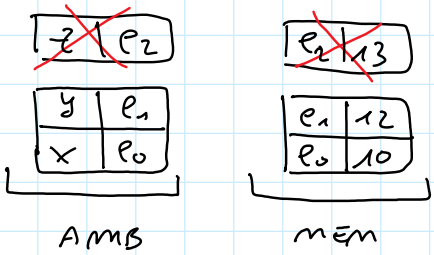
# MEMORIA DINAMICA

→ fino ad ora abbiamo visto la memoria nello STATO di un programma come una PILA DI FRAME (STACK)

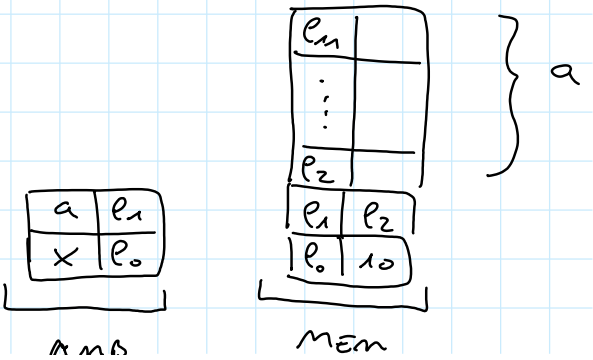
→ I frame in memoria corrispondono ai blocchi contenuti nel programma

```
{ int x = 10;
  int y = 12;
  .
  .
  .
    z = 13;
    :
  }
  :
}
```




in memoria posso allocare array di dimensione fissata

```
{ int x=10;
  int a[10];
  :
}
```



AND MEM

Se volessimo scrivere un programma che chiede all'utente di inserire una sequenza di numeri positivi terminata da un numero negativo, e poi stampa su schermo i numeri inseriti?

↳ Come facciamo a memorizzare tutti i numeri?

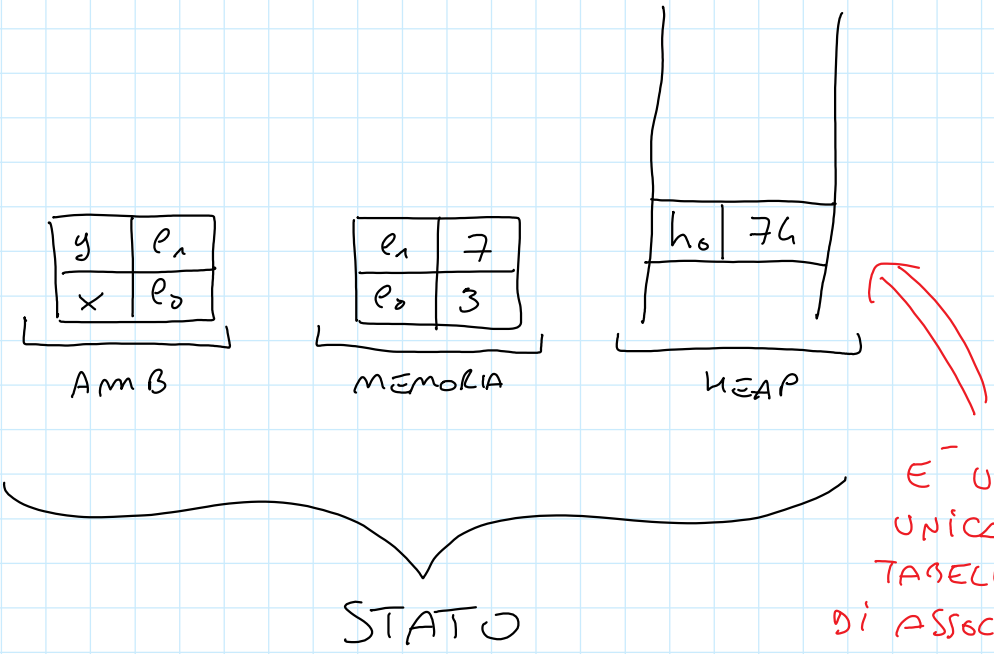
NON SAPPIAMO QUANTI SIANO....

NON POSSIAMO USARE UN ARRAY.

