Esercitazioni di Fondamenti di Informatica - Lez. 9
26/11/2020

Approfondimento sulle liste

1-nel primo esempio si vede che creo una lista,

uso una funzione per aggiungere elementi,

questa operazione senza la funzione addVal sarebbe onerosa poichè dovrei allocare diverse variabili del tipo ELEMENTO .....

2- nel secondo esercizio la reverse della lista (a,b,c,d) si realizza ricorsivamente

stacco la testa (a) (b,c,d) e la attacco alla coda invertita

dove stacco la test a (b) e la attacco alla (c,d) invertita .....

ottengo in pratica : (a) reverse(b reverse (c reverse (d)))

1. **Si consideri una lista dinamica di interi, creare una funzione che ha il seguente prototipo:**

   int somma(ELEMENTO *Testa, int M)

   che restituisce la somma dei soli valori della lista che sono multipli di M.

   Se la lista è vuota, la funzione restituisce il valore 0.

Soluzione

definiamo una struct di interi per la lista

deve avere un tipo per il valore da gestire ed uno per il puntatore ad una struct uguale a se stessa

```
typedef struct El {
  int dato;
  struct El *next;
} ELEMENTO;
```

Idea : si scorre la lista, se dato%M ==0 lo aggiungo a somma

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

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

// l'elemento della lista è del tipo  BOXELEM

typedef struct box{
    int dato;                // un dato
    struct box *next;     // con un puntatore  del tipo
                             struct box
} BOXELEM;


int somma(BOXELEM *testa, int m){
    int sum =0;
    int dato;
    while(testa !=NULL) {
        dato =testa->dato;
        if (dato%m == 0)
                    sum=sum+dato;
            testa=testa->next;
}
return sum;
}

// con la funzione addVal restituisco i puntatore a listabox

BOXELEM *addVal(BOXELEM *testa,int val ){

    BOXELEM *bb = (BOXELEM*)malloc(sizeof(BOXELEM));

    bb->dato=val;
    bb->next= testa;
    return bb;
}

int main() {

    BOXELEM *listabox = NULL;

// senza funzione devo allocare diverse variabili da unire a
listabox altrimenti perderei dei pezzi

    listabox = addVal(listabox,1);
    listabox = addVal(listabox,10);
    listabox = addVal(listabox,2);
    listabox = addVal(listabox,3);
    listabox = addVal(listabox,5);

    // se abbiamouna lista
```

```
    int sum = somma(listabox, 5);

    printf("la somma  dei numeri trovati è %d \n",sum);

    return 0;
    }
```

2. **Si consideri una lista di caratteri, scrivere una funzione che restituisce la lista in ordine inverso**

```
//
//  main.c
//  reverse
//

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


typedef struct nodo_t {
    char lettera;
    struct nodo_t *next;
} nodo_t;

typedef nodo_t* Lista;

Lista InsInTesta( Lista lista, char c );
Lista InsInCoda( Lista lista, char c );

void VisualizzaLista( Lista lista );


Lista Ultimo( Lista lista );
Lista TogliUltimo( Lista lista );
Lista Reverse( Lista lista );


int main() {
    Lista h = NULL;

    h = InsInCoda(h, 'a');
    h = InsInCoda(h, 'b');
```

```
    h = InsInCoda(h, 'c');
    h = InsInCoda(h, 'd');

    VisualizzaLista( h );

    // testiamo ultimo:
    // Lista u = Ultimo(h);
    // printf("%c \n", u->lettera);

    Lista r = Reverse(h);
    VisualizzaLista( r );

    return 0;
}

Lista InsInTesta( Lista lista, char c ) {
    Lista nodo = (Lista) malloc(sizeof(nodo));
    nodo->lettera = c;
    nodo->next = lista;
    return  nodo;
}


Lista InsInCoda( Lista lista, char c ) {

    Lista nodo = (Lista) malloc(sizeof(nodo));
    nodo->lettera = c;
    nodo->next = NULL;

    if (lista == NULL)
        return nodo;

    Lista ultimo = Ultimo(lista);
    ultimo->next = nodo;

    return lista;
}


void VisualizzaLista( Lista lista ) {
    if ( lista == NULL ){
        printf("-|\n");
        return;
    }

    printf("%c -> ", lista->lettera);
    VisualizzaLista(lista->next);
}
```

```
Lista Ultimo( Lista lista ){
    if(lista == NULL){
        return NULL;
    }

    if(lista->next == NULL){
        return lista; // se non c'e' altro, e' ultimo..
    }

    // passo ricorsivo: cerco l' ultimo sul resto delal lista
(coda)
    return Ultimo(lista->next);

}


Lista TogliUltimo( Lista lista ){
    if(lista == NULL){
        return NULL;
    }

    if(lista->next == NULL){
        free(lista);
        return  NULL;
    }

    if(lista->next->next == NULL){
        free(lista->next);
        lista->next = NULL;
        return lista;
    }

    Lista dopoRimozione = TogliUltimo(lista->next);
    return lista;
}


Lista Reverse( Lista lista ){
    if(lista == NULL){
        return NULL;
    }

    if(lista->next == NULL){
        return lista; // se non c'e' altro, e' questo ..
    }

    int primoValore = lista->lettera;
    Lista ultimo = Ultimo(lista);
```

```
    int ultimoValore = ultimo->lettera;

    lista = TogliUltimo(lista);
    VisualizzaLista(lista);

    Lista inversaCoda = Reverse(lista->next);
    inversaCoda = InsInCoda(inversaCoda, primoValore);

    Lista nuovaTesta = InsInTesta(inversaCoda, ultimoValore);

    return nuovaTesta;

}
```