RIPASSO C
Parte 5

(Slides cortesemente concesse in uso dal Prof. Fraschini)

Oggetti dinamici

Liste lineari concatenate

Oggetti dinamici

- ✓I **puntatori** possono essere usati per la creazione e la gestione di **oggetti dinamici**
- ✓Gli oggetti dinamici sono creati durante l'esecuzione del programma
- ✓Non hanno un nome esplicito (si usano i puntatori)
- ✓La memoria viene allocata attraverso l'utilizzo della funzione **malloc** che restituisce un puntatore

p = (int *) malloc (sizeof(int)); //stdlib.h

viene allocata la memoria necessaria per accogliere un intero (allocato dinamicamente) raggiungibile utilizzando il puntatore p

Oggetti dinamici

p = (int *) malloc (sizeof(int)); //stdlib.h

viene allocata la memoria necessaria per accogliere un intero (allocato dinamicamente) raggiungibile utilizzando il puntatore p

p = (int *) malloc (sizeof(int));

p ->

***p=10;**

q=p;

p ->
q ->

Per deallocare lo spazio della memoria si usa la funzione **free**

free(p); //libera lo spazio occupato dall'intero puntato da p!

Ad ora…

- Array.
  - Occupazione memoria (sovrastima)
  - Velocità (inserimento/eliminazione e spostamento)

Liste lineari

- Lista di elementi omogenei memorizzati in una posizione qualsiasi
- Ogni **elemento** è costituito da un'**informazione** e da un **puntatore** (che permette di concatenare gli elementi)
- Si accede alla lista per mezzo di un **puntatore al primo elemento**
- Il puntatore dell'ultimo elemento punta a **NULL**
- Lista vuota: puntatore a NULL

Liste lineari

```
6   struct elemento
7   {
8       int val;
9       struct elemento *next;
10  };
11
12  struct elemento *punt_lista;
```

Esempio. Crea e visualizza lista di N interi

```
10  struct elemento *crealista();
11  void visualizzalista(struct elemento *);
12
13  int main()
14  {
15      struct elemento *punt_lista;
16
17      punt_lista=crealista();
18      visualizzalista(punt_lista);
19      return 0;
20  }
21
```

```c
22  struct elemento *crealista()
23  {
24      struct elemento *p, *paux;
25      int i,n;
26
27      printf("Quanti elementi vuoi inserire?\n");
28      scanf("%d",&n);
29      if(n==0) p=NULL;
30      else
31      {
32          //primo elemento
33          p=(struct elemento *) malloc(sizeof(struct elemento));
34          printf("Inserisci valore del 1° elemento: \n");
35          scanf("%d",&p->val);
36          paux=p;
```

- **p** puntatore alla testa della lista
- **paux** puntatore ausiliario
  - permette di creare gli elementi successivi senza perdere il riferimento al primo elemento

- **p** puntatore alla testa della lista
- **paux** puntatore ausiliario
  - permette di creare gli elementi successivi senza perdere il riferimento al primo elemento

```
        //elementi successivi
        for(i=2;i<=n;++i)
        {
            paux->next=(struct elemento *) malloc(sizeof(struct elemento));
            paux=paux->next;
            printf("Inserisci valore del %d° elemento: \n",i);
            scanf("%d",&paux->val);
        }
        paux->next=NULL;
    }
    return p;
}
```

```
51  void visualizzalista(struct elemento *p)
52  {
53      while(p!=NULL)
54      {
55          printf("%d\n",p->val);
56          p=p->next;
57      }
58  }
59
```

Oggetti dinamici

Liste lineari concatenate – Parte 2

Esempio.
Crea lista con uscita (-1)

```c
struct elemento *crealista()
{
    struct elemento *p, *paux;
    int x;

    printf("Inserisci valore elemento (-1 per uscire!): \n");
    scanf("%d",&x);
    if(x==-1) p=NULL;
    else
    {
        p=(struct elemento *) malloc(sizeof(struct elemento));
        p->val=x;
        paux=p;
        while(x!=-1)
        {
            printf("Inserisci valore elemento (-1 per uscire!): \n");
            scanf("%d",&x);
            if(x!=-1)
            {
                paux->next=(struct elemento *) malloc(sizeof(struct elemento));
                paux=paux->next;
                paux->val=x;
            }
            else
                paux->next=NULL;
        }
    }
    return p;
}
```

Esempio. Crea lista come somma di due altre liste

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

struct elemento
{
    int val;
    struct elemento *next;
};

struct elemento *crealista();
void visualizzalista(struct elemento *);
struct elemento *sommaliste(struct elemento *, struct elemento *);


int main()
{
    struct elemento *punt_lista1,*punt_lista2,*punt_lista3;

    printf("Inserisci prima lista\n");
    punt_lista1=crealista();
    printf("Inserisci seconda lista\n");
    punt_lista2=crealista();
    printf("Elementi prima lista\n");
    visualizzalista(punt_lista1);
    printf("Elementi seconda lista\n");
    visualizzalista(punt_lista2);
    punt_lista3=sommaliste(punt_lista1,punt_lista2);
    printf("Elementi lista somma: \n");
    visualizzalista(punt_lista3);

    return 0;
}
```

```c
struct elemento *sommaliste(struct elemento *l1, struct elemento *l2)
{
    struct elemento *l3,*paux;

    l3=NULL;
    if(l1!=NULL && l2!=NULL)
    {
        l3=(struct elemento *) malloc(sizeof(struct elemento));
        l3->val=l1->val+l2->val;
        paux=l3;
        l1=l1->next;
        l2=l2->next;
        for(;l1!=NULL && l2!=NULL;)
        {
            paux->next=(struct elemento *) malloc(sizeof(struct elemento));
            paux=paux->next;
            paux->val=l1->val+l2->val;
            l1=l1->next;
            l2=l2->next;
        }
        paux->next=NULL;
    }
    return l3;
}
```

Elimina elemento da lista

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

struct elemento
{
    int val;
    struct elemento *next;
};

struct elemento *inserisci_ordine(struct elemento *);
struct elemento *elimina(struct elemento *);
void visualizzalista(struct elemento *);

```

```
27          switch(scelta)
28          {
29              case 1:
30                  punt_lista=inserisci_ordine(punt_lista);
31                  break;
32              case 2:
33                  punt_lista=elimina(punt_lista);
34                  break;
35              case 3:
36                  printf("La tua lista: \n");
37                  visualizzalista(punt_lista);
38                  break;
39              case 0:
40                  printf("Sei uscito dal programma!\n");
41                  break;
42              default:
43                  printf("Inserisci una scelta valida!\n");
44                  break;
45          }
```

```
82  struct elemento *elimina(struct elemento *p)
83  {
84      struct elemento *p1=p,*p2;
85      int el;
86      int trovato=0;
87
88      printf("Inserisci elemento da eliminare: \n");
89      scanf("%d",&el);
90      if(p1!=NULL)
91      {
92          if(p1->val == el)
93          {
94              p2=p1;
95              p=p->next;
96              free(p2);
97              return p;
98          }
99          else
```

```
99          else
100         {
101             while(p1->next!=NULL && trovato!=1)
102             {
103                 if(p1->next->val != el) p1=p1->next;
104                 else
105                 {
106                     trovato=1;
107                     p2=p1->next;
108                     p1->next=p1->next->next;
109                     free(p2);
110                     return p;
111                 }
112             }
113         }
114         if(!trovato) printf("Elemento non presente nella lista!\n");
115     }
116     else printf("La lista e' vuota!\n Nessun elemento da eliminare!\n\n");
117     return p;
118 }
```