Scrivere una funzione che calcola il numero di elementi positivi in un array di interi

```
int contapos (int a[], int dim)
{ int i; int cont = 0;
  for (i=0; i<dim; i++)
    if(a[i]>0) cont++;
  return cont;
}
```

equivalente a $i = i+1$

In C l'assegnamento è un'espressione !!!

$$w = (x = y + z) + 3;$$

è un assegnamento, quindi modifica la variabile x, ma è usato come espressione il suo valore è $y+z$

Un assegnamento $x = e$ è una espressione

1) calcola il valore di "e" e lo assegna a x
2) ha come valore il valore calcolato in 1)

$$w = (x = 5) + 3;$$

- assegna a x il valore 5
- assegna a w il valore 8

```
i = 8;
x = (i++) + 5;
```

- i vale 9
- x vale ~~14~~ 13

```
x = (++i) + 5;
```

- i vale 9
- x vale 14

Trovare il minimo e il massimo di un array

```
void minmax (int a[], int dim, int *min, int *max)
{ int i; *min = a[0]; *max = a[0];
  for (i = 1; i < dim; i++)
      if (a[i] < *min) *min = a[i];
      else if (a[i] > *max) *max = a[i];
}
```

```
{ int a[15]; int b[35]; int mina, minb;
  int maxa, maxb;
    :
  minmax(a, 15, &mina, &maxa);
  minmax(b, 35, &minb, &maxb);
```

Alternativa: soluzione "ibrida"

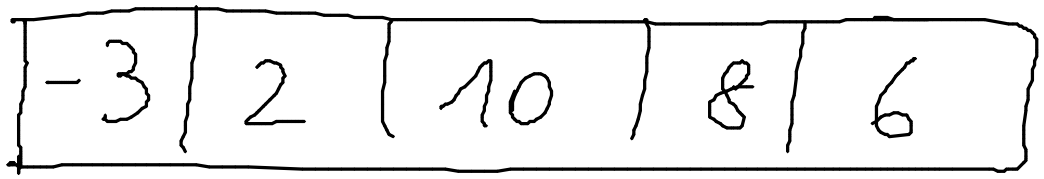
```
int minmax(int a[], int dim, int *max)
{ int i; int min = a[0];
  *max = a[0];
  for (i=0; i<dim; i++)
     if (a[i] < min) min = a[i];
     else if (a[i] > *max) *max = a[i];
  return min;
}
```

minimo

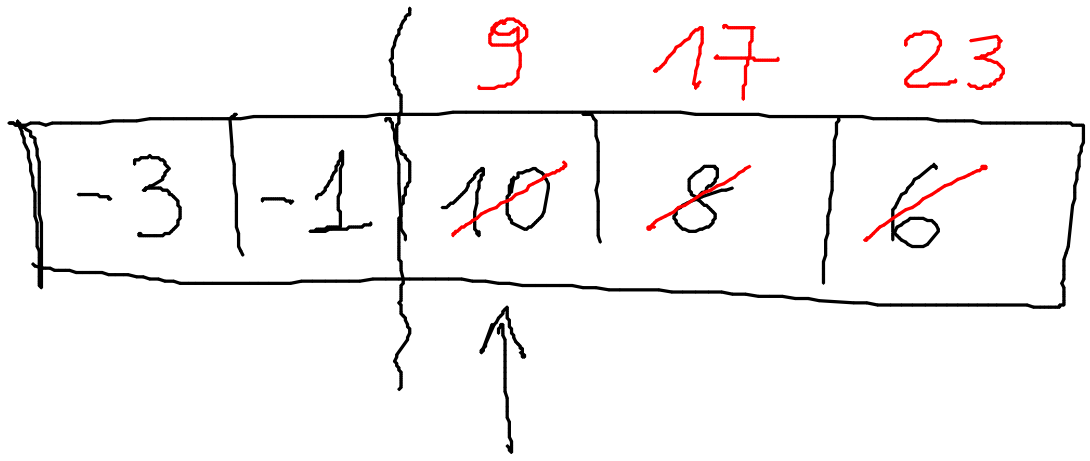
lascia in *max il massimo

```
{ int a[15]; int b[35]; int mina, minb;
  int maxa, maxb;

  ;

  mina = minmax(a, 15, &maxa);
  minb = minmax(b, 35, &maxb);
```

