INFORMATICA (PER AEROSPAZIALI)

A.A. 2018-19

Laboratorio n°5

Dott. Michele Zanella

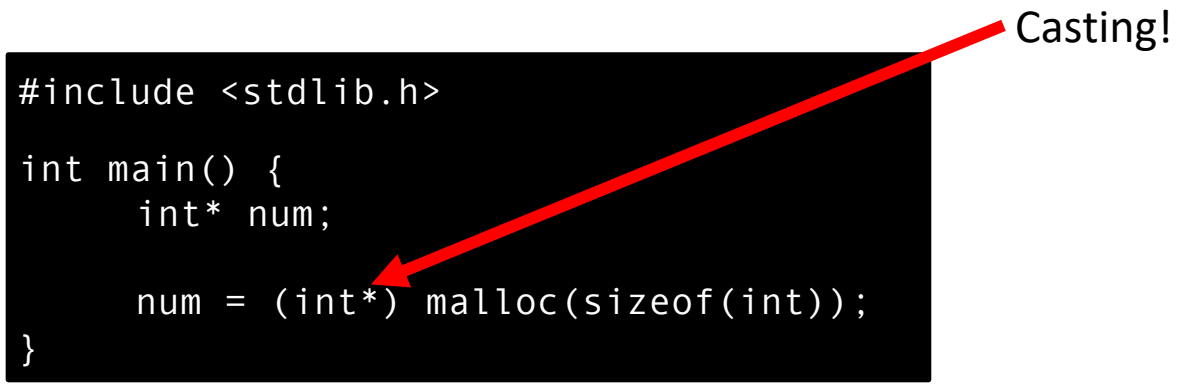
HEAP LAB

Allocazione dinamica

- La memoria è divisa sostanzialmente in due parti:
  - STACK: contiene le variabili locali delle funzioni, gli argomenti passati e viene allocata quando viene eseguita una chiamata a funzione
  - HEAP: contiene le variabili allocate durante l'esecuzione del programma ("runtime"), la cui dimensione non è nota a priori.

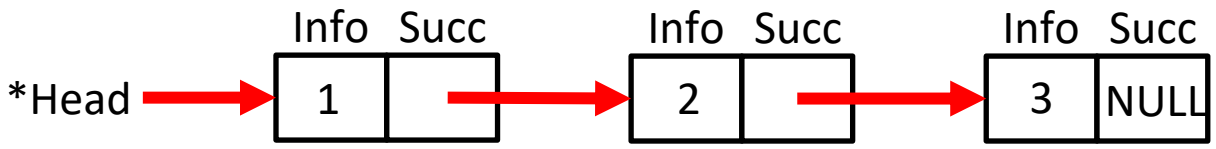
Allocazione dinamica

- Funzioni utili, bisogna includere *stdlib.h*:
  - `malloc()`: permette di allocare sullo heap blocchi di Byte in numero e di dimensione a priori (i.e., in fase di compilazione) non noto.
  - `free()`: libera la memoria allocata
  - `realloc()`: modifica uno spazio di memoria precedentemente allocato



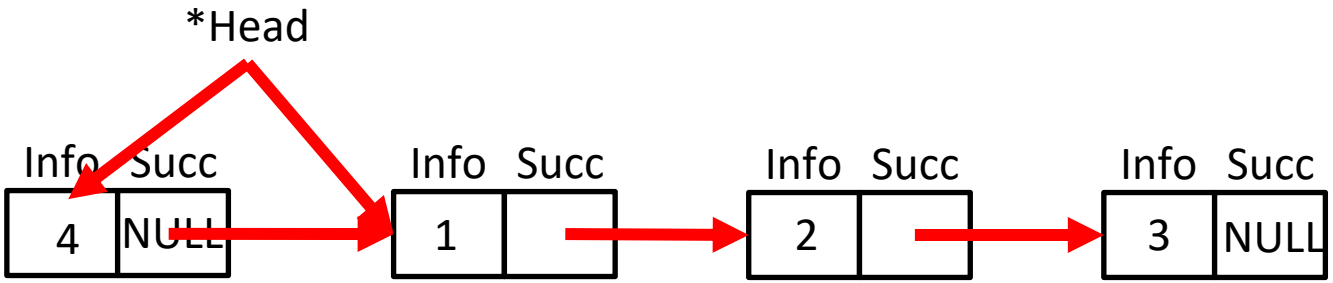
Liste

- Una **lista** è una collezione di elementi omogenei MA che occupano in memoria una posizione qualsiasi.
- La sua dimensione può cambiare dinamicamente durante l'esecuzione.
- Gli elementi della lista possono contenere vari campi per informazioni (come le *struct* normali), ma devono contenere un *puntatore all'elemento successivo* (in alcuni casi anche a quello precedente).