→ PER QUESTE SITUAZIONI DOBBIAMO USARE LA MEMORIA DINAMICA (HEAP)

# HEAP

→ É UNA PARTE DIVERSA DELLA MEMORIA IN CUI LE VARIABILI NON VENGONO ALLOCATE DAI BLOCCHI, MA TRAMITE DELLE OPERAZIONI FATTE DAL PROGRAMMA



LA MEMORIA NELLO HEAP VIENE ALLOCATA CHIAMANDO UNA FUNZIONE DELLA LIBRERIA

<stdlib.h>

malloc (....)

MEMORY ALLOCATION

→ QUESTA FUNZIONE CREA LA VARIABILE NELLO HEAP E RESTITUISCE UN PUNTATORE ALLA NUOVA VARIABILE

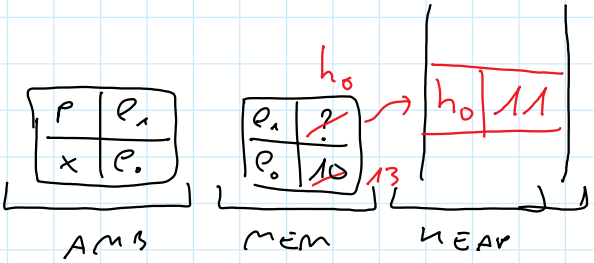
→ DOBBIAMO PASSARE ALLA MALLOC LA QUANTITÀ DI MEMORIA NECESSARIA PER LA VARIABILE DA ALLOCARE (IN BYTE)

- DIPENDE DAL TIPO DELLA VARIABILE
- CE LO DICE LA FUNZIONE sizeof (....)

sizeof (int)
sizeof (double)

# ESEMPIO

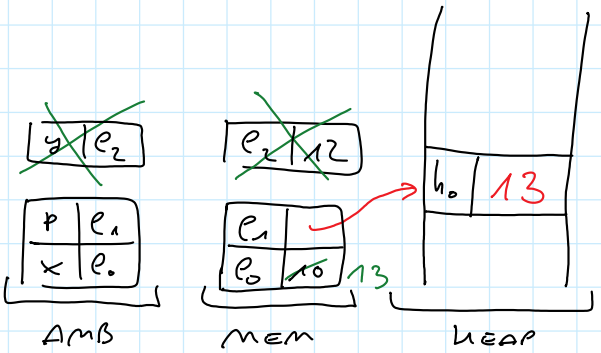
```
{ int x = 10;
  int *p;
  :
  p = malloc(sizeof(int));
  :
  *p = x+1;
  x = *p + 2;
}
```




HO CREATO UNA VARIABILE CHE NON HA UN NOME, MA DI CUI HO UN PUNTATORE

LA VARIABILE NELLO HEAP NON È INFLUENZATA DAI BLOCCHI

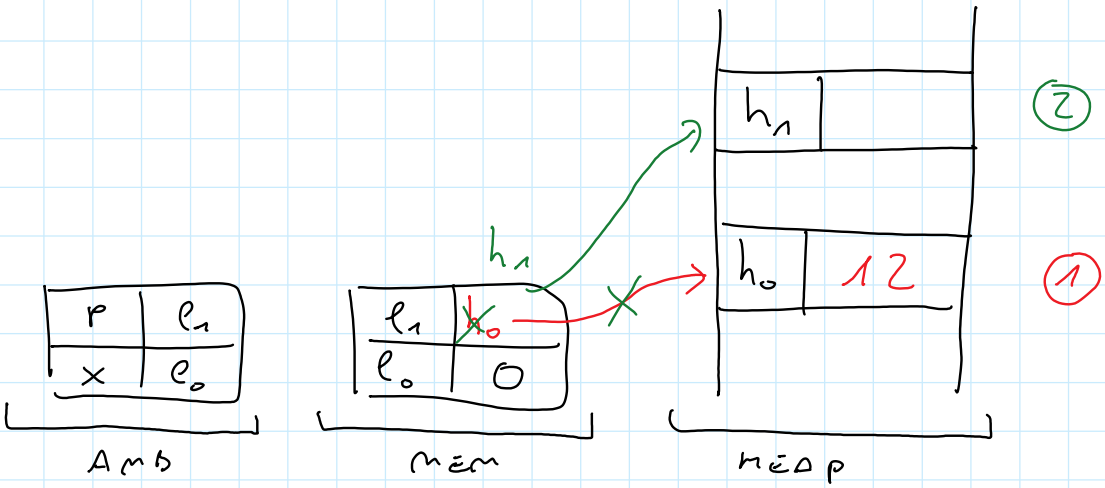
```
{ int x=10;
  int *p;
  {
    int y=12;
    p = malloc(sizeof(int));
    *p = y+1;
  }
  x = *p;
}
```




