

Informatica 1

Corso di Laurea Triennale in Matematica

Gianluca Rossi

`gianluca.rossi@uniroma2.it`

Dipartimento di Matematica
Università di Roma "Tor Vergata"

10: Strutture e liste - Parte seconda

Liste: Esempio “dinamico”

```
#include <stdio.h>
#include <stdlib.h>
#define n 20
```

```
struct listelem{
    double info;
    struct listelem *next;
};
```

```
int printList(struct listelem
*l){
    while( l != NULL){
        printf("%lf\n", (*l).info);
        l = (*l).next;
    }
}
```

```
int main(){
    struct listelem *L; /*rif. al primo elemento della lista*/
    struct listelem *t;
    double x;
    int i;
    L = NULL; /* la lista vuota */
    /* aggiungiamo alla lista i caratteri inseriti
       da tastiera*/
    for(i = 1; i < n; i++){
        if (scanf("%lf", &x) == 1){
            t = L;
            L = (struct listelem*)malloc(sizeof(struct listelem));
            (*L).info = x;
            (*L).next = t;
        }
    }
    printList(L);
}
```

L'operatore ->

Se p è un puntatore ad una struttura contenente il campo `nomecampo` allora con

`p->nomecampo;`

si accede al campo `nomecampo` della struttura puntata da p .
Equivalente a

`(*p).nomecampo;`

Il costrutto **typedef**

Permette di definire tipi derivati da altri tipi.

```
typedef tipo nuovo-nome;
```

Crea il tipo `nuovo-nome` da usare al posto di `tipo`.

```
typedef char boolean;  
typedef struct listelem listel;  
boolean b = 0;  
listel el;
```

Liste: Esempio “dinamico”

```
#include <stdio.h>
#include <stdlib.h>
#define n 20
```

```
struct listelem{
    double info;
    struct listelem *next;
};

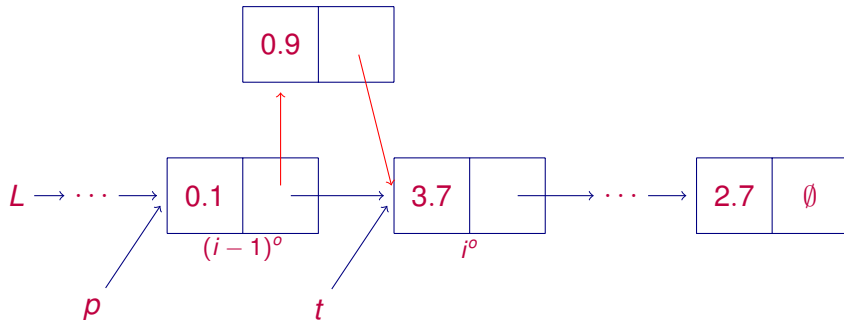
typedef struct listelem listel;
```

```
int printList(listel *l){
    while( l != NULL){
        printf("%lf\n", l->info);
        l = l->next;
    }
}
```

```
int main(){
    listel *L, *t;
    double x;
    int i;
    L = NULL;
    for(i = 1; i < n; i++){
        if (scanf("%lf", &x) == 1){
            t = L;
            L = (listel*)malloc(sizeof(listel));
            L->info = x;
            L->next = t;
        }
    }
    printList(L);
}
```

Liste: L'operatore **insert**

insert(L, 0.9, i);



Liste: L'operatore insert

```
#include <stdio.h>
#include <stdlib.h>
#define N 20

struct listelem{
    double info;
    struct listelem *next;
};

typedef struct listelem listel;

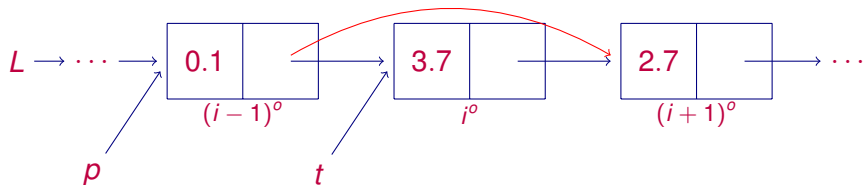
listel *insert(listel*, double, int);
int printList(listel*);

int main(){
    double x; int i;
    listel *L = NULL, *t;
    for(i = 1; i < N; i++){
        if (scanf( "%lf", &x) == 1)
            L = insert(L, x, 0);
        L = insert(L, 200, 2);
    }
    printList(L);
}
```

```
listel *insert(listel *L, double e, int n){
    listel *t = L, *p = L;
    int i;
    if(n==0){
        L = (listel*)malloc(sizeof(listel));
        L->info = e; L->next = t;
        return L;
    }
    for(i=0; i < n-1; i++){
        if(p == NULL)/*n troppo grande*/
            return L;
        p = p->next;
    }
    if(p == NULL)/*n troppo grande*/
        return L;
    t = p->next;
    p->next = (listel*)malloc(sizeof(listel));
    p->next->info = e; p->next->next = t;
    return L;
}
```

Liste: L'operatore **delete**

delete(L, i);



Liste: L'operatore delete

```
#include <stdio.h>
#include <stdlib.h>
#define N 20

struct listelem{
    double info;
    struct listelem *next;
};

typedef struct listelem listel;

listel *delete(listel*, int);
listel *insert(listel*, double, int);
int printList(listel *);

int main(){
    double x; int i;
    listel *L = NULL, *t;
    for(i = 1; i < N; i++)
        if (scanf("%lf", &x) == 1)
            L = insert(L, x, 0);
    L = delete(L, 0); printList(L);
}
```

```
listel *delete(listel *L, int n){
    listel *t = L, *p = L; int i;
    if(n==0){
        if (L != NULL){/*L non vuota*/
            L = L->next;
            free(t);
        }
        return L;
    }
    for(i=0; i < n-1; i++){
        if(p == NULL)
            return L;
        p = p->next;
    }
    t = p->next;
    if(t != NULL){/* p non e' l'ultimo */
        p->next = t->next;
        free(t);
    }
    return L;
}
```