Strutture dinamiche in C

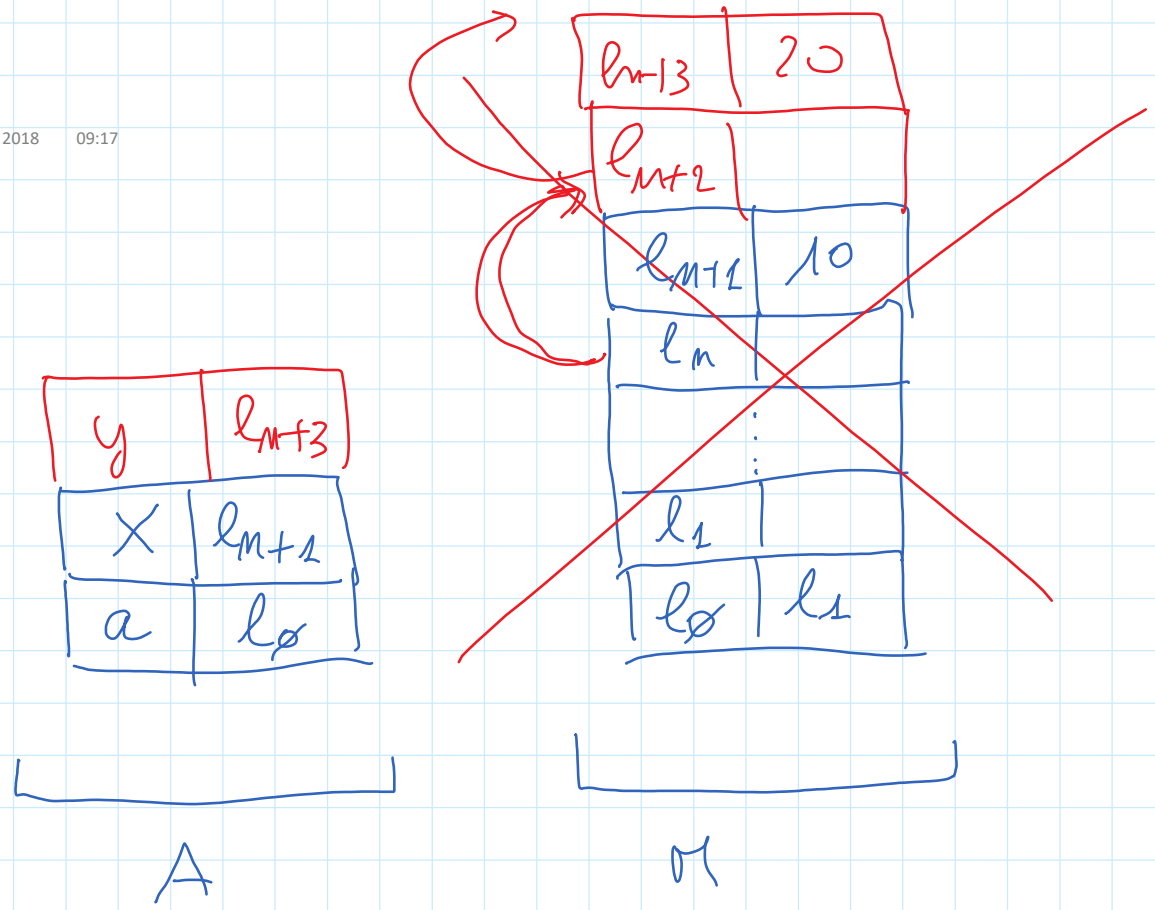
lunedì 5 novembre 2018 09:11

Strutture dinamiche:
una struttura dati (come gli array) che possono modificare la loro dimensione durante l'esecuzione dei programmi.

Array sono strutture STATICHE

Liste: strutture dati omogenee (tutti gli elementi che la compongono hanno lo stesso tipo) dinamica (posso aggiungere e togliere elementi).

lunedì 5 novembre 2018 09:17



Le strutture dinamiche non possono essere (facilmente) gestite nella memoria a pila.

