Algoritmi e Strutture Dati (IN110)
Esercitazione n. 7

Marco Liverani *

Esercizio n. 1

Letta in input una sequenza di numeri interi memorizzarli in una lista e stampare la lista stessa. Eliminare dalla lista i "nodi" che contengono un numero dispari e stampare la lista modificata.

Codifica in linguaggio C

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

struct nodo {
  int info;
  struct nodo *next;
};

struct nodo *leggi_lista(void) {
  struct nodo *p, *primo;
  int i, n;
  primo = NULL;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d numeri interi: ", n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  return(primo);
}

void stampa_lista(struct nodo *p) {
  while (p != NULL) {
    printf("%d ---> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
```

*Università degli Studi Roma Tre, Corso di Laurea in Matematica, Corso di Algoritmi e Strutture Dati (IN110) – sito web del corso http://www.mat.uniroma3.it/users/liverani/IN110/

```
31    return;
32  }
33
34
35  struct nodo *elimina_dispari(struct nodo *primo) {
36    struct nodo *p, *prec;
37    p = primo;
38    prec = NULL;
39    while (p != NULL) {
40      if (p->info % 2 != 0) {
41        if (p == primo) {
42          primo = primo->next;
43          free(p);
44          p = primo;
45        } else {
46          prec->next = p->next;
47          free(p);
48          p = prec->next;
49        }
50      } else {
51        prec = p;
52        p = p->next;
53      }
54    }
55    return(primo);
56  }
57
58  int main(void) {
59    struct nodo *primo;
60    primo = leggi_lista();
61    primo = elimina_dispari(primo);
62    stampa_lista(primo);
63    return(0);
64  }
```

Esercizio n. 2

Letti in input n e m, costruire due liste di n ed m numeri interi casuali compresi tra 0 e 100 e stamparle. Agganciare la lista la cui media dei valori sia minore a quella con la media dei valori maggiore. Stampare la nuova lista.

Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

struct nodo {
  int info;
  struct nodo *next;
};

struct nodo *genera_lista(void) {
  struct nodo *primo=NULL, *p;
  int i, n;
  srand((unsigned)time(NULL));
  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % MAX;
    p->next = primo;
    primo = p;
  }
  return(primo);
}

void stampa_lista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
  return;
}

float media_lista(struct nodo *p) {
  int somma = 0, n = 0;

  while (p != NULL) {
    somma = somma + p->info;
    n++;
    p = p->next;
  }
  return((float)somma/n);
}

```

```
46 void concatena(struct nodo *a, struct nodo *b) {
47   struct nodo *p;
48   p = a;
49   while (p->next != NULL) {
50     p = p->next;
51   }
52   p->next = b;
53   return;
54 }
55
56 int main(void) {
57   struct nodo *u, *v;
58   u = genera_lista();
59   v = genera_lista();
60   stampa_lista(u);
61   stampa_lista(v);
62   if (media_lista(u) > media_lista(v)) {
63     concatena(u, v);
64     stampa_lista(u);
65   } else {
66     concatena(v, u);
67     stampa_lista(v);
68   }
69   return(0);
70 }
```

Esercizio n. 3

Generare due liste ordinate in ordine crescente di n ed m numeri interi positivi casuali minori di 100. Stampare le due liste. Costruire una terza lista ordinata in ordine decrescente, ottenuta "fondendo" le prime due; stampare la nuova lista.

Soluzione

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

struct nodo {
  int info;
  struct nodo *next;
};

struct nodo *genera_lista_ordinata(void) {
  struct nodo *primo=NULL, *p, *q;
  int i, n;

  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % MAX;
    if (primo == NULL || primo->info > p->info) {
      p->next = primo;
      primo = p;
    } else {
      q = primo;
      while (q->next != NULL && q->next->info < p->info) {
        q = q->next;
      }
      p->next = q->next;
      q->next = p;
    }
  }
  return(primo);
}

void stampa_lista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
  printf("NULL\n");
  return;
}
```

```c
struct nodo *fusione(struct nodo *p, struct nodo *q) {
  struct nodo *r, *primo = NULL;
  while (p != NULL && q != NULL) {
    r = malloc(sizeof(struct nodo));
    if (p->info < q->info) {
      r->info = p->info;
      p = p->next;
    } else {
      r->info = q->info;
      q = q->next;
    }
    r->next = primo;
    primo = r;
  }
  while (p != NULL) {
    r = malloc(sizeof(struct nodo));
    r->info = p->info;
    r->next = primo;
    primo = r;
    p = p->next;
  }
  while (q != NULL) {
    r = malloc(sizeof(struct nodo));
    r->info = q->info;
    r->next = primo;
    primo = r;
    q = q->next;
  }
  return(primo);
}

int main(void) {
  struct nodo *a, *b, *c;

  srand((unsigned)time(NULL));
  a = genera_lista_ordinata();
  b = genera_lista_ordinata();
  stampa_lista(a);
  stampa_lista(b);
  c = fusione(a, b);
  stampa_lista(c);
  return(0);
}
```