Liste a puntatori: un esercizio passo passo

Prof. Orazio Mirabella

Liste a puntatori: un esercizio passo passo (for dummies)

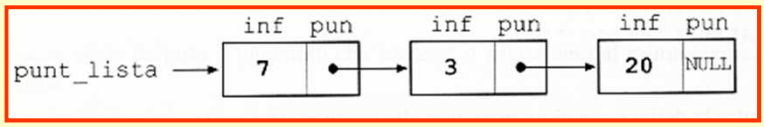
Prof. Orazio Mirabella

Ripassiamo le Liste lineari

- Una lista lineare è una successione di elementi omogenei che occupano in memoria una posizione qualsiasi.
- Ciascun elemento contiene almeno un'informazione e un puntatore per mezzo del quale è legato al successivo.
- L'accesso alla lista avviene con il puntatore al primo elemento. :

  Elemento = informazione + puntatore

- Il puntatore è il riferimento a un elemento, il suo valore è l'indirizzo dell'elemento nella memoria del sistema.

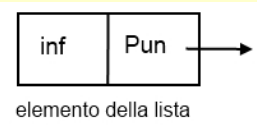


- Il campo puntatore dell'ultimo elemento della lista non fa riferimento a nessun altro elemento; il suo contenuto corrisponde a un segnale di fine lista che in C è il valore NULL.

Elementi della lista

- La parte informazione dell'elemento dipende dal tipo di dati che stiamo trattando.
- In C può essere costituita da uno qualsiasi dei tipi semplici che conosciamo: int, float ecc.
- Nel caso il cui il campo informazione sia di tipo intero, la dichiarazione della struttura di ogni elemento può essere la seguente:

```
struct elemento {
int inf;
struct elemento *pun;
};
```



- L'elemento della lista sarà come mostrato in figura

  La definizione: struct elemento *punt_lista;

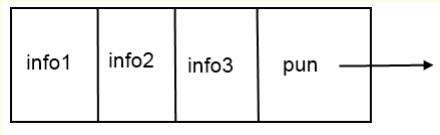
  stabilisce che punt_lista è un puntatore che può riferirsi a variabili di tipo elemento.

Elementi della lista

- La parte informazione di ogni elemento della lista può essere formata da diversi campi. In funzione di questo definiamo la struct.

- Ad esempio nel caso il cui la parte informazione sia costituita da un tipo intero, un float ed un char la dichiarazione della struttura di ogni elemento può essere la seguente:

```
struct elemento {
        int info1;
        float info2;
        char info3;
        struct elemento *pun;
};
```

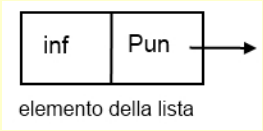


La struct elemento sarà costituita da 4 campi. L'ultimo deve essere un puntatore a struct elemento

Gestione della lista

- Nel seguito faremo riferimento ad una lista in cui, per semplicità, ogni elemento sia costituito solo da un intero ed un puntatore. La dichiarazione della struttura di ogni elemento sarà la seguente:

```
struct elemento {
int inf;
struct elemento *pun;
};
```



- L'elemento della lista sarà come mostrato in figura

- La prima operazione che faremo sarà creare una lista vuota. Successivamente creeremo ed inseriremo man mano i vari elementi della lista, dapprima in modo banale, poi pian piano in modo più efficiente.

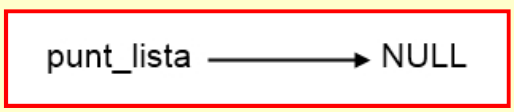
Creiamo una lista vuota

- #include <stdio.h>
- struct elemento {
- int inf;
- struct elemento *pun;};
- main()
- {
- ……
- ……
- ……
- ……
- }

Che cosa metteremo nel main?

Creiamo una lista vuota

- #include <stdio.h>
- struct elemento {
- int inf;
- struct elemento *pun;};
- main()
- {
- struct elemento *punt_lista; // Puntatore alla testa della lista
- punt_lista = NULL ;/* la lista è vuota */
- ……
- ……
- }



Abbiamo così creato una lista vuota in cui il puntatore alla testa della lista ha valore NULL.

Creiamo una lista vuota

```
#include <stdio.h>
struct elemento {
int inf;
struct elemento *pun;};
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL ;/* la  lista è vuota */
{printf("\npunt_lista---> "); // stampa la lista vuota
printf("NULL\n\n");}
}
```

```
punt_lista---> NULL
--------------------------------
Process exited with return
value 0
Press any key to continue . . .
```

Abbiamo aggiunto le istruzioni per stampare la lista vuota in cui il puntatore alla testa della lista ha valore NULL.

