





Facoltà di Scienze
Matematiche
Fisiche Naturali

Laboratorio di Algoritmi e Strutture Dati

Prof. Aniello Murano

Implementazioni di Liste Puntate Semplici

Corso di Laurea
Codice insegnamento
Email docente
Anno accademico

Informatica
13917
murano@na.infn.it
2007/2008

Lezione numero: 7
Parole chiave: **Liste dinamiche**

next





27/09/2007



Facoltà di Scienze
Matematiche
Fisiche Naturali

Indice

- **Liste puntate semplici:** Gli elementi sono organizzati in modo sequenziale e si possono scorrere in un unico verso. La lista ha un primo elemento (testa) e un ultimo elemento (coda)


- **Liste doppiamente puntate:** Simili alle liste puntate semplici, ma permettono di scorrere gli elementi in entrambi i versi


- **Liste puntate semplici circolari:** Sono liste puntate semplici senza testa ne coda.


- **Liste doppiamente puntate circolari:** Liste doppiamente puntate senza testa ne coda.



back
X
next

Federica 27/09/2007 3 Facoltà di Scienze Matematiche Fisiche Naturali

Torniamo al linguaggio C

Per l'implementazione delle liste in linguaggio C, possiamo utilizzare due importanti costrutti:

- STRUTTURE
- ALLOCAZIONE DINAMICA DELLA MEMORIA

back ✖ next

Federica 27/09/2007 4 Facoltà di Scienze Matematiche Fisiche Naturali

Strutture

Le strutture del C sono simili ai record del Pascal:

- sostanzialmente permettono un'aggregazione di variabili, molto simile a quella degli array, ma a differenza di questi non ordinata e non omogenea (una struttura può contenere variabili di tipo diverso).

Per denotare una struttura si usa la parola chiave struct seguita dal nome identificativo della struttura, che è opzionale.

- Nell'esempio sottostante si definisce una struttura "libro" e si crea un'istanza di essa chiamata "biblio":

```
struct libro{
    char titolo[100];
    char autore[50];
    int anno_publicazione;
    float prezzo;    };
struct libro biblio;
```

back ✖ next

Federica 27/09/2007 5 Facoltà di Scienze Matematiche Fisiche Naturali

Strutture (2)

- La variabile "biblio" può essere dichiarata anche mettendo il nome stesso dopo la parentesi graffa:


```
struct libro          {
    char titolo[100];
    char autore[50];
    int anno_pubblicazione;
    float prezzo;          } biblio;
```
- Inoltre, è possibile pre-inizializzare i valori, alla dichiarazione, mettendo i valori (giusti nel tipo) compresi tra parentesi graffe:
 - struct libro biblio = {"Guida al C", "Fabrizio Ciacchi", 2003, 45.2};
- Per accedere alle variabili interne della struttura si usa l'operatore "."
 - Esempio: Per assegnare alla variabile interna prezzo il valore 50 usiamo biblio.prezzo = 50;

back ✖ next

Federica 27/09/2007 6 Facoltà di Scienze Matematiche Fisiche Naturali

Nuovi tipi di dato

- **Per definire nuovi tipi di dato si utilizza la funzione typedef.**
- **Con typedef e l'uso di struct è possibile creare tipi di dato molto complessi, come mostrato nell'esempio seguente:**
 - typedef struct libro {


```
char titolo[100];
char autore[50];
int anno_pubblicazione;
float prezzo;          } t_libro;
```
 - Per creare una variabile "guida" di tipo "t_libro", usiamo:
 - t_libro guida={"Guida al C", "Fabrizio Ciacchi", 2003, 45.2};

back ✖ next

Federica 27/09/2007 7 Facoltà di Scienze Matematiche Fisiche Naturali

Nuovi tipi di dato

Come per ogni altro tipo di dato, anche con "t_libro" si possono creare degli array:

- t_libro raccolta[5000];

Nel caso di array, per accedere ad un variabile interna, si utilizza l'indice insieme all'operatore punto (.)