LA VARIABILE CREATA SOPRAVVIVE ALLA CHIUSURA DEL BLOCCO

PROBLEMA: E SE MODIFICO p?

```
{ int x=0;
  int *p = malloc(sizeof(int));   (1)
  *p = 12;
  p = malloc(sizeof(int));        (2)
  ⋮
}
```




LA VARIABILE CREATA CON LA PRIMA MALLOC NON È PIÙ UTILIZZABILE (NON LO PIÙ NESSUN PUNTATORE CHE LA RAGGIUNGE)

È GARBAGE

PER EVITARE DI ACCUMULARE GARBAGE

É BENE LIBERARE LO HEAP DOPO

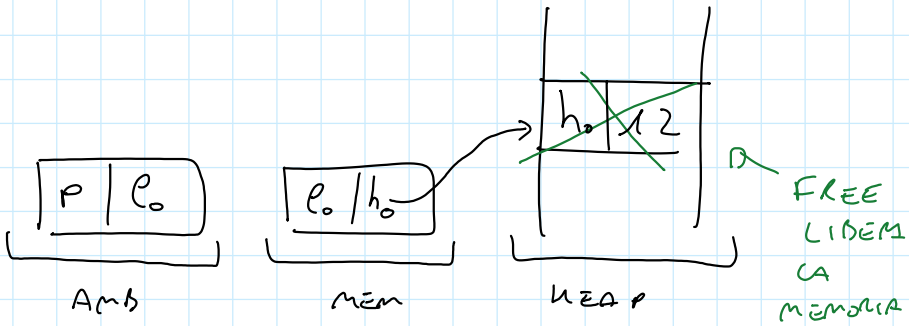
AVER FINITO DI USARE UNA VARIABILE

↳ SI USA LA FUNZIONE

```
free(p)
```

↑ PUNTATORE ALLA VARIABILE DA LIBERARE

```
{ int *p = malloc(sizeof(int));
  *p = 12;
   ⋮
  free(p);
   ⋮
}
```