Definire una procedura che, dato un array a, sostituisce ogni elemento di a con la somma tra l'elemento stesso e tutti quelli che lo precedono



array prima della chiamata della procedura

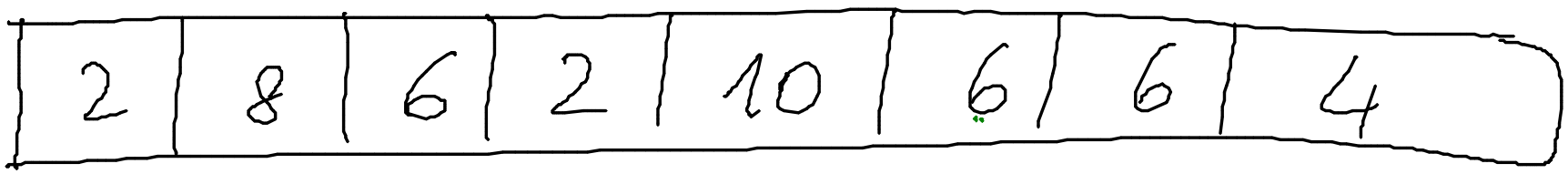


array dopo la chiamata

```
void sum (int a[], int dim)
{ int i;
   for (i = 1; i < dim; i++)
       a[i] = a[i] + a[i-1];
}
```

```
{ int b[5];
  ......
  sum (b,5);
```

Definire una procedura C che dato un array raddoppia tutti gli elementi che sono preceduti nell'array da sé stessi



prima

dopo

```
void doppio (int a[], int dim)
  {
    int i;
    for (i = dim-1; i > 0; i = i-1)
    { int trovato = 0; int j = 0;
      while (!trovato && j < i)
        if (a[j] == a[i])
          trovato = 1;
        else j = j+1;
      if (trovato) a[i] = 2 * a[i];
    }
  }
```