- Esempio: Per assegnare il prezzo 50 al libro con indice 10 usiamo `raccolta[10].prezzo = 50;`

back ✖ next

Federica 27/09/2007 8 Facoltà di Scienze Matematiche Fisiche Naturali

Puntatori e Strutture

Consideriamo il seguente esempio di uso congiunto di strutture e puntatori:

```
struct PIPPO { int x, y; } elemento;
struct PIPPO *puntatore;
puntatore = &elemento;
puntatore->x = 6;
puntatore->y = 8;
```

Abbiamo dunque creato una struttura di tipo PIPPO e di nome "elemento", ed un puntatore ad una struttura di tipo PIPPO.

Per accedere ai membri interni della struttura "elemento" abbiamo usato l'operatore -> sul puntatore alla struttura.

- In pratica, `puntatore->x = 6` semplifica `(*puntatore).x=6;`

back ✖ next

Federica 27/09/2007 9 Facoltà di Scienze Matematiche Fisiche Naturali

Allocazione dinamica della memoria

A differenza di altri linguaggi, all'occorrenza il C permette di assegnare la giusta quantità di memoria alle variabili del programma.

Le funzioni utilizzate per gestire dinamicamente la memoria delle variabili sono principalmente:

- **malloc()** e **calloc()**, adibite all'allocazione della memoria;
- **free()** che serve per liberare la memoria allocata,
- **realloc()**, che permette la modifica di uno spazio di memoria precedentemente allocato.
- Infine, un comando particolarmente utile è **sizeof**, che restituisce la dimensione del tipo di dato da allocare.

Queste funzioni sono incluse nella libreria malloc.h,

back X next

Federica 27/09/2007 10 Facoltà di Scienze Matematiche Fisiche Naturali

Esempio di allocazione dinamica

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
main()
{
    int numero=100, allocati, *array, i; char buffer[15];
    printf("Numero di elementi dell'array: %d", numero);
    array = (int *)malloc(sizeof(int) * numero);
    if(array == NULL) { printf("Memoria esaurita\n"); exit(1); }
    allocati = sizeof(int) * numero;
    for(i=0; i<numero; i++)    array[i] = i;
    printf("\n Valori degli elementi \n");
    for(i=0; i< numero; i++)    printf("%d", array[i]);
    printf("\n\n Numero elementi %d \n", numero);
    printf("Dimensione elemento %d \n", sizeof(int));
    printf("Bytes allocati %d \n", allocati);
    free(array);
    printf("\n Memoria Liberata \n");
}
```

back X next

Federica 27/09/2007 11 Facoltà di Scienze Matematiche Fisiche Naturali

Uso di realloc()

La sintassi della funzione `realloc()` ha due argomenti, il primo riguarda l'indirizzo di memoria, il secondo specifica la nuova dimensione del blocco;

Esempio di frammento di codice:

```
while(scanf("%d", &x))
{
    allocati += sizeof(int)
    array = (int *)realloc(array, allocati);
    if(array == NULL)
    {
        printf("Memoria insufficiente\n");
        exit(1);
    }
    i++;
    array[i] = x;
}
```

back X next

Federica 27/09/2007 12 Facoltà di Scienze Matematiche Fisiche Naturali

Rischi della gestione dinamica della memoria

Produzione di "garbage":

- quando la memoria allocata dinamicamente resta logicamente inaccessibile, perché si sono persi i riferimenti:
- Esempio:
 - **P=malloc(sizeof(TipoDato));**
 - **P=Q;**

Riferimenti "dangling"(fluttuanti):

- quando si creano riferimenti a zone di memoria logicamente inesistenti
- Esempio:
 - **P=Q; free(Q);**
- l'istruzione **free** libera l'area di memoria ma non provoca un assegnamento automatico di **NULL** al puntatore Q, per cui P e Q si riferiscono perciò a celle di memoria non più esistenti

back X next

Federica 27/09/2007 13 Facoltà di Scienze Matematiche Fisiche Naturali

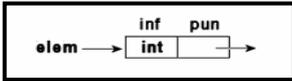
Liste puntate

Una lista è una collezione di elementi omogenei

A differenza dell'array, la dimensione di una lista non è nota a priori e può variare nel tempo. Inoltre un elemento nella lista occupa una posizione qualsiasi, che tra l'altro può cambiare dinamicamente durante l'utilizzo della lista stessa.

Ogni elemento nella lista ha uno o più campi contenenti informazioni, e, necessariamente, deve contenere un puntatore per mezzo del quale è legato all'elemento successivo

Esempio:



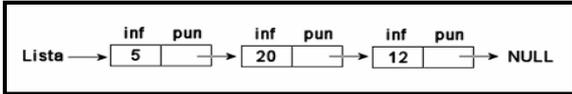
back X next

Federica 27/09/2007 14 Facoltà di Scienze Matematiche Fisiche Naturali

Liste puntate

Una lista puntata (semplice) ha una gestione sequenziale, in cui è sempre possibile individuare la testa e la coda della lista.

Esempio:



back X next

Federica 27/09/2007 15 Facoltà di Scienze Matematiche Fisiche Naturali

Operazioni sulle liste

Le operazioni che agiscono su un a lista rappresentano gli operatori elementari che agiscono sulle variabili di tipo lista

Corrispondono a dei sottoprogrammi (funzioni)

Alcune operazioni modificano la lista

Operazioni tipiche:

- **Inizializzazione** - modifica la lista
- **Inserimento in testa** - modifica la lista
- **Inserimento in coda** - modifica la lista
- **Inserimento all'interno** - modifica la lista
- **Verifica lista vuota** - non modifica la lista
- **Ricerca elemento** - non modifica la lista
- **Stampa lista** - non modifica la lista

back X next

Federica 27/09/2007 16 Facoltà di Scienze Matematiche Fisiche Naturali

Verifica Lista vuota

Data una lista L, per verificare se essa è vuota è sufficiente controllare se essa punta a NULL.

La seguente verifica se L è vuota

```
void controlla_lista_vuota(struct elemento *L)
{
    return (L==NULL);
}
```

back X next

Federica 27/09/2007 17 Facoltà di Scienze Matematiche Fisiche Naturali

Inizializzazione

Consideriamo un semplice programma per l'inizializzazione di una lista di interi.

```
#include <stdio.h>
#include <malloc.h>
struct elemento {
    int inf;
    struct elemento *next; }
int main()
{
    struct elemento *lista; /*puntatore della lista */
    lista = crea_lista(); /* crea la lista */
    visualizza_lista(lista); /* stampa la lista */
}
```

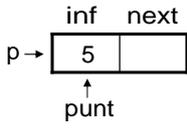
back X next

Federica 27/09/2007 18 Facoltà di Scienze Matematiche Fisiche Naturali

Funzione crea_lista() 1/2

La funzione `crea_lista()` crea due puntatori ad elemento, uno di nome `p` (punta al primo elemento) e l'altro di nome `punt` (permette di scorrere la lista);

```
struct elemento *crea_lista()
{
    struct elemento *p, *punt;
    int i, n;
    printf("\n Specificare il numero di elementi... ");
    scanf("%d", &n);
    if(n==0)
        p = NULL;
    else {
        /* creazione primo elemento */
        p = (struct elemento *)malloc(sizeof(struct elemento));
        printf("\nInserisci il primo valore: ");
        scanf("%d", &p->inf);
        punt = p;
        .....
```



back X next

Federica 27/09/2007 19

Facoltà di Scienze
Matematiche
Fisiche Naturali

Funzione crea_lista() 2/2