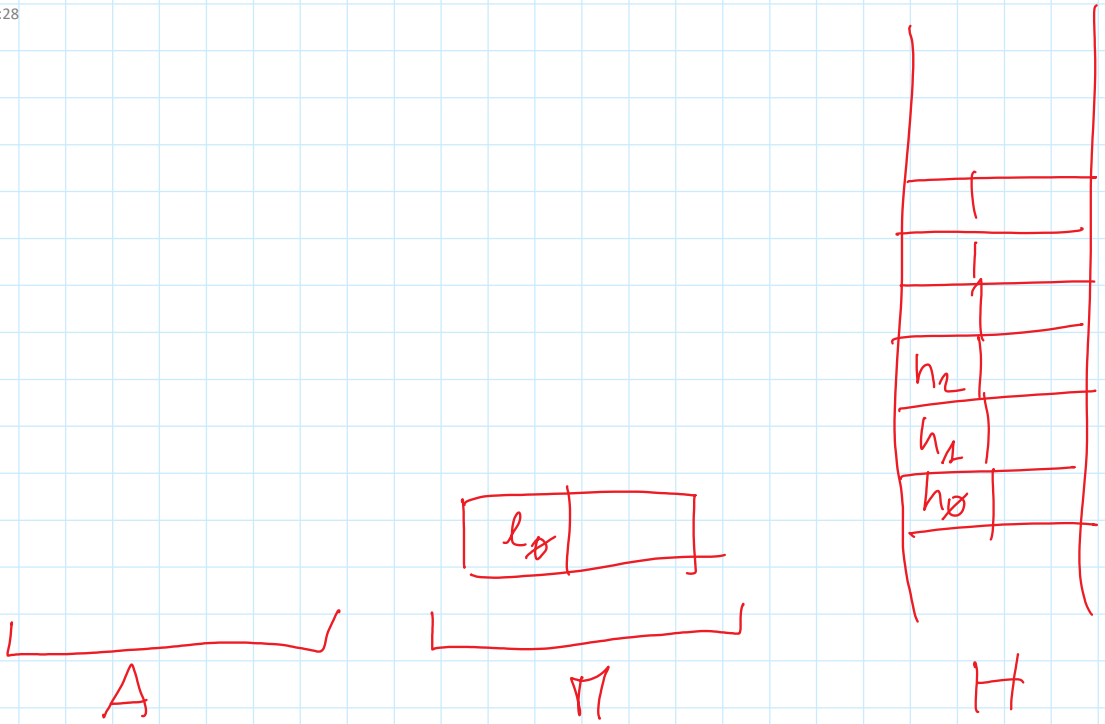
Le strutture dinamiche si gestiscono su una memoria particolare, la memoria dinamica (HEAP).

Per capire come vengono trattate le strutture dinamiche dobbiamo aggiungere allo stato astratto (ambiente e memoria) la memoria dinamica (heap).

In concreto, una parte della memoria del calcolatore, a disposizione del programma che sta eseguendo, viene gestita come una pila e una parte come memoria dinamica.

La memoria dinamica non viene gestita dalla esecuzione dei BLOCCHI. I indipendente dai blocchi e viene allocata (riservata) e deallocata (liberata) in modo ESPLICITO (attraverso funzioni e procedure predefinite)

La memoria dinamica è gestita come una sequenza di parole di memoria senza strutture. (Non ci sono frame, né pile)



