lunedì 23 settembre 2019 09:12

Blocchi

Funzioni e procedure

www.di.unipi.it
materiale didattico

Blocco (comando)

{

Sequenza di dichiarazioni
Sequenza di comandi

}

all'uscita del blocco tutte le variabili dichiarate all'interno del blocco vengono cancellate dallo stato.

Si ottiene facilmente utilizzando lo stato strutturato come una "pila di frame".

Tre regole per l'esecuzione dei blocchi:

- Quando si esegue un blocco (quando si incontra una parentesi graffa aperta) si aggiunge un frame vuoto sia sull'ambiente che sulla memoria (push)

- Ogni dichiarazione aggiunge una associazione sul primo frame di ambiente e sul primo frame di memoria.
- Quando si esce da un blocco (quando si incontra una parentesi graffa chiusa) si toglie un frame dell'ambiente e della memoria. (pop)

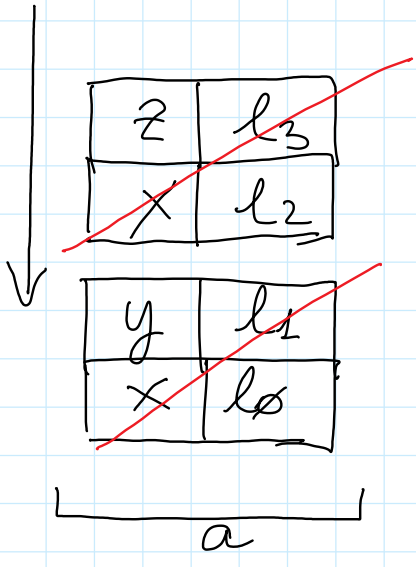
```
{ int x = 5;
  int y = 10;
  { int x = 7;
    int z = 5;
    x = x + z;
  }
  y = y + x;
}
```

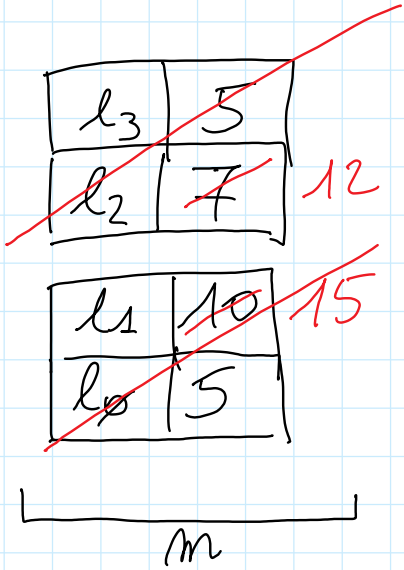
blocco esterno

blocco interno

12

15





lunedì 23 settembre 2019 09:30

FUNZIONI e PROCEDURE

parti di programma parametriche, con un nome, che possono essere usate su argomenti diversi.

PROCEDURE sono parti di programma che, quando usate (chiamate), modificano lo stato (come i comandi)

FUNZIONI " . In più restituiscono un valore.

La funzione che calcola il MCD tra due interi positivi (dichiarazione)

tipo del risultato — nome della funzione

```
int MCD ( int n, int m)
{ while (n != m)
    if (n > m)
        n = n - m;
    else
        m = m - n;
  return n;
}
```

