Esercizi Laboratorio Programmazione

Prof. Stefano Guerrini
Canale P-Z

22 marzo 2002

Gli esercizi dovranno essere inviati **entro le 24 di lunedì 25/03/02**.

Si prenda la solita strutura dati per la rappresentazione delle liste

```
typedef struct nodo {
    int info;
    struct nodo *next;
} NODO;
typedef NODO *LISTA;
```

contenuta in `liste.h` (vedi appendice A) insieme ai prototipi delle funzioni di base per la manipolazione di liste definite nel modulo `liste.c` (vedi appendice B).

1 Esercizio 1: rovesciamento di liste

Si devono scrivere delle funzioni che eseguono il rovesciamento di una lista, ovvero, delle funzioni che, ricevuta una lista `ls`, costruiscono una lista `lsinv` il cui il primo elemento è l'ultimo di `ls`, il secondo elemento è l'ultimo di `ls`, il terzo è il terzultimo, etc. Le funzioni da scrivere sono 4 ed i loro prototipi, contenuti nel file `es1.h` (vedi appendice C), sono:

```
LISTA cproviter(LISTA);
LISTA cprovric(LISTA);
void roviter(*LISTA);
void rovric(*LISTA);
```

Le suddette funzioni devono soddisfare le seguenti specifiche:

1. la lista risultato della `cproviter` e della `cprovric` è una *nuova lista*, ovvero, la lista di input non deve essere modificata dalle suddette funzioni;
2. la `cproviter` è una funzione *iterativa*, la `cprovric` è *ricorsiva*;
3. la `roviter` e la `rovric` non creano una nuova lista in cui mettere il risultato, ma *costruiscono la lista risultano usando gli stessi nodi della lista di input*;

4. la `roviter` è una funzione *iterativa*, la `rovric` è *ricorsiva*.

Si possono utilizzare le funzioni definite nel modulo `liste.c`, il cui header file è `liste.h` (vedi appendice B e A). Infatti, al momento della verifica, il modulo `es1.c` che gli studenti dovranno scrivere verrà compilato insieme a `provaes1.c` e `liste.c` col comando

```
gcc provaes1.c es1.c liste.c -o provaes1
```

2 Esercizio 2: operazioni su liste

Si devono scrivere le funzioni

```
LISTA fondi(LISTA, LISTA);
LISTA inters(LISTA, LISTA);
```

che prendono come input una coppia di liste (non ordinate e che possono contenere ripetizioni) e producono come risultato:

1. la lista ordinata e senza ripetizioni degli elementi che appaiono in almeno una delle liste, nel caso della `fondi`;
2. la lista ordinata e senza ripetizioni degli elementi che appaiono in entrambe le liste, nel caso della `unisci`.

Il file `es2.h` (vedi appendice E) contiene i prototipi delle suddette funzioni.

Le due funzioni (insieme con le altre funzioni che si ritengono necessarie) dovranno essere implementate in un modulo `es2.c` che verrà poi verificato per mezzo del main in `provaes2.c` (vedi appendice F).

Anche in questo caso si può usare il modulo `liste.c`. Infatti, il modulo `es2.c` inviato dagli studenti verrà compilato insieme a `provaes2.c` e `liste.c` col comando

```
gcc provaes2.c es2.c liste.c -o provaes2
```

A liste.h

```
/* -*- Mode: C -*- */
/* Time-stamp:  <liste.h 02/03/21 11:32:39 guerrini@sgport.dsi.uniroma1.it> */

/*************************************************************
 * header file della
 * libreria per la manipolazione di liste di interi
 *************************************************************/

/* nodo della lista */
typedef struct nodo {
    int info;
    struct nodo *next;
} NODO;

typedef NODO * LISTA;

void inizzlista(LISTA *);
/* inizializza una lista */

int listavuota(LISTA);
/* verifica se una lista e' vuota */

void *insnodo(LISTA *, int, LISTA);
/* crea un nuovo nodo della lista e ne restituisce
   il valore nel parametro passato per riferimento
   i campi info e next del nuovo nodo sono inizializzati
   ai valori del secondo e terzo pararametro, risp.
   ritorna il puntatore al nuovo nodo se OK,
   atrimenti, ritorna NULL
*/

void cancnodo(LISTA *);
/* elimina un nodo passato per riferimento
   il nuovo valore del parametro per riferimento
   e' il campo next del nodo cancellato
   la memoria utilizzata dal nodo viene
   restituita mediante una free */
```