Funzione per la visualizzazione della lista.

La seguente funzione va richiamata ogni volta che facciamo una operazione che modifica la lista, per visualizzare il risultato ottenuto.

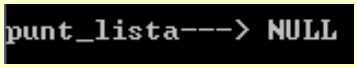
```
/* Il parametro in ingresso è il puntatore alla testa della lista */
void visualizza_lista(struct elemento *p)
{
printf("\npunt_lista---> ");
/* Ciclo di scansione della lista */
while(p!=NULL) {
printf("%d", p->inf); /* Visualizza il campo informazione */
printf("---> ");
p = p->pun; /* Scorri di un elemento in avanti */
}
printf("NULL\n\n");
}
```

Creiamo una lista con un solo elemento

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
}
```

Creiamo una lista con un solo elemento

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
}
```



Creiamo una lista con un solo elemento

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
}
```

Creiamo una lista con un solo elemento

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
}
```

Aggiungiamo un elemento in testa alla lista

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=7;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);}
```

Aggiungiamo un elemento in testa alla lista

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista; // Puntatore alla testa della lista
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=7;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);}
```

```
punt_lista---> NULL

punt_lista---> 5---> NULL

punt_lista---> 7---> NULL
```

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=7;
paux->pun=NULL;
visualizza_lista(punt_lista);}
```

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=7;
paux->pun=NULL;
visualizza_lista(punt_lista);}
```

Cosa stamperà?

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=7;
paux->pun=NULL;
visualizza_lista(punt_lista);}
```

```
punt_lista---> NULL

punt_lista---> 5---> NULL

punt_lista---> 5---> NULL
```

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=7;
paux->pun=NULL;
visualizza_lista(punt_lista); visualizza_lista(paux);}
```

Avevamo sbagliato il parametro della funzione visualizza_lista?

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=7;
paux->pun=NULL;
visualizza_lista(punt_lista); visualizza_lista(paux);}
```

Che cosa stamperà il monitor questa volta?

Aggiungiamo un elemento alla lista con paux

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=5;
paux->pun=NULL;
visualizza_lista(punt_lista); visualizza_lista(paux);}
```

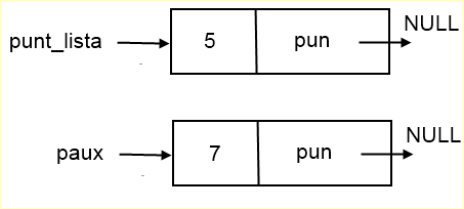
```
punt_lista---> NULL

punt_lista---> 5---> NULL

punt_lista---> 5---> NULL

punt_lista---> 7---> NULL
```

Aggiungiamo un elemento alla lista con paux

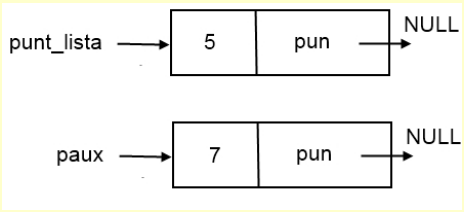
```
#include <stdio.h>
#include <malloc.h>
struct elemento {
int inf;
struct elemento *pun; };
void visualizza_lista(struct elemento *);
main()
{
struct elemento *punt_lista, *paux;
punt_lista = NULL; /* Lista vuota */
visualizza_lista(punt_lista); /* Chiamata funzione visualizza la lista */
/* Creazione del primo elemento */
punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
punt_lista->inf=5;
punt_lista->pun=NULL;
visualizza_lista(punt_lista);
paux = (struct elemento *)malloc(sizeof(struct elemento));
paux->inf=5;
paux->pun=NULL;
visualizza_lista(punt_lista); visualizza_lista(paux);}
```

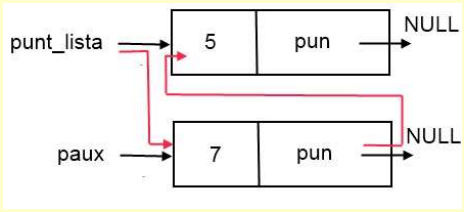
Che cosa non ha funzionato correttamente?