i qui dentro è fissato

i--

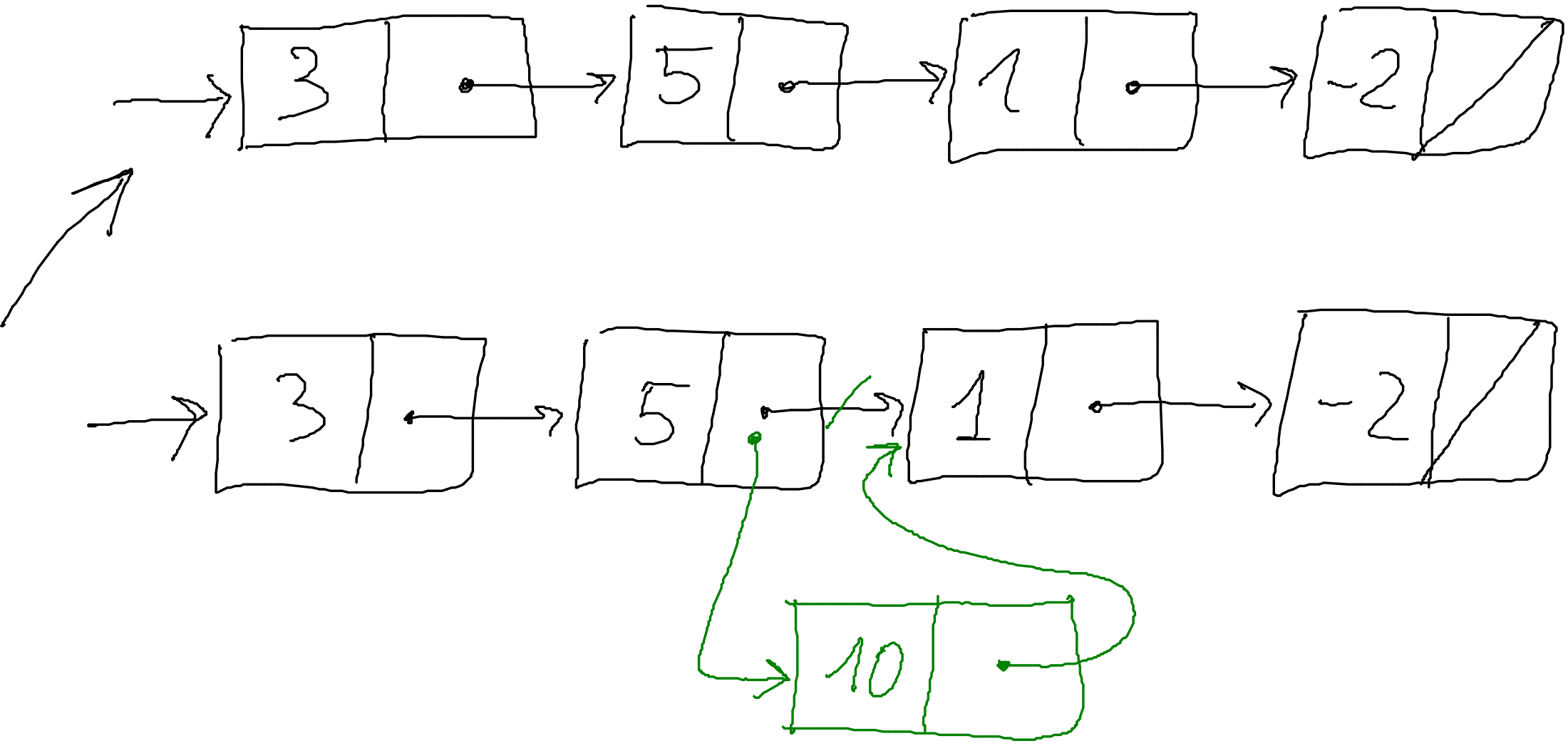
```
void doppio ( . - - - . - )
        :
        :
        :
    for (i = dim - 1; i > 0; i--)
        { int j = 0; int trovato = 0;
          while (! trovato && j < i)
          if (a[i] == a[j])
              { trovato = true ; a[i] = 2 * a[j]; }
          else    j = j + 1;
    }
```

LISTE : sequenze di elementi omogenei
ma di lunghezza non nota a priori

STRUCT : consente di definire variabili
STRUTTURATE

```
struct persona { int giorno; int mese; int anno;
                 char sesso;
                 char nome[40];
                 char cognome[100];
                 int coniugato; }
```

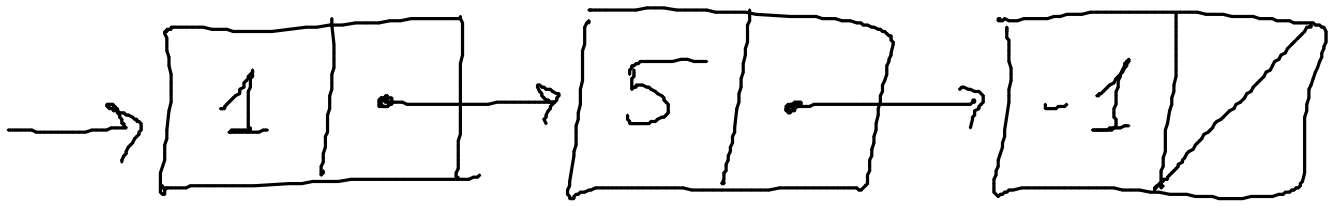
LISTA COLLEGATA (linked list)



Ci serve una struttura che contiene

1) un intero (l'informazione)
2) un COLLEGAMENTO all'elemento successivo nella sequenza