Allocazione esplicita

funzione

```
p = malloc( 10 )
```

memory allocation

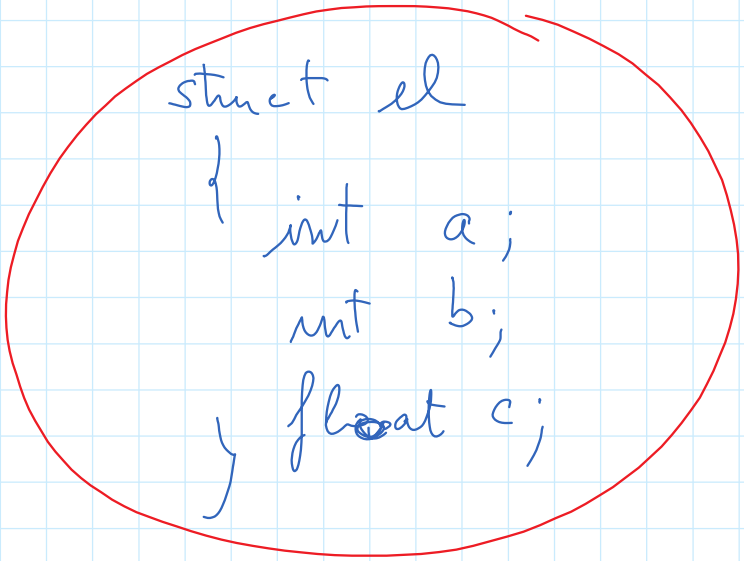
p: il primo indirizzo delle 10 parole riservate

10: lo spazio da allocare: 10 parole di memoria

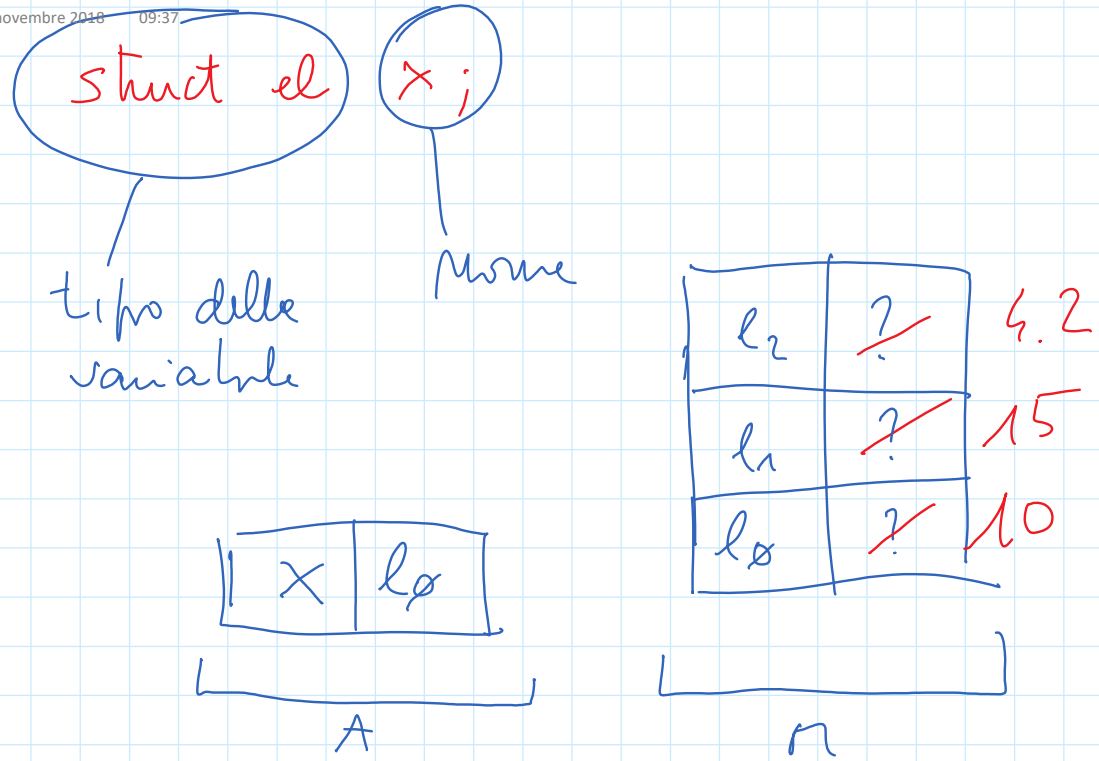
Se vogliamo allocare in memoria dinamica una struttura con un po' di elementi, dobbiamo sapere quanto spazio è necessario per quella struttura.