B liste.c

```
/* -*- Mode: C -*- */
/* Time-stamp:  <liste.c 02/03/22 00:27:03 guerrini@sgport.dsi.uniroma1.it> */

/*************************************************************
 * implementazione della
 * libreria per la manipolazione di liste di interi
 *************************************************************/

#include <memory.h>
#include "liste.h"

void inizzlista(LISTA *ls) {
    /* inizializza ls */
    *ls = NULL;
}

int listavuota(LISTA ls) {
    /* verifica se ls e' vuota */
    return (ls == NULL);
}

void *insnodo(LISTA *p, int n, LISTA ls) {
    /* crea un nuovo nodo della lista e ne restituisce in *p
       i campi info e next del nuovo nodo sono inizializzati
       ai valori di n ed ls, rispettivamente
       ritorna il puntatore al nuovo nodo se OK,
       atrimenti, ritorna NULL
    */
    LISTA nuovo = (LISTA)malloc(sizeof(NODO));
    if (nuovo != NULL) {
        nuovo->info = n;
        nuovo->next = ls;
        *p = nuovo;
    }
    return (nuovo);
}

void cancnodo(LISTA *p) {
    /* elimina il nodo puntato da *p
       il nuovo valore di *p e' il campo next
       del nodo cancellato
       la memoria utilizzata dal nodo viene
       restituita mediante una free */
    LISTA ls;
    if (p != NULL && *p != NULL) {
```

```
        ls = *p;
        *p = ls->next;
        free(ls);
    }
}
```

C es1.h

```
/* -*- Mode: C -*- */
/* Time-stamp:  <rov.h 02/03/21 11:48:32 guerrini@sgport.dsi.uniroma1.it> */

LISTA cproviter(LISTA);
/* copia una lista rovesciandola - versione iterativa */

LISTA cprovric(LISTA);
/* copia una lista rovesciandola - versione ricorsiva */

void roviter(LISTA *);
/* rovescia una lista in loco - versione iterativa */

void rovric(LISTA *);
/* rovescia una lista in loco - versione ricorsiva */
```

D provaes1.c

```
/* -*- Mode: C -*- */
/* Time-stamp:  <provaes1.c 02/03/22 02:24:25 guerrini@sgport.dsi.uniroma1.it> */

#include <stdio.h>
#include "liste.h"
#include "es1.h"

void stampalista(LISTA ls) {
    for (; ls != NULL; ls = ls->next)
        printf("%d ", ls->info);
}

int main(void) {
    int n;
    LISTA ls[3];
    /* inizializzazione liste */
    for (n = 0; n < 3; n++)
        inizzlista(ls+n);
    printf("Inserire una sequenza di numeri positivi terminata da un num <=0:\n");
    scanf("%d", &n);
    while (n > 0) {
        insnodo(ls, n, ls[0]);
        scanf("%d", &n);
    }
    putchar('\n');
    printf("Ho letto la lista\n");
    printf("(NB. E' rovesciata rispetto all'ordine di inserimento):\n");
    stampalista(ls[0]);
    putchar('\n');
    putchar('\n');
    /* Provo cproviter */
    printf("** Rovescio la lista copiandola (vers. iterativa) **\n");
    ls[1] = cproviter(ls[0]);
    printf("Verifico che l'input non e' stato toccato:\n");
    stampalista(ls[0]);
    putchar('\n');
    printf("La lista risultato:\n");
    stampalista(ls[1]);
    putchar('\n');
    putchar('\n');
    /* Provo cprovric */
    printf("** Rovescio la lista copiandola (vers. ricorsiva) **\n");
    ls[2] = cproviter(ls[0]);
    printf("Verifico che l'input non e' stato toccato:\n");
    stampalista(ls[0]);
```

```
    putchar('\n');
    printf("La lista risultato:\n");
    stampalista(ls[2]);
    putchar('\n');
    putchar('\n');
    /* Provo roviter */
    printf("** Rovescio la lista in loco (vers. iterativa) **\n");
    roviter(ls);
    printf("Verifico che l'input e' stato rovesciato:\n");
    stampalista(ls[0]);
    putchar('\n');
    putchar('\n');
    /* Provo roviter */
    printf("** Rovescio in loco la lista rovesciata al passo prec ");
    printf("(vers. ricorsiva) **\n");
    rovric(ls);
    printf("Verifico che ho riottenuto la lista di partenza:\n");
    stampalista(ls[0]);
    putchar('\n');
    putchar('\n');
}
```