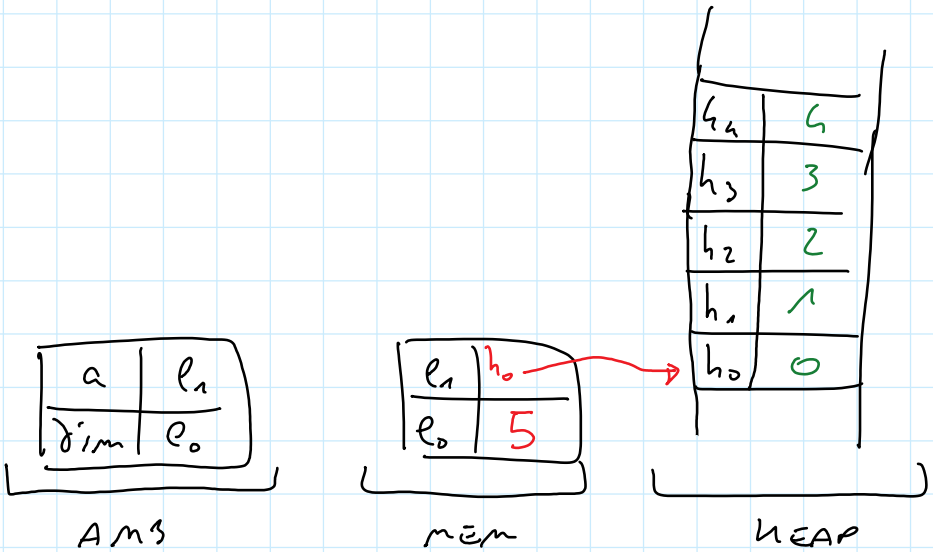

DOPO AVER FATTO LA FREE DEVO FAR ATTENZIONE A NON DEREFERENZIARE p!! *p

POSSO RIASSEGNARE p...

```
p = &x;
```

POSSO USARE LO HEAP PER ALLOCARE UN ARRAY DI DIMENSIONE NON FISSATA A PRIORI

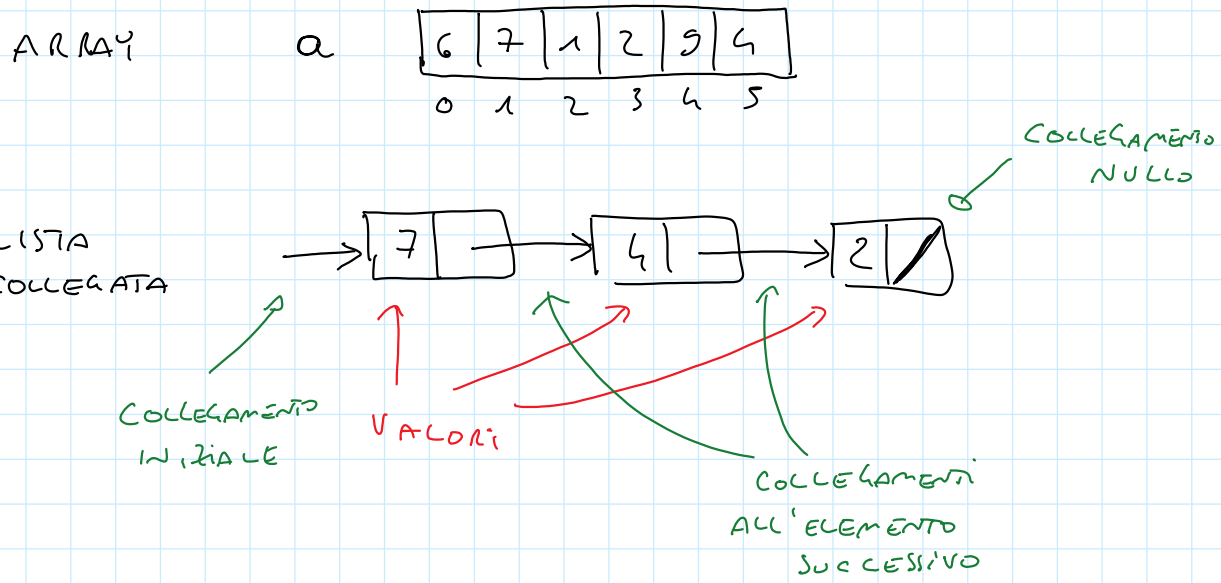
```
{ int dim;
  scanf("%d", &dim);
  int *a = malloc(dim * sizeof(int));

  for (int i=0; i<dim; i++)
      a[i]= i;

  ⋮
}
```



# LISTE IN C

Una lista collegata (linked list) è una sequenza di elementi dello stesso tipo che può avere una dimensione variabile.

## RAPPRESENTAZIONE

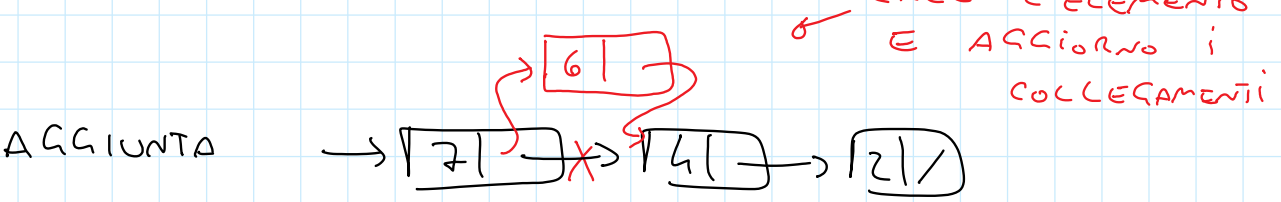



→ Gli array hanno un ACCESSO DIRETTO: scrivo a[i] e immediatamente ottengo il valore in posizione i

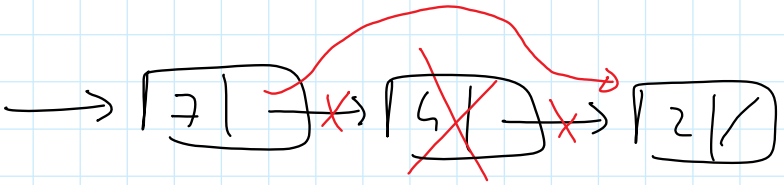
→ Le liste hanno un ACCESSO SEQUENZIALE: l'unico punto di accesso è il collegamento iniziale. Per ottenere l'i-esimo elemento devo scorrere tutti i precedenti

