Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

Prima di cominciare: si scarichi dal sito **`http://esamix.labx`** il file **`StartKit3A.zip`** contenente i file necessari (*progetto Visual Studio* ed eventuali altri file di esempio).
Avvertenze per la consegna: 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.
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*, eseguire di tanto in tanto *"Rebuild All"*.

Una nota catena di supermercati, al fine di fidelizzare i propri clienti, offre la possibilità di ottenere dei punti in base alle spese effettuate presso i propri supermercati.
A tal fine, in un file di testo, memorizza su ogni riga il **`nome`** di un prodotto (una stringa di al più 35 caratteri, senza spazi) ed i **`punti`** associati a tale prodotto (un intero). Non è noto a priori quante righe siano memorizzate nel file. Si veda, a titolo di esempio, il file "**`punteggi.txt`**" fornito nello StartKit.
Tutti i registratori di cassa sono collegati ad un computer, e tutti comunicano contemporaneamente con tale computer. Per ogni prodotto consegnato alla cassa, viene aggiunta una riga ad un file di testo di nome "**`registro.txt`**". In ogni riga viene memorizzato l'**`id`** del cliente (un intero), il **`nome`** del prodotto (ancora una stringa di al più 35 caratteri, senza spazi), ed il **`numero`** di confezioni acquistate di tale prodotto (un intero). Dopo qualche ora di funzionamento, nel file sono presenti, mischiate tra di loro, le informazioni relative agli acquisti effettuati da diversi clienti. Inoltre, a causa dell'accesso contemporaneo di tutte le casse a tale sistema, accade (per errore) che siano inserite righe relative a clienti esistenti, dove però il numero di oggetti acquistati è pari a zero. Non è noto a priori quante righe siano presenti nel file. Si veda, a titolo di esempio, il file "**`registro.txt`**" fornito nello StartKit.
Si vuole realizzare un programma che, per ogni cliente presente nel file "**`registro.txt`**", calcoli il punteggio raggiunto attraverso le spese effettuate, tenendo conto dei punteggi specificati e del numero di oggetti acquistati. Ad esempio, se un prodotto di nome "pasta" ha associato un punteggio di 15, e un utente ne acquista 3 confezioni, il punteggio di tale utente sarà incrementato di 3x15=45 punti.

Esercizio 1 – Struttura dati Cliente e funzioni di lettura/scrittura (moduli element.h e prodotti.h/prodotti.c)

Si definisca un'opportuna struttura dati **`Prodotto`**, al fine di rappresentare il nome ed il punteggio assegnato ad un prodotto, come descritto in precedenza.
Si definisca poi la funzione:

```
list leggiProdotti(char* fileName);
```

che, ricevuto in ingresso il nome di un file di testo contenente i punteggi relativi ad un prodotto, restituisca una lista di strutture dati di tipo **`Prodotto`**.
Si definisca anche una procedura:

```
void stampaProdotti(list l);
```

che, ricevuta in ingresso una lista di strutture dati di tipo **`Prodotto`**, stampi a video il contenuto di tale lista.
Infine si realizzi una funzione:

```
int punteggio(list l, char * nome);
```

che, ricevuti in ingresso una lista di strutture dati di tipo Prodotto ed il nome di un prodotto, restituisca il punteggio assegnato al prodotto. Qualora il prodotto non sia presente in lista, la funzione restituisca il valore 0.
Il candidato infine abbia cura di inserire nel **`main(…)`** alcune istruzioni per "testare" la corretta implementazione delle funzioni, usando come prova il file di esempio fornito nello StartKit, e de-allocando eventuale memoria allocata dinamicamente. Tali istruzioni per il test devono rimanere, commentate, nella funzione main.

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

Esercizio 2 – Lettura degli acquisti effettuati (modulo element.h e prodotti.h/prodotti.c)

Si definisca un'opportuna struttura dati **`Acquisto`**, al fine di rappresentare ogni singolo acquisto, come descritto in precedenza (si rappresenti quindi nella struttura l'**`id`** del cliente, il **`nome`** del prodotto ed il **`numero`** di oggetti acquistati).
Si realizzi una funzione:

```
Acquisto * leggiAcquisti(char * fileName, int * dim);
```

che, ricevuto in ingresso il nome di un file contenente tutti gli acquisti effettuati, legga da tale file le informazioni e le memorizzi in un'area di memoria allocata dinamicamente. La funzione deve restituire un array di elementi di tipo **`Acquisto`**, e tramite il parametro **`dim`** la dimensione di tale array.
Si definisca poi una procedura:

```
void stampaAcquisti(Acquisto * v, int dim);
```

che stampi a video le strutture dati di tipo **`Acquisto`** contenute nel vettore **`v`** passato come parametro, di dimensione **`dim`**.
Si inseriscano poi nel main apposite istruzioni per verificare la corretta implementazione delle funzioni. Al termine di questo test, si abbia cura di de-allocare la memoria allocata dinamicamente. Si lascino tali istruzioni, eventualmente commentate, nel main.