In C esiste un costruttore di tipo (definisce nuovi tipi a partire da tipi conosciuti) che si chiama STRUCT

struct permette oggetto di tipo diverso e costruisce un nuovo tipo



struct el ed è composto da 3 valori, 2 interi e uno float che si possono identificare attraverso i nomi a, b e c.



x.a = 10;

il campo (o elemento) di nome a delle strutture associate a x.

x.b = 15;

x.c = 4.2;

struct el x;

Creare, in memoria dinamica, lo

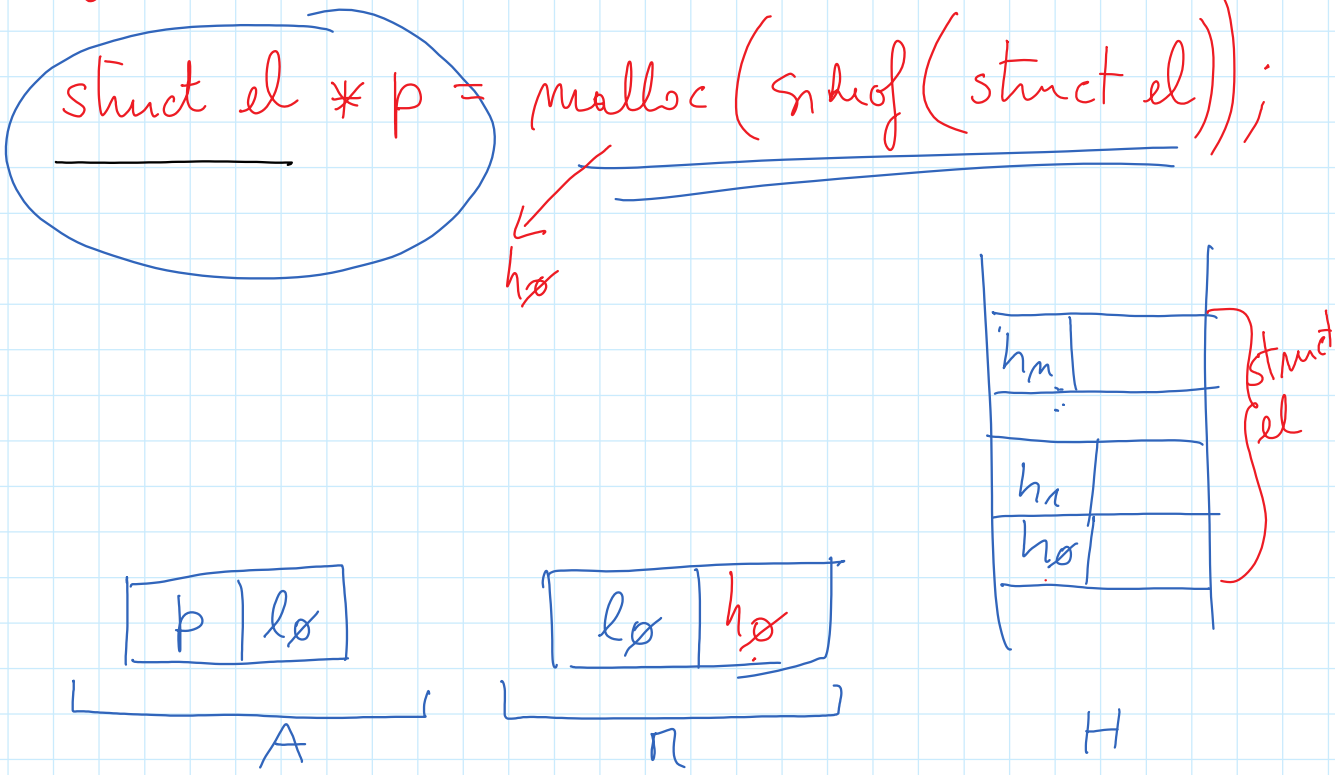
Spazio per un valore di tipo struct el.

p = malloc(sizeof(struct el))

quanto spazio?

struct el
{
...
};