IL VANTAGGIO DELLE LISTE È CHE POSSO AGGIUNGERE E RIMUOVERE ELEMENTI SEMPLICEMENTE MODIFICANDO I COLLEGAMENTI

AGGIUNTA




RIMOZIONE

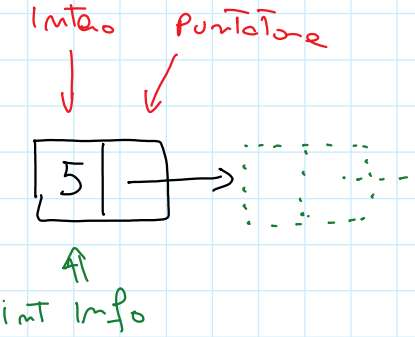


COME SI REALIZZA UNA LISTA (DI INTERI) IN C?

→ i collegamenti tramite puntatori

→ gli elementi tramite strutture da allocare nello heap.

## STRUTTURE IN C (STRUCT)




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

in questo modo si definisce una struttura fatta di due parti: una variabile di tipo int e un puntatore ad un'altra struttura analoga

Possiamo dare un nome al Tipo descritto dalla struttura tramite il costrutto typedef

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

DEFINISCE UN NUOVO TIPO ElementoDiLista LE CUI VARIABILI SARANNO STRUTTURE elemento

```
typedef ElementoDiLista* ListaDiElementi;
```

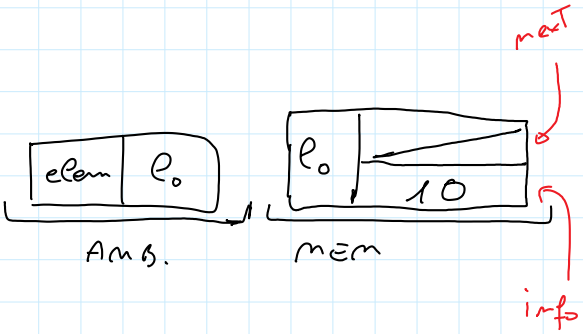
DEFINISCO UN NUOVO TIPO ListaDiElementi LE CUI VARIABILI SARANNO PUNTATORI A ElementoDiLista

# USIAMO GLI ELEMENTI

martedì 7 novembre 2017 12:42

## ESEMPI

```
main () {
    ElementoDiLista elem;
    elem.info = 10;
    elem.next = NULL;
}
```




COSTANTE CHE RAPPRESENTA UN PUNTATORE NULLO (NON PUNTA A NIENTE)

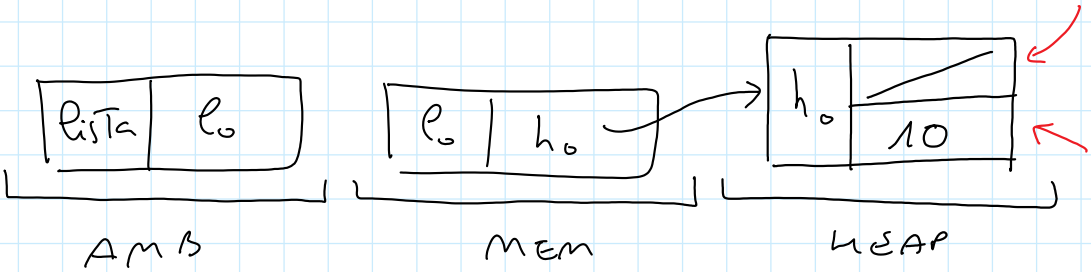
. mi fa accedere ai componenti della struttura (info e next)

FACCIAMO LA STESSA COSA NELLO HEAP

corrisponde a ElementoDiLista*

```
main() {
    ListaDiElementi lista;
    lista = malloc(sizeof(ElementoDiLista));
    (*lista).info = 10;
    (*lista).next = NULL;
}
```

`lista->info=10;`
`lista->next=NULL`




QUANDO ACCEDO ALLA STRUTTURA TRAMITE UN PUNTATORE POSSO SCRIVERE lista->... AL POSTO DI *lista. ...