- Ho spostato nel main.c la funzione *MENUcompare*.
- Ho introdotto due funzioni semplici all'interno del modulo menu, ossia la *MENU *MENUinit* e la *MENUprint*, mirate rispettivamente all'inizializzazione di un menù e alla stampa.
- Ho aggiunto ai parametri della *COMB_RIP sol* e *mark*, i quali erano stati dichiarati ed inizializzati correttamente nella *COMB_RIP_WRAPPER* ma non erano stati passati per dimenticanza alla funzione ricorsiva vera e propria. Ne riporto il prototipo:
  *void COMB_RIP(MENU *ElencoMenu,piatti *ElencoPiatti,int pos,int start,int *sol,int *mark,int P,int N);*
- Ho aggiunto le due *free* finali, deallocando **ElencoPiatti* e **ElencoMenu.*

COMMENTI FINALI

In conclusione, ho implementato il modulo BST.h, per dare un'idea di come avrei voluto gestire la parte inerente ai BST, ossia alla memorizzazione di ciascun BST, costituente in pratica un menù.