MEMORIA HEAP

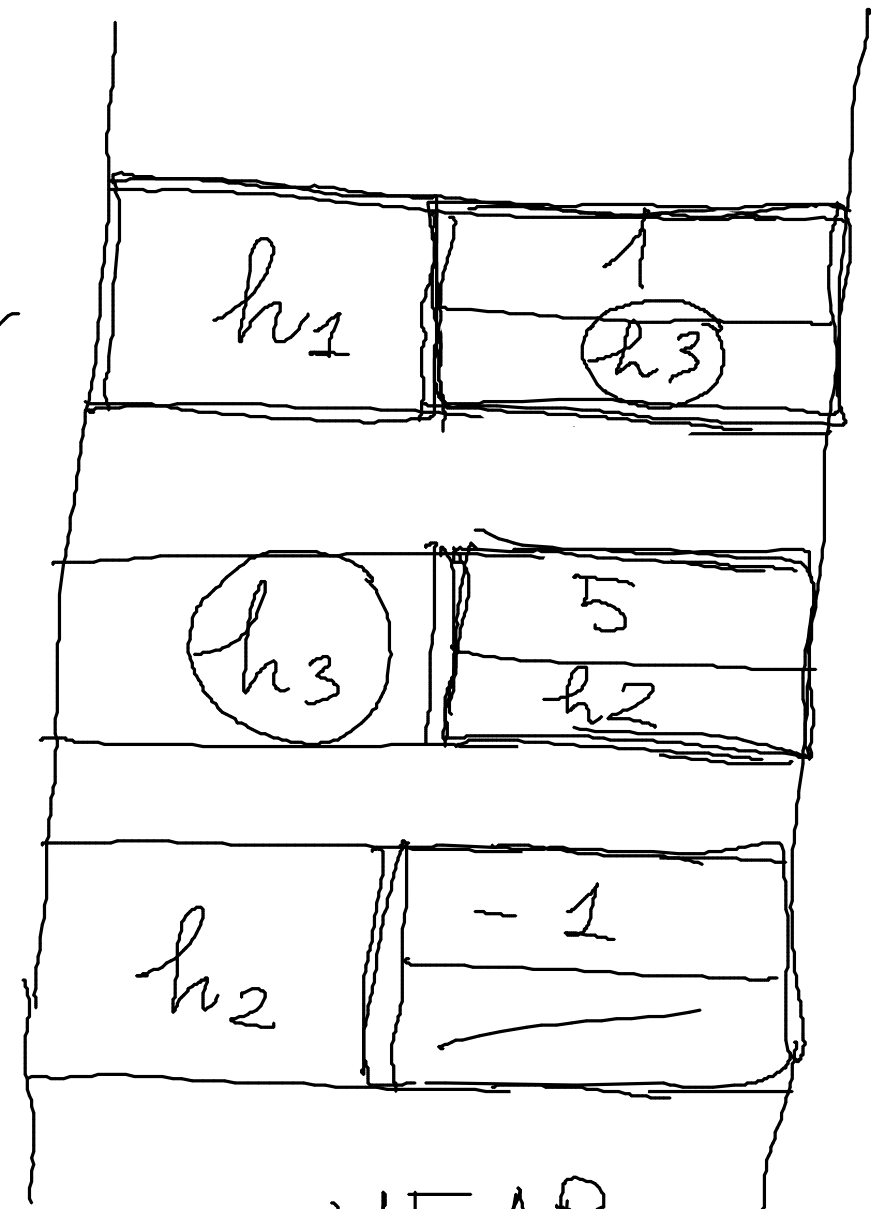


1 :: 5 :: -1 :: [ ]

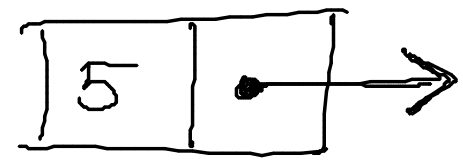
CAML

h1, h2, h3

Sono indirizzi nella memoria heap (locazioni)