```

for(i=2; i<=n; i++)
{
    punt->next = (struct elemento *)malloc(sizeof(struct elemento));
    punt = punt->next;
    printf("\nInserisci il %d elemento: ", i);
    scanf("%d", &punt->inf);
} /* chiudo il for */
punt->next = NULL; /* marcatore fine lista */
} /* chiudo l'if-else */
return(p);
} /* chiudo la funzione */

```

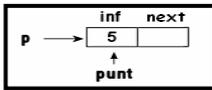
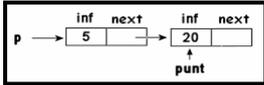
Questo tipo di inserimento viene chiamata "inserimento in coda"

back X next

Federica 27/09/2007 20

Facoltà di Scienze
Matematiche
Fisiche Naturali

Esempio di funzionamento di crea_lista

- Assumiamo che si voglia creare una lista di 3 elementi (5,20,12); Alla prima iterazione abbiamo la seguente situazione:
 
- Supponiamo adesso di aver inserito i primi due elementi e stiamo per inserire il terzo. La lista avrà la seguente forma:
 

back X next

Federica 27/09/2007 21 Facoltà di Scienze Matematiche Fisiche Naturali

Esempio di funzionamento di crea_lista

- A questo punto inserendo il valore 12 per prima cosa viene creato un altro oggetto della lista, identificato con punt -> next,
- poi "punt", il puntatore ausiliario, viene fatto puntare, non più al secondo elemento, bensì al terzo, all'atto pratico "punt" diventa il puntatore dell'oggetto da lui puntato (cioè, punt = punt -> next;).
- Quindi viene inserito il campo informazione dell'elemento tramite l'input da tastiera dell'utente; in questo caso viene inserito il valore 12;
- Alla fine, punt punta al valore NULL che identifica la fine della lista.

back X next

Federica 27/09/2007 22 Facoltà di Scienze Matematiche Fisiche Naturali

Funzione visualizza_lista()

La seguente funzione iterativa permette di stampare tutti gli elementi interi presenti in una lista, nell'ordine in cui sono memorizzati

```
void visualizza_lista(struct elemento *p)
{
    printf("\n lista ---> ");
    while(p != NULL)
    {
        printf("%d", p->inf); /* visualizza l'informazione */
        printf(" ---> ");
        p = p->next; /* scorre la lista di un elemento */
    }
    printf("NULL\n\n");
}
```

back X next

Federica 27/09/2007 23 Facoltà di Scienze Matematiche Fisiche Naturali

Ricorsione su liste (1)

La ricorsione risulta particolarmente utile sulle liste collegate. Questo è dovuto al fatto che le liste si possono definire in modo ricorsivo:

Una lista è la lista vuota, oppure un elemento seguito da un'altra lista.

In altre parole, una variabile di tipo lista L può valere NULL (che rappresenta la lista vuota), oppure può essere un puntatore a una struttura che contiene un dato più un altro puntatore. Possiamo quindi dire che la struttura è composta da un elemento e da un puntatore, che rappresenta un'altra lista.

back X next

Federica 27/09/2007 24 Facoltà di Scienze Matematiche Fisiche Naturali

Ricorsione su liste (2)

La lista si ottiene guardando la struttura puntata e poi seguendo i puntatori fino a NULL.

Sia L una lista definita da

```
struct lista {int val; struct lista *next;} L;
```

Una funzione ricorsiva su L avrà come argomento L, e al suo interno una chiamata ricorsiva a cui si passa L→next.

Queste funzioni normalmente operano su L→val (il primo elemento della lista), e poi agiscono sul resto della lista solo attraverso la chiamata ricorsiva.

back X next

Federica 27/09/2007 25 Facoltà di Scienze Matematiche Fisiche Naturali

Funzione visualizza_lista() ricorsiva

```
void visualizza_lista(struct elemento *p)
{
    if(p==NULL) return;
    printf("%d ", p->inf);
    visualizza_lista(p->next);
}
```

Se p rappresenta la lista vuota, non si stampa niente; si esce semplicemente dalla funzione senza fare nulla.

Al passo i-esimo, si stampa la testa della lista e si richiama la funzione visualizza_lista sulla lista meno la testa.

back X next

Federica 27/09/2007 26 Facoltà di Scienze Matematiche Fisiche Naturali

Esercizio

Sia L una lista definita da

```
struct lista {int val; struct lista *next;} L;
```

Scrivere in linguaggio C una funzione ricorsiva che preso in input L, raddoppi tutti gli elementi dispari della lista

back X next

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.