Esercizio 3 – Ordinamento degli acquisti effettuati (modulo element.h/element.c e prodotti.h/prodotti.c)

Si realizzi una funzione:

```
Acquisto * filtraEOrdina(Acquisto * v, int dim, int *dimResult);
```

che, ricevuti in ingresso un array **`v`** di strutture dati di tipo **`Acquisto`** e la sua dimensione **`dim`**, restituisca un nuovo vettore di strutture dati di tipo **`Acquisto`**, contenente i dati originali contenuti in **`v`**, esclusi gli acquisti con numero di oggetti pari a zero. Inoltre, la funzione restituisca tramite il parametro **`dimResult`** la dimensione del nuovo vettore; tale vettore sia ordinato in maniera crescente in base al codice identificativo del cliente; in caso di parità di codice identificativo, si consideri il nome del prodotto e si utilizzi il criterio lessicografico (ancora in senso crescente). Non è necessario che il vettore restituito sia della minima dimensione necessaria; si suggerisce di creare tale vettore e di riempirlo con i dati "corretti", e poi di ordinarlo in un secondo momento.
Si inseriscano poi nel main apposite istruzioni per verificare la corretta implementazione delle funzioni. Al termine di questo test, si abbia cura di de-allocare la memoria allocata dinamicamente. Si lascino tali istruzioni, eventualmente commentate, nel main.

Esercizio 4 – Stampa dei punteggi totali dei clienti (main.c)

Il candidato realizzi nella funzione **`main(…)`** un programma che, usando le informazioni fornite tramite i file "punteggi.txt" e "registro.txt", stampi a video, per ogni cliente presente nel file "**`registro.txt`**", il punteggio totale ottenuto da tale cliente tramite tutti i suoi acquisti registrati. A tal proposito, si ricorda che ogni prodotto acquistato contribuisce con un punteggio specifico, che deve essere moltiplicato per il numero di prodotti di quel tipo acquistati dal cliente. Si suggerisce di usare le funzioni sviluppate negli esercizi precedenti.

Al termine del programma, il candidato abbia cura di de-allocare tutta la memoria allocata dinamicamente, ivi compresa la memoria allocata per le liste.

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

"element.h":

```
#ifndef _ELEMENT_H
#define     _ELEMENT_H

#define DIM_NOME 36



typedef struct {
      char nome[DIM_NOME];
      int punteggio;
} Prodotto;

typedef Prodotto element;


typedef struct {
      int id;
      char nome[DIM_NOME];
      int numero;
} Acquisto;



int compare(Acquisto c1, Acquisto c2);

#endif      /* _ELEMENT_H */
```

"element.c":

```
#include "element.h"

#include <stdio.h>
#include <string.h>


int compare(Acquisto c1, Acquisto c2) {
      if (c1.id < c2.id)
            return -1;
      else if (c1.id>c2.id)
            return +1;
      else
            return strcmp(c1.nome, c2.nome);
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

"list.h"

```
#ifndef LIST_H
#define LIST_H
#include "element.h"
typedef struct list_element {
    element value;
    struct list_element  *next;
} item;
typedef  item* list;
typedef  int boolean;
/* PRIMITIVE  */
list emptylist(void);
boolean empty(list);
list cons(element, list);
element head(list);
list tail(list);
void freelist(list l);
#endif
```

"list.c":

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"
/* OPERAZIONI PRIMITIVE */
list  emptylist(void) {       /* costruttore lista vuota */
    return NULL; }

boolean  empty(list l) {      /* verifica se lista vuota */
    return (l==NULL); }

list  cons(element e, list l) {
  list t;       /* costruttore che aggiunge in testa alla lista */
  t=(list)malloc(sizeof(item));
  t->value=e;
  t->next=l;
  return(t); }

element head(list l) { /* selettore testa lista */
  if (empty(l)) exit(-2);
  else return (l->value); }

list  tail(list l) {         /* selettore coda lista */
  if (empty(l)) exit(-1);
  else return (l->next); }

void freelist(list l) {
      if (empty(l)) return;
      else {
            freelist(tail(l));
            free(l); }
      return; }
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

"prodotti.h":

```
#ifndef PRODOTTI
#define PRODOTTI

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "element.h"
#include "list.h"

list leggiProdotti(char* fileName);
void stampaProdotti(list l);
int punteggio(list l, char * nome);

Acquisto * leggiAcquisti(char * fileName, int * dim);
void stampaAcquisti(Acquisto * v, int dim);

Acquisto * filtraEOrdina(Acquisto * v, int dim, int *dimResult);

#endif
```

"prodotti.c":

```
#include "prodotti.h"


list leggiProdotti(char* fileName) {
	FILE * fp;
	list result;
	Prodotto temp;

	result = emptylist();
	if ( (fp=fopen(fileName, "rt")) == NULL) {
		printf("Error opening the file: %s\n", fileName);
		system("pause");
		exit(-1);
	}
	while (fscanf(fp, "%s %d", temp.nome, &temp.punteggio) == 2)
		result = cons(temp, result);

	fclose(fp);
	return result;
}


void stampaProdotti(list l) {
	while (!empty(l)) {
		printf("%s %d\n", head(l).nome, head(l).punteggio);
```

