Titolo nota 22/10/2015

RICEVIMENTO : MERCOLEDÌ alle 14:30 (aula su www.di.unipi.it/~paolo/)
VENERDÌ alle 14:00 (" " ")

GIOVEDÌ MATTINA : esercitazione
" POMERIGGIO : simulazione del compitino

o ——— o

Titolo nota 22/10/2015

SEMANTICA del BLOCCO

$Com ::= \{ Declist\ Comlist \}$

$Sem_{cl}\ C\ \rho\ \mu = Sem_c\ C\ \rho\ \mu$ se $C \in Com$

$Sem_{cl}\ [C\ CL]\ \rho\ \mu = Sem_{cl}\ CL\ \rho\ (Sem_c\ C\ \rho\ \mu)$

memoria modificata dall'esecuzione di C

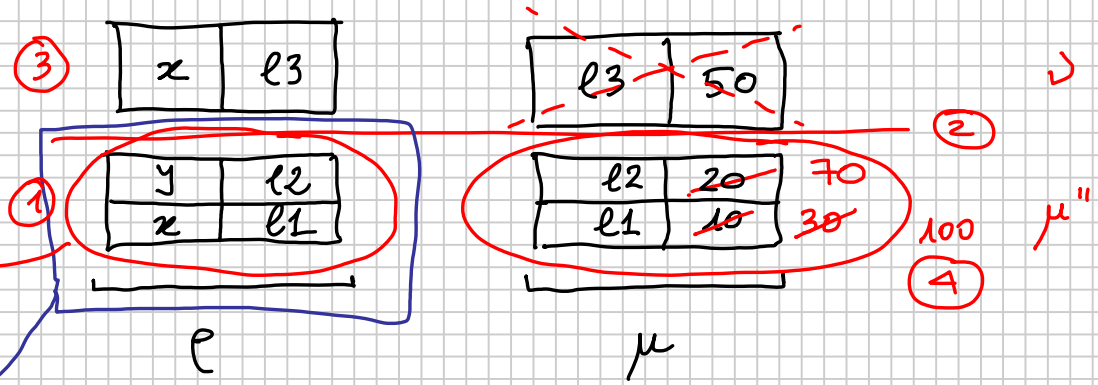
$Sem_c\ \{DL\ CL\}\ \rho\ \mu = \mu''$

dove $Sem_{dl}\ DL\ \omega\ \rho\ \omega\ \mu = \varphi.\rho\ \nu'.\mu$

$Sem_{cl}\ CL\ \varphi.\rho\ \nu'.\mu = \nu.\mu''$

```
{ int x = 10;   (1)
  int y = 20;
  x = x+y;      (2)
  {int x = 50;  (3)
    y = y+x;
  }
  x = x+y;      (4)
}
```

→ viene eseguito nell'ambiente



Titolo nota 22/10/2015

```
{   int x = 10;
    int y = 20;
    { int x = 50;
      x = x + 1;
      y = x + y;
    }
    y = x + y;
}
```

x = x + 1; ← assegnamento alla variabile "locale" al blocco

y = x + y; (interno) ← assegnamento alla variabile globale

y = x + y; (esterno)

viene eseguito nello stato $\varphi_1 . \Omega$, $\nu_1 . \Omega$

viene eseguito nello stato $\varphi_2 . \varphi_1 . \Omega$, $\nu_2 . \nu_1 . \Omega$

viene eseguito nello stato $\varphi_2 . \varphi_1 . \Omega$, $\nu_3 . \nu_1 . \Omega$

viene eseguito nello stato $\varphi_1 . \Omega$, $\nu_4 . \Omega$ e produce la memoria $\nu_5 . \Omega$

φ_2

x	l3

ν_2

l3	~~50~~ 51

$\nu_3 = \nu_2 \left[51 / l_3 \right]^{mod}$

φ_1

y	l2
x	l1

ν_1

l2	~~20~~ ~~71~~ 81
l1	10

$\nu_4 = \nu_1 \left[71 / l_2 \right]^{mod}$

$\nu_5 = \nu_4 \left[81 / l_2 \right]^{mod}$

ρ μ

Titolo nota 08/10/2015

Assegnamento

Com ::= Ide = Exp

$$Sem_c\ x = E\ \rho\ \mu = \mu\left[{}^{v}/_{\rho(x)}\right]^{mod}$$

dove $v = Sem_e\ E\ \rho\ \mu$

Sia $\mu = \nu_1 . \nu_2 . \ldots . \nu_k . \Omega$ e supponiamo che $\rho(x)$ sia una locazione tale che $\nu_i(\rho(x)) \neq \perp$

Allora, per definizione di $\pi\left[{}^{b}/_{a}\right]^{mod}$, $\mu\left[{}^{v}/_{\rho(x)}\right]^{mod}$ è la pila