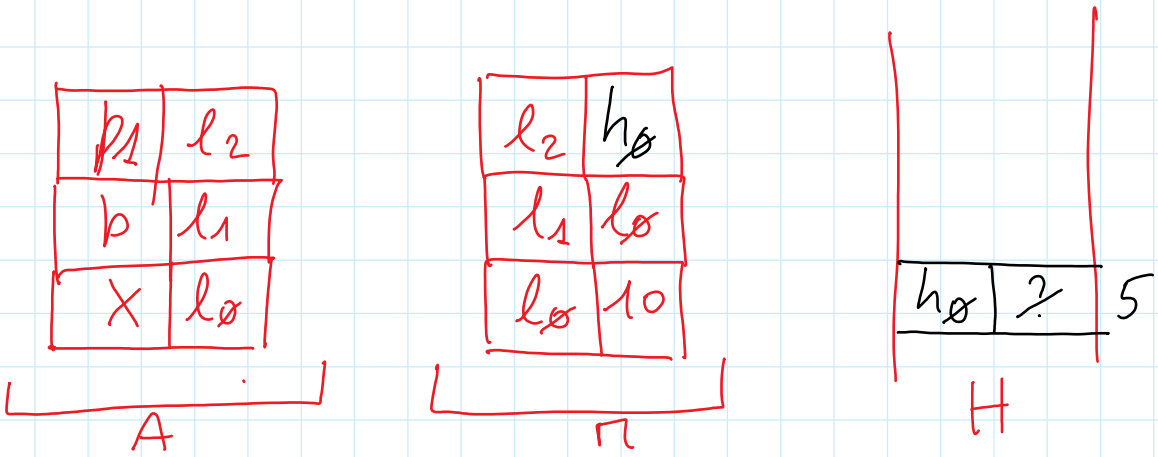
struct el *p = malloc(sizeof(struct el));



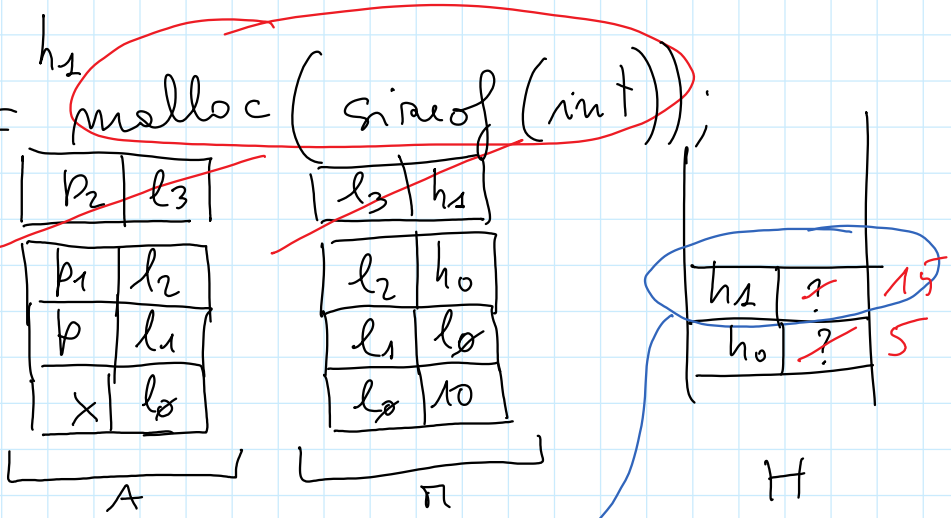
La memoria dinamica si gestisce attraverso indirizzi dati come valore a variabili puntatore.

```
{ int x = 10;
int * p = &x;
int * p1 = malloc(sizeof(int));
*p1 = 5;
```

h_0



```
{ int x = 10;
  int * p = & x;
  int * p1 = malloc (sizeof (int));
  * p1 = 5;
  { int * p2 = malloc (sizeof (int));
    * p2 = 15;
  }
  ⋮
```



→ garbage

gestione della memoria heap è indipendente dall'esecuzione dei blocchi

Procedura per liberare la memoria heap.

free (q)

l'indirizzo della memoria heap da liberare.