1) Definire una STRUCT che rappresenta gli oggetti della sequenza




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

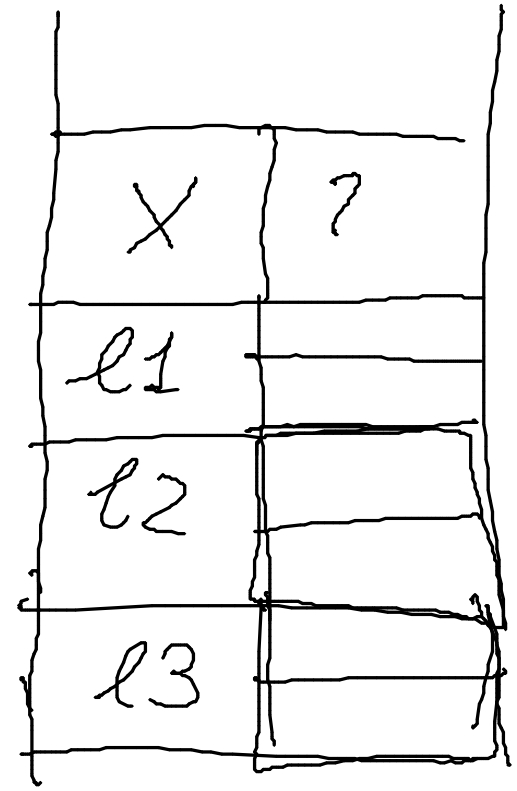
```
typedef struct el ElementoLista;
```

{ struct el ..... ; typedef struct el Elementolista;
struct elc {char c; struct elc *next;}

Elementolista el1, el2, el3;

int x;

l



μ

ALLOCAZIONE DINAMICA DI MEMORIA (HEAP)

malloc : vuole come argomento il numero di byte di memoria richiesti

sizeof (tipo) : numero di byte necessario per contenere valori del tipo

malloc (sizeof (Elemento lista))

restituisce l'indirizzo dello spazio allocato.

```
{  Elemento_lista *p;
   int x;   int *q;
   :
   x = 10;   q = &x;
   p = malloc (sizeof (Elemento_lista));
```

q	l3
x	l1
p	l2

ρ

l3	l1
l2	? h1
l1	10

μ

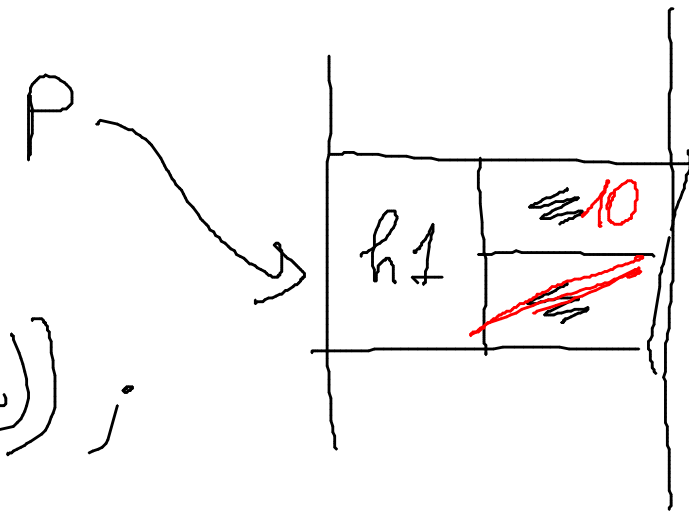
h1	~~1~~

heap

Come si accede (in lettura e scrittura)
ai campi di una struttura

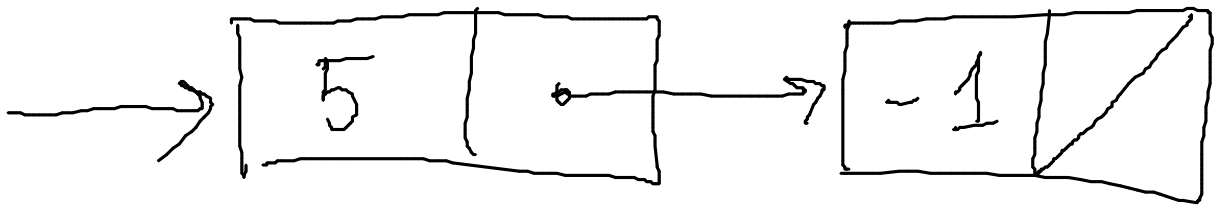
Supponiamo di avere:

```
Elementolista *p;
p = malloc(sizeof(Elementolista));
p -> info = 10;
p -> next = NULL;
```




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

Costruiamo una lista di 2 elementi




```
{ Elementolista *p;

  p = malloc (sizeof (Elementolista));
  p -> info = 5;
  p -> next = malloc (-----);
  (p -> next) -> info = -1;
  (p -> next) -> next = NULL;
```