E es2.h

```
/* -*- Mode: C -*- */
/* Time-stamp:  <es2.h 02/03/22 02:29:51 guerrini@sgport.dsi.uniroma1.it> */

LISTA fondi(LISTA, LISTA);
/* crea la lista ordinata degli elementi che appaiono in almeno
   una delle liste; lascia immodificate le liste di input */

LISTA inters(LISTA, LISTA);
/* crea la lista ordinata degli elementi che appaiono in
   entrambe le liste; lascia immodificate le liste di input */
```

F provaes2.c

```
/* -*- Mode: C -*- */
/* Time-stamp:  <provaes2.c 02/03/22 02:32:51 guerrini@sgport.dsi.uniroma1.it> */

#include <stdio.h>
#include "liste.h"
#include "es2.h"

void stampalista(LISTA ls) {
    for (; ls != NULL; ls = ls->next)
        printf("%d ", ls->info);
}

int main(void) {
    int n;
    LISTA ls[4];
    /* inizializzazione liste */
    for (n = 0; n < 4; n++)
        inizzlista(ls+n);
    printf("Inserire una sequenza di numeri positivi terminata da un num <=0:\n");
    scanf("%d", &n);
    while (n > 0) {
        insnodo(ls, n, ls[0]);
        scanf("%d", &n);
    }
    putchar('\n');
    printf("Inserire una sequenza di numeri positivi terminata da un num <=0:\n");
    scanf("%d", &n);
    while (n > 0) {
        insnodo(ls+1, n, ls[1]);
        scanf("%d", &n);
    }
    putchar('\n');
    printf("Ho letto le liste\n");
    printf("(NB. Sono rovesciate rispetto all'ordine di inserimento):\n");
    stampalista(ls[0]);
    putchar('\n');
    stampalista(ls[1]);
    putchar('\n');
    putchar('\n');
    /* provo fondi */
    printf("** Lista ordinata degli elementi in almeno una lista **\n");
    ls[2] = fondi(ls[0], ls[1]);
    printf("Verifico che l'input non e' stato toccato:\n");
    stampalista(ls[0]);
    putchar('\n');
```

```
    stampalista(ls[1]);
    putchar('\n');
    printf("La lista risultato:\n");
    stampalista(ls[2]);
    putchar('\n');
    putchar('\n');
    /* provo inters */
    printf("** Lista ordinata degli elementi in entrambe le lista **\n");
    ls[3] = inters(ls[0], ls[1]);
    printf("Verifico che l'input non e' stato toccato:\n");
    stampalista(ls[0]);
    putchar('\n');
    stampalista(ls[1]);
    putchar('\n');
    printf("La lista risultato:\n");
    stampalista(ls[3]);
    putchar('\n');
    putchar('\n');
}
```