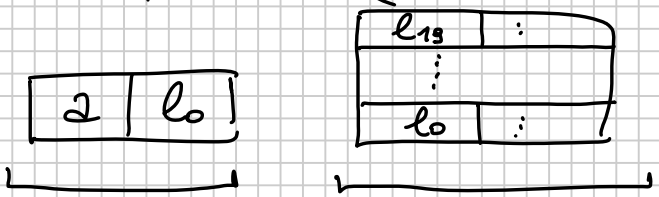
$$\nu_1 . \nu_2 . \ldots . \nu_{i-1} . \nu_i\left[{}^{v}/_{\rho(x)}\right] . \nu_{i+1} . \ldots . \nu_k . \Omega$$

è una NUOVA pila memoria

Titolo nota 08/10/2015

MEMORIA DINAMICA (heap)

Il linguaggio fino ad ora ci consente solo di usare memoria per le variabili dichiarate nei blocchi. Abbiamo anche la possibilità di usare STRUTTURE (array) però di dimensione prefissata (int a[20], array di 20 elementi)



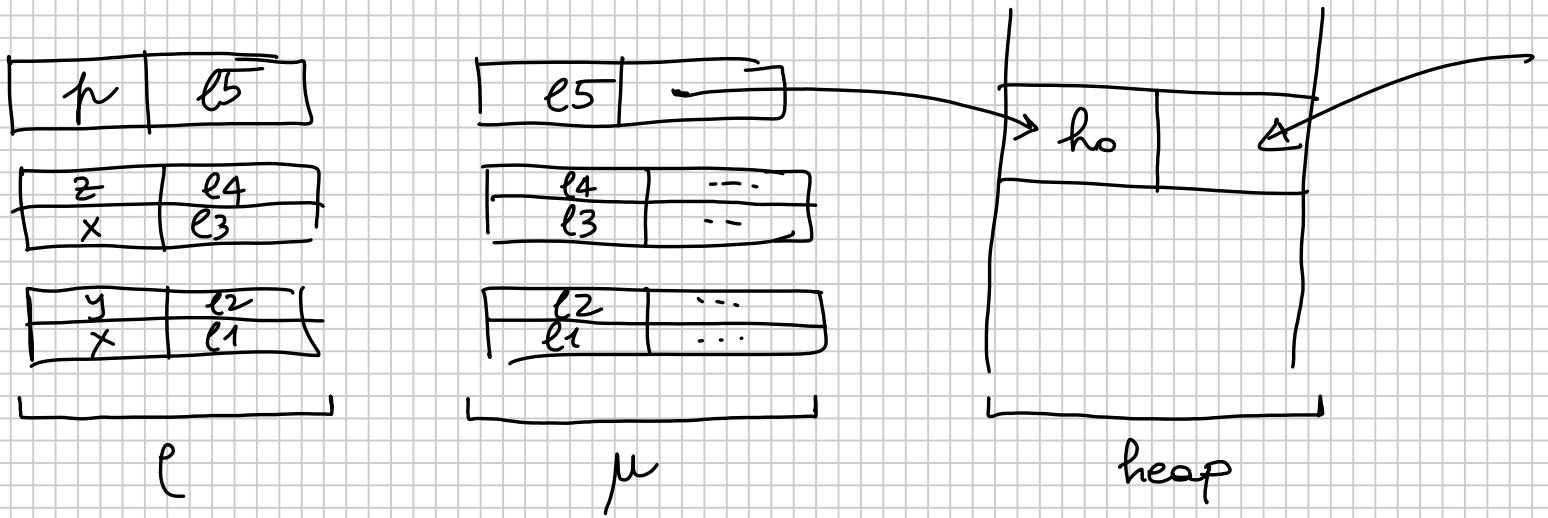
C'è la necessità spesso di utilizzare MEMORIA DINAMICA (la memoria necessaria non è nota nel momento in cui si scrive il programma, ma dipende dall'esecuzione)

La memoria µ viene gestita A PILA: ogni volta che si dichiara una variabile (o che si chiama una procedura / funzione) viene riservata in cima alla pila la quantità di memoria necessaria per "accogliere" variabili / parametri ecc.

Titolo nota 08/10/2015

La memoria DINAMICA (HEAP) è una ULTERIORE porzione dello stato che consente di ALLOCARE spazio per contenere variabili ANONIME.
Sono variabili che NON vengono DICHIARATE mediante una DICHIARAZIONE, ma vengono "richieste" al momento dell'esecuzione, quando servono!