sequenza dei parametri, ciascuno dei quali è definito da tipo e nome

PARAMETRI

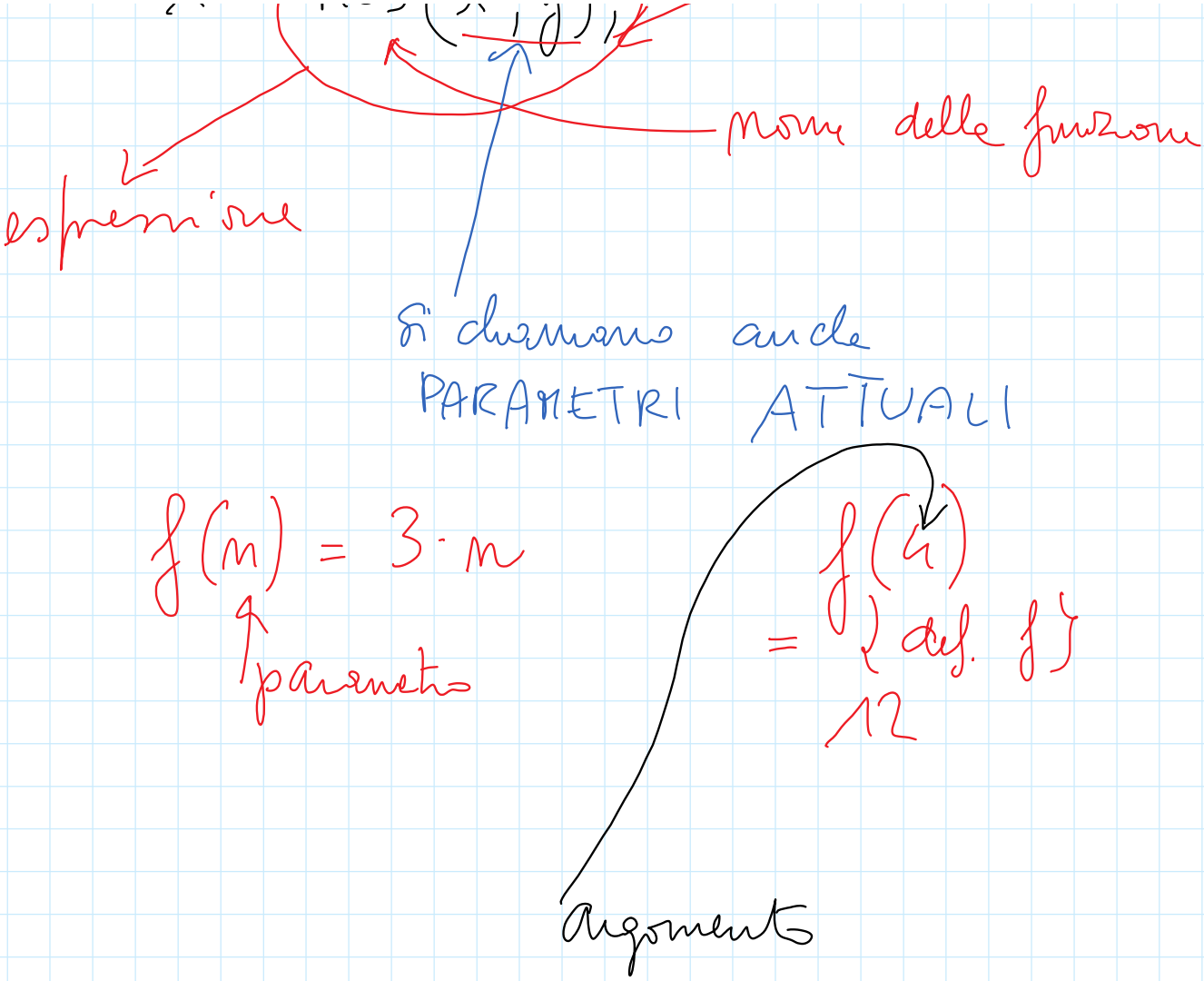
si chiamano anche PARAMETRI FORMALI

```
...
main()
{ int x, y;
  int z, w;
  scanf("%d %d", &x, &y);
  scanf("%d %d", &z, &w);
  x = MCD(x, y);
```

int x; int y;

int x, y, z, w;

argomenti


nome della funzione
espressione
si chiamano anche
PARAMETRI ATTUALI
f(n) = 3 · n
parametro
f(4)
= { def. f }
12
argomento

```
int MCD (int n, int m)
{ while (n != m)
      if (n > m)
          n = n - m;
      else
          m = m - n;
  return n;
}

main()
  ⋮
  x = MCD(x, y);
```

6 21 — 6 6 21 — 21 21 6

Non ha significato perché l'argom. dove mod. il valore di una VARIABILE

21 = 21 - 6

Supponiamo che x valga 6 e y 21.

Il collegamento tra argomenti e parametri di una funzione (collegamento tra parametri attuali e parametri formali) si chiama:

MODALITA' DI PASSAGGIO DEI PARAMETRI

Nel C (come in molti linguaggi imperativi attuali) esiste una sola modalità di passaggio dei parametri:

MODALITA' PER VALORE

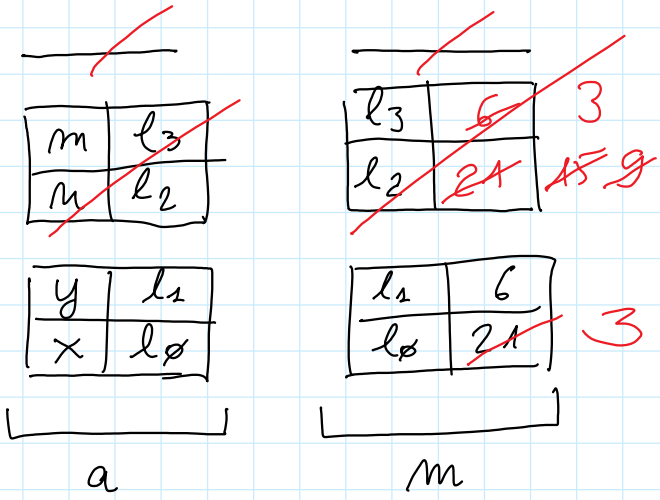
Nella modalità di passaggio dei parametri per valore si effettuano le seguenti operazioni:

- Quando una funzione (o procedura) viene chiamata si crea un nuovo frame di ambiente e di memoria per contenere nuove variabili con il nome dei parametri (formali)
  I parametri diventano nuove variabili
- A queste variabili viene dato inizialmente il VALORE degli argomenti (parametri attuali)
- Viene eseguito il blocco che definisce la funzione (o procedura)
- Viene cancellato dalla pila (pop) il frame con le variabili corrispondenti ai parametri (formali)

```
int MCD (int n, int m)
{ while (n != m)
    if (n > m) n = n - m;
    else m = m - n;
  return n;
}
```

```
main()
{ int x = 21;
  int y = 6;
  x = MCD(x, y);
}
```

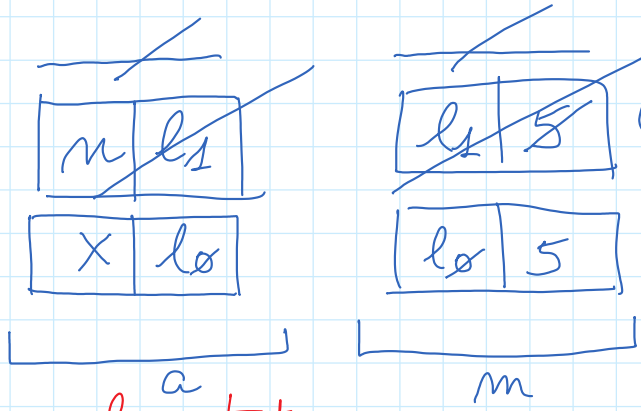
risultato = 3



lunedì 23 settembre 2019 10:26

non ha tipo del risultato

```
void incr (int n)
{ n = n + 1;
}

main()
{ int x = 5;
  incr(x);
}
```



5

Modifica lo stato e non restituisce un valore.

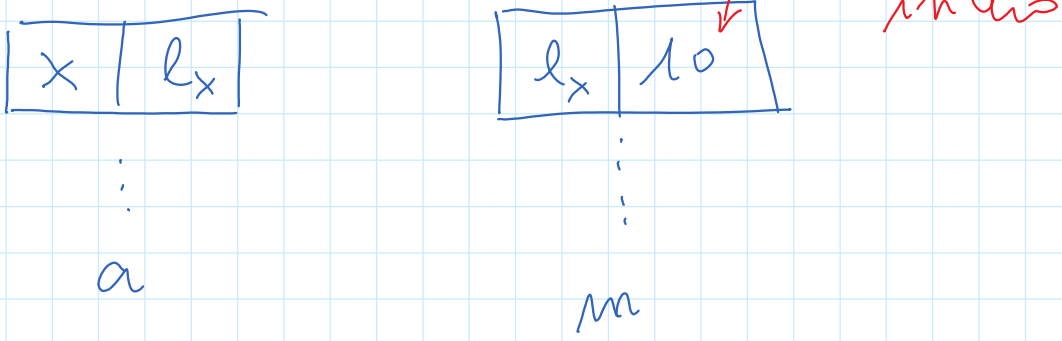
(chiamata di procedura è un COMANDO)

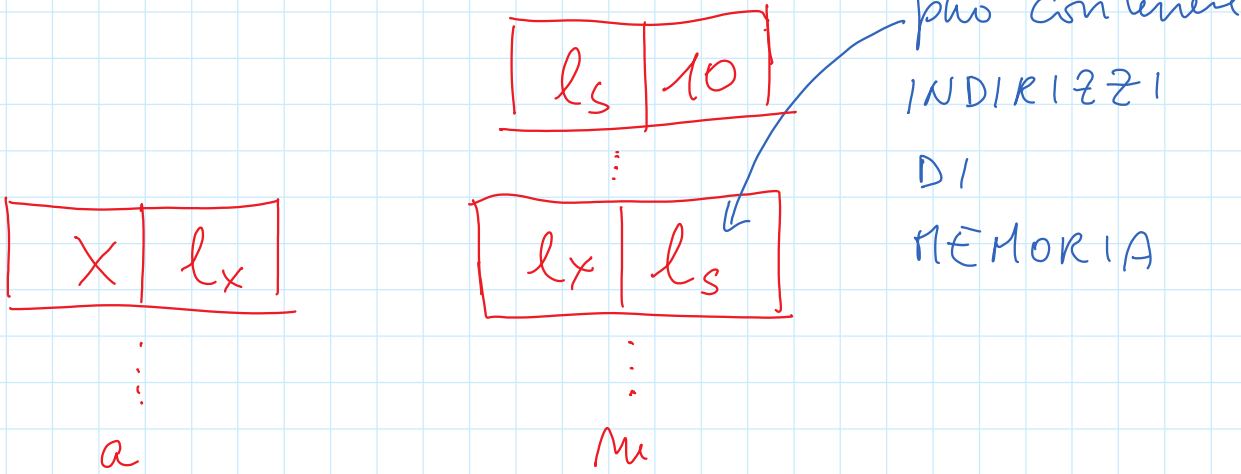
```
x = incr(x);
```

concettualmente sbagliata!!