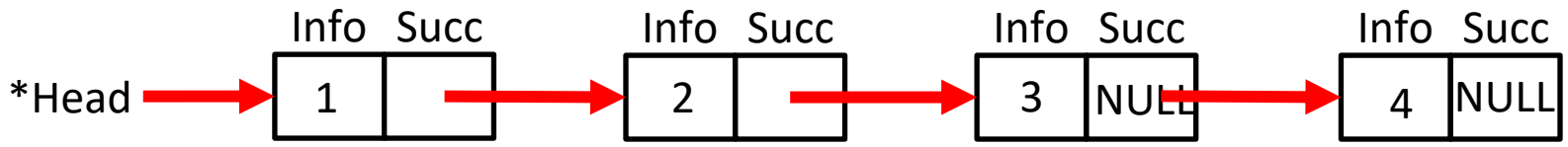
Liste: Aggiunta elementi

- Aggiungere un elemento in testa alla lista:
  1. Allocare la memoria per quell'elemento
  2. Sostituire NULL con l'indirizzo dell'attuale *head*
  3. Aggiornare la *head* con l'indirizzo dell'elemento creato al punto 1



Liste : Aggiunta elementi

- Aggiungere un elemento in fondo alla lista:
  1. Allocare la memoria per quell'elemento
  2. Percorrere la lista dalla corrente *head* fino a quando non si raggiunge il campo *succ* dell'ultimo elemento (che sarà NULL)
  3. Sostituire NULL con l'indirizzo dell'elemento creato al punto 1
  4. Assegnare NULL al campo *succ* di questo ultimo elemento



Esercizio 5.1: Array dinamico

Si scriva un programma che permetta di inserire dei numeri interi in un array di dimensione decisa dall'utente

Hint: Allocare dinamicamente la memoria dell'array una volta saputa la dimensione dall'utente

Esercizio 5.2: Runtime Array

Si scriva un programma che permetta di inserire dei numeri interi in un array di dimensione calcolata runtime.

In questo caso l'utente può continuare ad aggiungere numeri ad ogni iterazione.

Hint: Reallocare la memoria dinamicamente ad ogni iterazione.

Esercizio 5.3 : Rubrica dinamica (da Es. 3.3)

Si scriva un programma che implementi una semplice rubrica del telefono.

Requisiti:

- Possibilità di inserire/cancellare contatti
- Possibilità di stampare elenco contatti con informazioni
- Possibilità di ricercare un contatto dato il cognome

Hints:

- Riutilizzare eventuali funzioni contenuti in librerie (es. `string.h`)
- Settare un numero massimo `CONT_MAX` di contatti

Esercizio 5.3: Rubrica dinamica (da Es. 3.3) (cont'd)

Contatto:

Definiamo un tipo di dato personalizzato *contatto*. Il tipo deve includere le informazioni relative a:

- Cognome
- Nome
- Età
- Numero di telefono

…e fin qui nulla di nuovo…

Esercizio 5.3: Rubrica dinamica (da Es. 3.3) (cont'd)

MA poiché il numero di contatti non e' noto, va usata una struttura dinamica (lista monodirezionale)

Il programma (dotato di menu minimale) deve permettere:

1) inserimento in testa
2) inserimento in coda
3) stampa di tutti i contatti dall'inizio della lista
4) cancellazione di tutti i contatti
5) ricerca di un contatto dato il cognome
6) (aggiuntivo) stampa di tutti i contatti dalla fine della lista (modificare opportunamente gli elementi della lista

Hint:

- Fare un menu che legga l'intero dell'utente corrispondente alla funzione scelta

Esercizio 5.3: Rubrica dinamica (da Es. 3.3) (cont'd)

Le funzioni da implementare saranno (dati i prototipi):

- Inserimento all'inizio e alla fine

```
Contatto_t* inserisci_inizio(Contatto_t *head);
```

```
Contatto_t* inserisci_fine(Contatto_t *head);
```

- Inserimento info contatto

```
void inserisci_contatto(Contatto_t *c);
```

- Stampa contatti (e singolo contatto)

```
void stampa_contatto(Contatto_t *c);
```

```
void stampa_contatti(Contatto_t *head);
```

- Cancella tutta la lista

```
Contatto_t* cancella_lista(Contatto_t *head);
```

- Trova contatto

```
void trova_contatto(Contatto_t *head);
```

- Stampa dalla fine

```
void stampa_da_fine(Contatto_t *head);
```