Le variabili ANONIME vengono gestite attraverso PUNTATORI



a questo valore non si accede attraverso un nome (come per x, y, z, p in figura) ma solo attraverso un puntatore

*p

Titolo nota 08/10/2015

ALLOCAZIONE DINAMICA di MEMORIA nello HEAP

In C si usa la funzione di <stdlib.h> malloc

```
int *p;
.....
p = malloc (sizeof (int))
```

tipo

⟶ `p = new;`

da qui in poi, attraverso p, posso utilizzare una NUOVA variabile di tipo int.

- Nel modello formale "astraiamo" dai dettagli e usiamo una versione semplificata di malloc

  new : è una nuova espressione che restituisce un indirizzo nello heap.

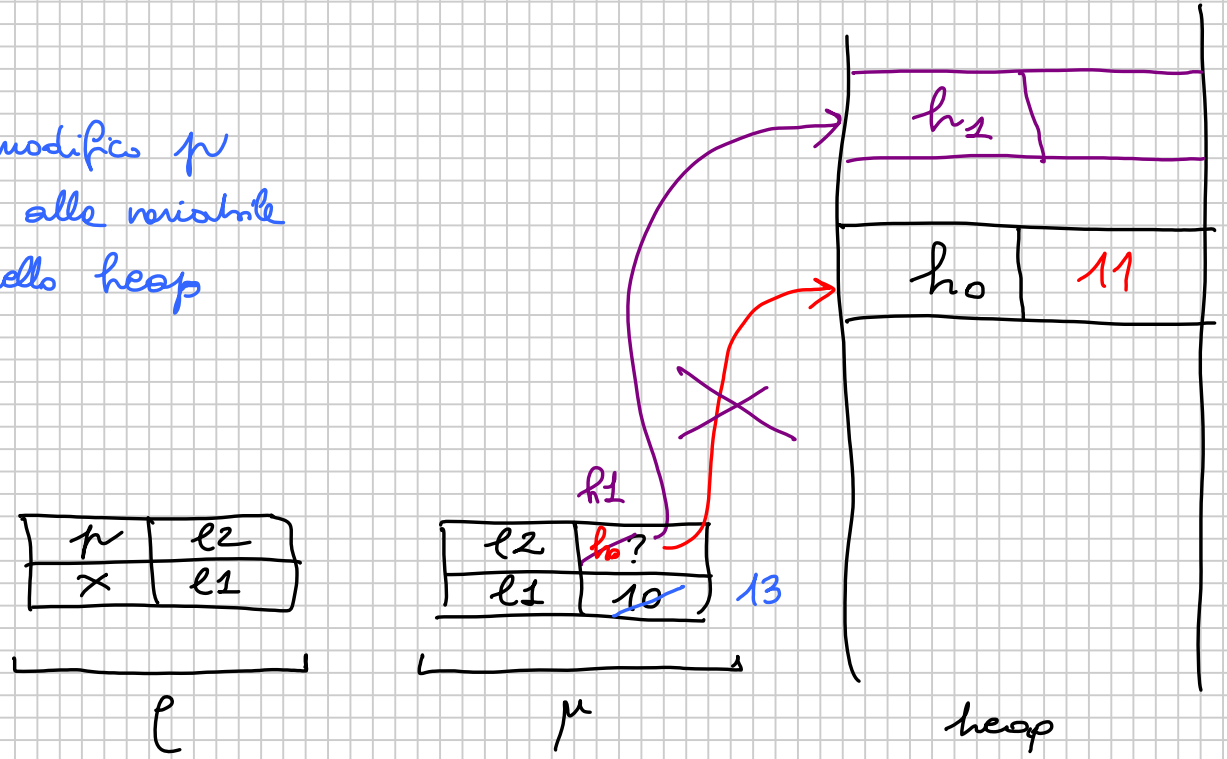
Titolo nota 08/10/2015

ESEMPIO

```
{
    int x = 10;
    int *p;
    ⋮
    p = new;
    *p = x+1;
    x = *p+2;
    ⋮
    p = new;
}
```

finché non modifico p $*p$ punta alla variabile ANONIMA dello heap

questo assegnamento fa sì che la variabile ANONIMA all'indirizzo h_0 non sia più utilizzabile (abbiamo prodotto GARBAGE)



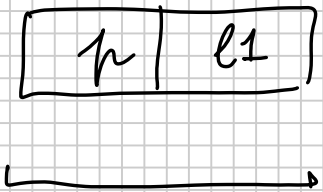
I sistemi prevedono algoritmi sofisticati di GARBAGE COLLECTION

