

Lezione 6 e 7



Appunti di Davide Scarlata 2024/2025



Prof: Michele Garetto



Mail: michele.garetto@unito.it




Corso:  C



[Moodle corso C](#)



[Moodle Lab matricole dispari](#)

** Data 17/03/2025



Funzioni ricorsive su liste



Esempio della somma:

```
int somma_nodi(lista L) {  
    if (L == NULL) {  
        return 0;  
    }  
    return L->dato + somma_nodi(L->next);  
}
```



Rimozione ricorsiva di una lista

Rimuovere una lista significa non solo togliere degli elementi da essa ma anche liberare la memoria da essi occupata tramite `free`.



Rimozione in coda con ricorsione in testa



Versione V1 con lista passata per valore

```
lista elimina_lista1(lista L) {  
    if (L) { // se la lista non è vuota  
        L->next = elimina_lista1(L->next);  
        free(L); // libero il nodo (attuale ultimo)  
    }  
}
```

```
    return NULL; // segno che dopo non viene più nulla
}
```

● Versione V2 con lista passata per riferimento

```
void elimina_lista2(lista *Lptr) {
    if (Lptr && *Lptr) { // se il parametro è definito e la lista non è vuota
        elimina_lista2(&((*Lptr)->next)); // mi sposto avanti
        free(*Lptr); // libero il nodo (attuale ultimo)
        *Lptr = NULL; // segno che dopo non viene più nulla
    }
}
```



Inserzione di un nodo in lista ordinata

● Con passaggio per valore

```
lista ins_ord1(lista L, lista nodo) {
    if (L) { // se la lista esiste
        if (nodo->dato <= L->dato) { // nodo va inserito in prima posizione
            nodo->next = L;
            return nodo;
        }
        else if (!L->next) { // nodo diventa ultimo
            L->next = nodo;
            nodo->next = NULL;
            return L;
        }
        else if (nodo->dato < L->next->dato) { // nodo si inserisce tra due
nodi
            nodo->next = L->next;
            L->next = nodo;
            return L;
        }
        else { // ricorsione per trovare la posizione corretta
            L->next = ins_ord1(L->next, nodo);
            return L;
        }
    }
    return nodo; // se L era NULL, nodo diventa il primo e unico elemento
}
```

● Con passaggio per riferimento

```
void ins_ord2(lista *Lptr, lista nodo) {
    if (Lptr) {
        if (*Lptr) {
            if (nodo->dato <= (*Lptr)->dato) { // nodo in prima posizione
                nodo->next = *Lptr;
                *Lptr = nodo;
            }
            else if (!((*Lptr)->next)) { // nodo diventa ultimo
                (*Lptr)->next = nodo;
                nodo->next = NULL;
            }
            else if (nodo->dato < (*Lptr)->next->dato) { // nodo tra due nodi
                nodo->next = (*Lptr)->next;
                (*Lptr)->next = nodo;
            }
            else { // ricorsione
                ins_ord2(&((*Lptr)->next), nodo);
            }
        }
        else {
            *Lptr = nodo; // lista era NULL, nodo diventa il primo
        }
    }
}
```



Stampa ricorsiva di una lista

● Ricorsione in coda

```
void stampa(lista L) {
    if (L != NULL) {
        printf("%d,", L->dato); // elaboro nodo
        stampa(L->next); // ricorsione sul next
    }
    else {
        printf("\n"); // caso base: lista vuota
    }
}
```

● Ricorsione in testa

```
void stampa(lista L) {  
    if (L != NULL) {  
        stampa(L->next); // ricorsione sul next  
        printf(" %d,", L->dato); // elaboro nodo  
    }  
    else {  
        printf("\n"); // caso base: lista vuota  
    }  
}
```

✗ Errori tipici

- ⚠ **Lista passata per valore:** dimenticare di verificare se `L == NULL`.
- ⚠ **Lista passata per riferimento:** dimenticare la doppia verifica `if (Lptr) if (*Lptr)`.
- ⚠ **Saltare un caso** nella gestione della ricorsione.
- ⚠ **Non eseguire operazioni nel giusto ordine**, causando errori di memoria.
- ⚠ **Dimenticare di assegnare NULL** al `next` dell'ultimo elemento in coda.
- ⚠ **Assegnare NULL al next di un elemento** da inserire tra due nodi, interrompendo la lista.