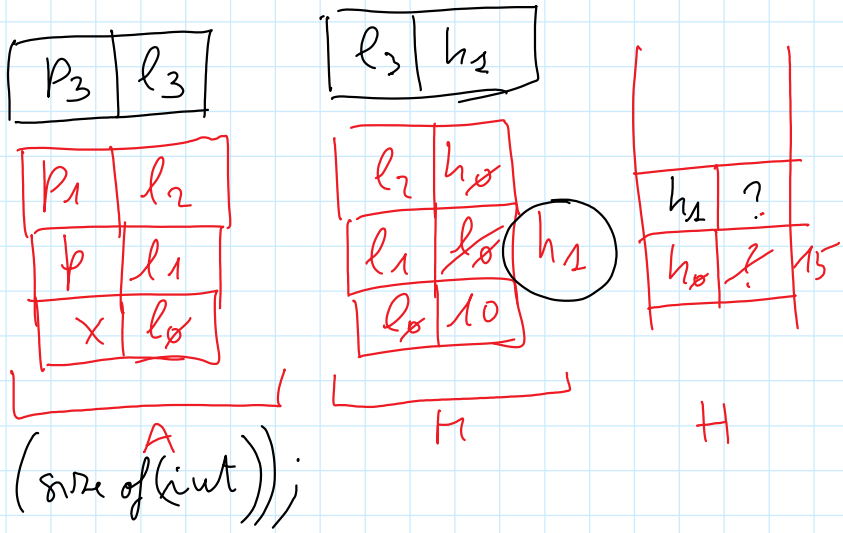
free (variabile che punta allo spazio da liberare)

ha un tipo (puntatore a un oggetto con un certo tipo)

```
struct el { ... };
struct el* p = malloc( sizeof (struct el));
free (p);
```

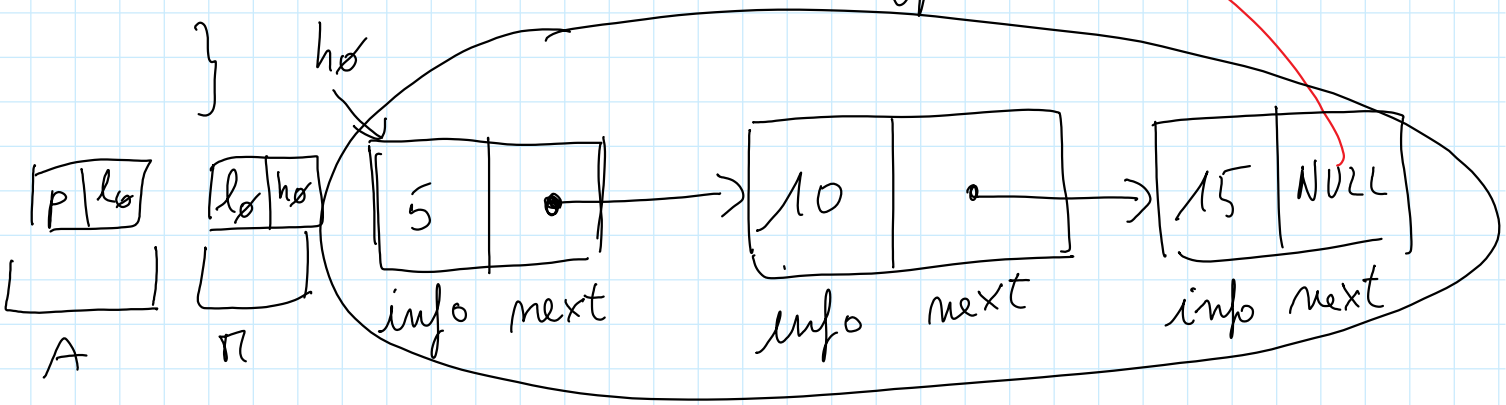
libera lo spazio di memoria occupato da un valore "struct el" puntato da p.

```
{
  int x = 10;
  int *p = &x;
  int *p1 = malloc(sizeof(int));
  *p1 = 15;
  {int *p2 = malloc(sizeof(int));
    *p2 = 20;
    p = p2;
  }
  free(p2);
  {int *p3 =
      malloc(sizeof(int));
```



```
struct node
{ int info;
  struct node* next;
};
```