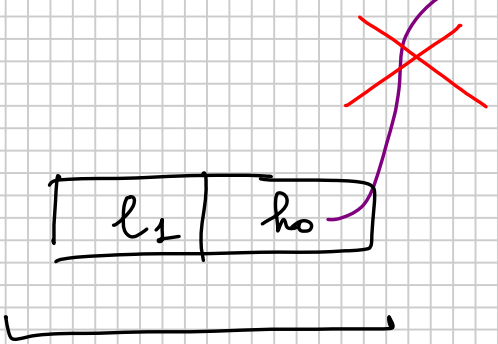
Titolo nota 08/10/2015

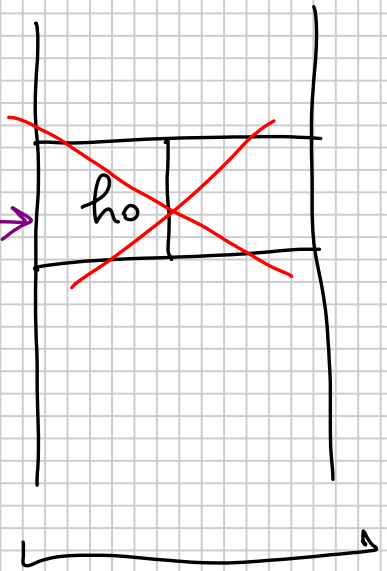
RESTITUZIONE di MEMORIA DINAMICA

free (p) è un comando che restituisce allo heap la variabile anonima puntata da p.

```
{ int *p;
  ⋮
  p = new;  (1)
  ⋮
  free (p);  (2)
  ⋮
}
```





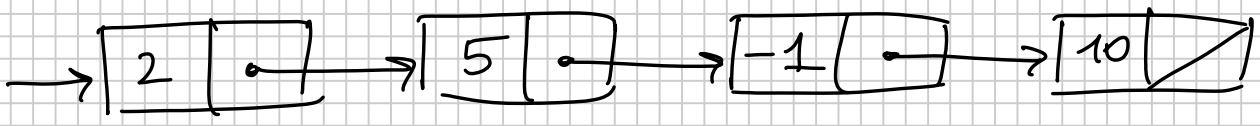


Titolo nota 08/10/2015

Esempio

```
{ int i;   int *a[10];
  ⋮
  for (i=0; i<10; i++)
     a[i] = new;
```

L'uso che faremo dello heap è quello che consente di creare LISTE di oggetti



Ceremi sul modello formale dopo il 1° compitino ....

Titolo nota 08/10/2015

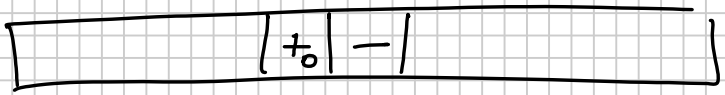
ESERCIZI di C

Scrivere una funzione

int check (int a[], int dim)

che controlla che nell'array a tutti i valori negativi precedano tutti i valori non negativi.

(1) true se la prop. è verificata
(0) false altrimenti



$$\left(\forall i \in [0, dim-1) \;.\; a[i] \geq 0 \Rightarrow a[i+1] \geq 0\right)$$

Cerco in [0, dim-1) un indice che NON verifica la proprietà

$$a[i] \geq 0 \wedge a[i+1] < 0$$

```
int check (int a[], int dim)
{ int i = 0; int trovato = 0;
  while (i < dim-1 && !trovato)
     if(a[i] >= 0 && a[i+1] < 0)
           trovato = 1 ;
     else  i++;
  return (!trovato);
}
```

```
int ok = 1;
        && ok
ok = 0;
return (ok);
```

Titolo nota 08/10/2015

```
int check (int a[], int dim)
{ int i; int ok; = 1
  for (i=0; i< dim-1; i++)
    if (a[i]>=0 && a[i+1]<0) ok=0
        else ok=1;
  return (ok);
}
```

errore tipico di chi usa impropriamente l'iterazione determinata al posto dell'iterazione indeterminata

Che valore restituisce questa funzione?

restituisce il valore di verità di $!(a[dim-2]>=0 \;\&\&\; a[dim-1]<0)$

Titolo nota 08/10/2015

```
void magØ (int a[], int dim, int *num)
/* restituisce nella variabile puntata da num, il numero di elementi di a
   maggiori di Ø.  */
{ int i; int conta =Ø; *num =Ø;
  for (i=Ø; i<dim; i++)
    if (a[i]>Ø) conta++; (*num)++;
  *num = conta;
}
```

```
main ( )
{ int b [20]; int x;
  < Legge gli elementi
    di b >
  magØ (b, 20, &x);
}
```

Titolo nota 08/10/2015

```
int diff (int a[], int dim)
/* restituisce il valore massimo delle differenze tra
   un valore di a e quello successivo
      max { a[i] - a[i+1] | i ∈ [0, dim-1) } */
{     int i; int massimo = a[0]-a[1];

      for( i=0; i<dim-1; i++ )
        if((a[i] - a[i+1]) > massimo)
              massimo = a[i] - a[i+1];
      return (massimo);
}
```