Situazione attuale



Cosa modificare





In pratica

- punt_lista = NULL; /* Lista vuota */
- visualizza_lista(punt_lista); /*visualizza la lista vuota*/
- /* Creazione del primo elemento */
- punt_lista = (struct elemento *)malloc(sizeof(struct elemento));
- punt_lista->inf=5;
- punt_lista->pun=NULL;
- visualizza_lista(punt_lista);//visualizza lista con 1 elemento
- //creiamo un altro elemento
- paux = (struct elemento *)malloc(sizeof(struct elemento));
- paux->inf=7;
- paux->pun=punt_lista;
- punt_lista=paux;
- visualizza_lista(punt_lista); // visualizza lista con 2 elementi

Risultato finale

```
punt_lista---> NULL

punt_lista---> 5---> NULL

punt_lista---> 7---> 5---> NULL
```

Finalmente è corretto

Gestione della lista con funzioni

- Il passo successivo è gestire la lista mediante delle funzioni.
- Iniziamo col creare una lista vuota. Servono due funzioni la cui dichiarazione è ad esempio:

  struct elemento *crea_lista_vuota();

  void visualizza_lista(struct elemento *);
- Nel main verrà definito il puntatore che conterrà il riferimento al primo elemento della lista:

  struct elemento *punt_lista;
- Le due funzioni vengono chiamate in sequenza dallo stesso main;
- punt_lista = crea_lista_vuota();
- visualizza_lista(punt_lista);

Gestione della lista con funzioni

- Il passo successivo è gestire la lista mediante delle funzioni.
- Iniziamo col creare una lista vuota. Servono due funzioni la cui dichiarazione è ad esempio:

  struct elemento *crea_lista_vuota();

  void visualizza_lista(struct elemento *);
- Nel main verrà definito il puntatore che conterrà il riferimento al primo elemento della lista:

  struct elemento *punt_lista;
- Le due funzioni vengono chiamate in sequenza dallo stesso main;
- punt_lista = crea_lista_vuota();
- visualizza_lista(punt_lista);

Gestione della lista con funzioni

- Il passo successivo è gestire la lista mediante delle funzioni.
- Iniziamo col creare una lista vuota. Servono due funzioni la cui dichiarazione è ad esempio:

  struct elemento *crea_lista_vuota();

  void visualizza_lista(struct elemento *);
- Nel main verrà definito il puntatore che conterrà il riferimento al primo elemento della lista:

  struct elemento *punt_lista;
- Le due funzioni vengono chiamate in sequenza dallo stesso main;
- punt_lista = crea_lista_vuota();
- visualizza_lista(punt_lista);

- La procedura crea_lista_vuota restituisce al main il puntatore alla lista che è assegnato a punt_lista e che viene successivamente passato a visualizza_lista.

Cominciamo con le Dichiarazioni

- /* Esercizio elementare. crea una una lista vuota e la stampa */

- #include <stdio.h>
- /* struttura degli elementi della lista */

- struct elemento {
- int inf;
- struct elemento *pun;
- };

- struct elemento *crea_lista_vuota();// funzione crea lista vuota
- void visualizza_lista(struct elemento *);// funzione che visualizza la lista

Main

- main()
- {
- struct elemento *punt_lista; /* dichiaro il puntatore alla testa della lista */
- punt_lista = crea_lista_vuota(); /* Chiamata funzione per creare la lista vuota */
- visualizza_lista(punt_lista); /* Chiamata funzione per
- visualizzare la lista */
- }

Funzione per la creazione della lista vuota

- /*Questa funzione Restituisce il puntatore alla testa */

- struct elemento *crea_lista_vuota()
- {
- struct elemento *p;
- p = NULL; /*la lista è vuota */
- return(p);
- }

Funzione per la visualizzazione della lista.

E' la solita funzione che va richiamata ogni volta che facciamo una operazione che modifica la lista, per visualizzare il risultato ottenuto.

```
/* Il parametro in ingresso è il puntatore alla testa della lista */
void visualizza_lista(struct elemento *p)
{
printf("\npunt_lista---> ");
/* Ciclo di scansione della lista */
while(p!=NULL) {
printf("%d", p->inf); /* Visualizza il campo informazione */
printf("---> ");
p = p->pun; /* Scorri di un elemento in avanti */
}
printf("NULL\n\n");
}
```

Funzione inserisci in testa

```
struct elemento *inserisci_in_testa(struct elemento *p)
{
struct elemento  *paus;
int i, n;

/* Creazione del nuovo elemento */
paus = (struct elemento *)malloc(sizeof(struct elemento));
printf("\nInserisci la 1 informazione: ");
scanf("%d", &paus->inf);
paus->pun = p;
p=paus;
return (p);}
```