il valore puntatore che non punta a niente



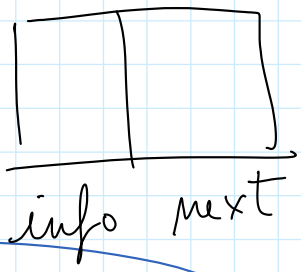
Una struttura dinamica che ci diamo

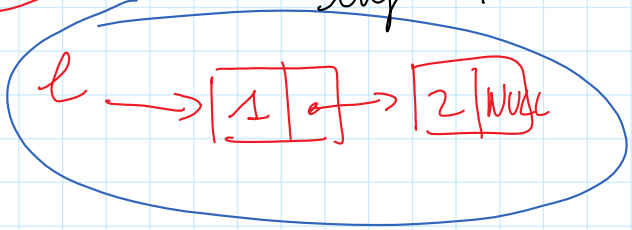
lista collegata (linked list)

typedef tipo nome;

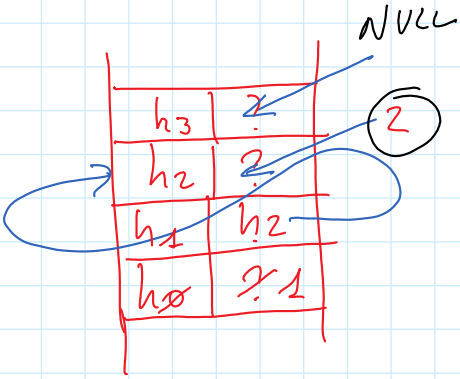
```
typedef
struct node
{
  int info;
  struct node * next;
}
ElementoDiLista;
```

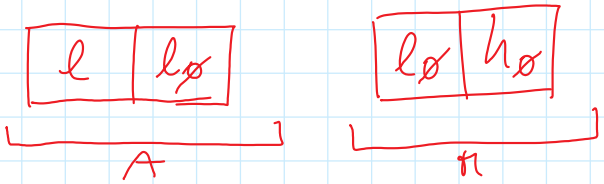
- tipo





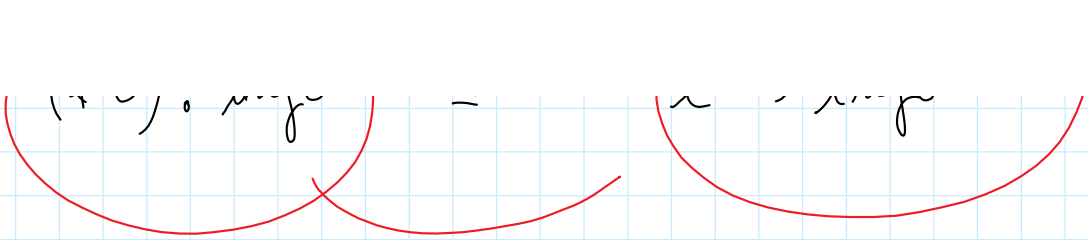

```
main()
{
  ElementoDiLista * l = malloc(sizeof(ElementoDiLista));
  (*l).info = 1;
  (*l).next = malloc(sizeof(ElementoDiLista));
  (*(*l).next).info = 2;
  (*(*l).next).next = NULL;
```





->

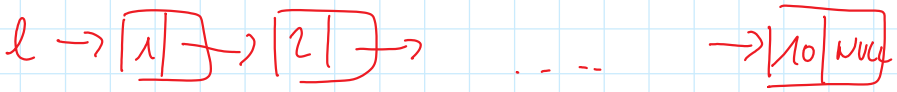
(*l).info ≡ l->info



lunedì 5 novembre 2018 10:51

```
{ ElementoDiLista * l = malloc(sizeof(ElementoDiLista));
  l -> info = 1;
  l -> next = malloc(sizeof(ElementoDiLista));
  l -> next -> info = 2;
  l -> next -> next = NULL;
```

l -> info ≡ (*l).info



lunedì 5 novembre 2018 10:54

```
main()
{ ElementoDiLista * corr;
  ElementoDiLista * l = malloc(sizeof(ElementoDiLista));
  int i;  l->info = 1;
  corr = l;
  for( i = 2 ; i <= 10 ; i++)
    { corr->next = malloc(sizeof(EDL));
      corr = corr->next;
      corr->info = i;
    }
  corr->next = NULL;
```