```
            l= tail(l);
      }
      return;
}


int punteggio(list l, char * nome) {
      int result = 0;
      int trovato = 0;
      while (!empty(l) && !trovato) {
            if (strcmp(nome, head(l).nome) == 0) {
                  result = head(l).punteggio;
                  trovato = 1;
            }
            else
                  l = tail(l);
      }
      return result;
}


Acquisto * leggiAcquisti(char * fileName, int * dim) {
      FILE * fp;
      Acquisto temp;
      Acquisto * result;
      int i;

      *dim = 0;
      if ( (fp=fopen(fileName, "rt")) == NULL) {
            printf("Error opening the file: %s\n", fileName);
            system("pause");
            exit(-1);
      }
      while (fscanf(fp, "%d %s %d", &temp.id, temp.nome, &temp.numero)==3)
            (*dim)++;
      result = (Acquisto *) malloc(sizeof(Acquisto) * *dim);
      rewind(fp);
      i = 0;
      while (fscanf(fp, "%d %s %d", &temp.id, temp.nome, &temp.numero)==3) {
            result[i] = temp;
            i++;
      }
      fclose(fp);
      return result;
}


void stampaAcquisti(Acquisto * v, int dim) {
      int i;
      for (i=0; i<dim; i++)
            printf("%d %s %d\n", v[i].id, v[i].nome, v[i].numero);
```

```
}


void scambia(Acquisto *a, Acquisto *b) {
      Acquisto tmp = *a;
      *a = *b;
      *b = tmp;
}
void bubbleSort(Acquisto v[], int n) {
      int i, ordinato = 0;
      while (n>1 && !ordinato) {
            ordinato = 1;
            for (i=0; i<n-1; i++)
                  if (compare(v[i],v[i+1])>0) {
                        scambia(&v[i],&v[i+1]);
                        ordinato = 0;
                  }
            n--;
      }
}
Acquisto * filtraEOrdina(Acquisto * v, int dim, int *dimResult) {
      Acquisto * result;
      int i;

      *dimResult = 0;
      result = (Acquisto *) malloc(sizeof(Acquisto) * dim);
      for (i=0; i<dim; i++)
            if (v[i].numero !=0) {
                  result[*dimResult] = v[i];
                  (*dimResult)++;
            }
      bubbleSort(result, *dimResult);
      return result;
}
```

Fondamenti di Informatica T-1, 2010/2011 – Modulo 2

Prova d'Esame 3A di Mercoledì 16 Febbraio 2011 – tempo a disposizione 2h

"main.c":

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "element.h"
#include "list.h"
#include "prodotti.h"


int main() {
	/* Es. 1 */ {
		list prod;
		char nome[DIM_NOME];
		prod = leggiProdotti("punteggi.txt");
		stampaProdotti(prod);
		printf("Inserire il nome di un prodotto: ");
		scanf("%s", nome);
		printf("il punteggio assegnato al prodotto %s e': %d\n", nome,
punteggio(prod, nome));
		freelist(prod);
		system("pause");
	}
	/* Es. 2 */ {
		Acquisto * v;
		int dim;
		v = leggiAcquisti("registro.txt", &dim);
		stampaAcquisti(v, dim);
		free(v);
		system("pause");
	}
	/* Es. 3 */ {
		Acquisto * v;
		Acquisto * v_filt;
		int dim;
		int dimOrd;
		v = leggiAcquisti("registro.txt", &dim);
		v_filt = filtraEOrdina(v, dim, &dimOrd);
		stampaAcquisti(v_filt, dimOrd);
		free(v);
		free(v_filt);
		system("pause");
	}
	/* Es. 4 */ {
		list prod;
		Acquisto * v;
		Acquisto * v_filt;
		int dim;
		int dimOrd;
		int current_id;
```

```
            int tot;
            int i;

            prod = leggiProdotti("punteggi.txt");
            v = leggiAcquisti("registro.txt", &dim);
            v_filt = filtraEOrdina(v, dim, &dimOrd);
            if (dimOrd >0) {
                  current_id = v_filt[0].id;
                  tot = 0;
                  for (i=0; i<dimOrd; i++) {
                        if (v_filt[i].id == current_id) {
                              tot = tot + v_filt[i].numero * punteggio(prod,
v_filt[i].nome);
                        }
                        if (v_filt[i].id != current_id || i==(dimOrd-1)) {
                              printf("Cliente %d punti: %d\n", current_id, tot);
                              current_id = v_filt[i].id;
                              tot = v_filt[i].numero * punteggio(prod,
v_filt[i].nome);
                        }
                  }
            }
            free(v);
            free(v_filt);
            freelist(prod);
            system("pause");
      }
      return 0;
}
```

"punteggi.txt":

```
pane 3
pasta 5
tartufo 20
olio 10
pannolini 7
birra 4
vino 15
```

"registro.txt":

```
17 pane 2
17 pasta 8
17 libro 1
17 olio 2
134 pasta 10
134 pannolini 4
134 vino 6
569 pane 4
134 birra 6
569 pasta 4
17 birra 0
```