VARIABILI PUNTATORE

Le variabili hanno un valore in memoria





Il valore di x, che si trova alla locazione l_x, è un indirizzo di memoria.

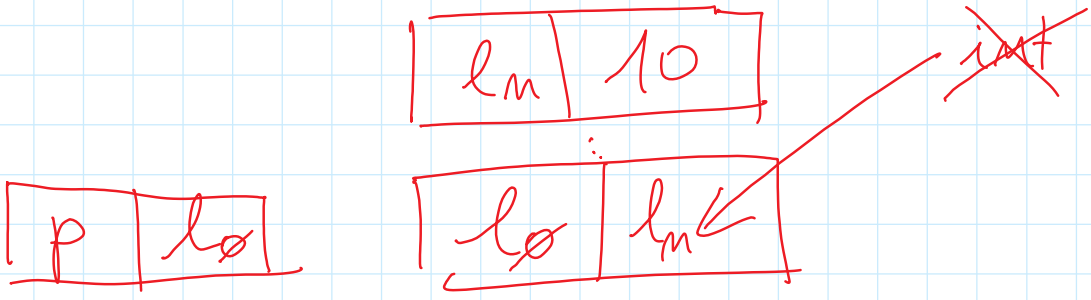
Le variabile che hanno come valore in memoria un indirizzo si chiamano VARIABILI PUNTATORE

Dichiarazione di una variabile puntatore:

tipo * nome;

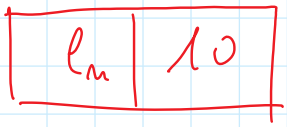
int *p;

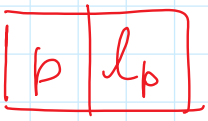
il valore di p in memoria non sarà un valore intero, ma sarà l'indirizzo in memoria di un valore intero

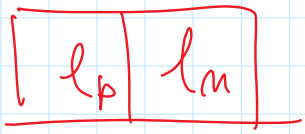


(*p)

non è il valore di p, ma il valore PUNTATO da p, cioè il valore contenuto nella locazione che è valore di p (l_m)








```
int *p;
```

Qual'è il valore dell'espressione

p → l_m ← valore di p

*p → 10 ← valore puntato da p

&p → l_p l'indirizzo in memoria che contiene il valore di p

```
{ int x = 5;
  int *p = &x;
  x = x + 1;
  *p = *p + 1;
}
```

corretto!!

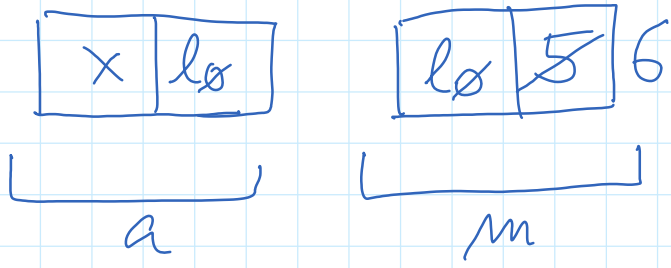
p	l_1
x	$l_\emptyset$

a

l_1	$l_\emptyset$
$l_\emptyset$	~~5~~ 6 7

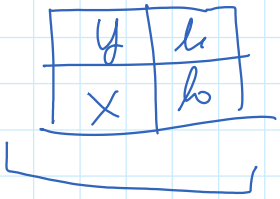
m

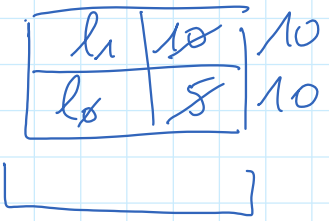
```
void inc (int * n)
{
    *n = *n + 1;
}

main()
{
    int x = 5;
    inc (&x);
}
```



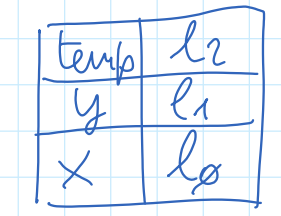
```
{ int x = 5;
  int y = 10;
  x = y;
  y = x;
```

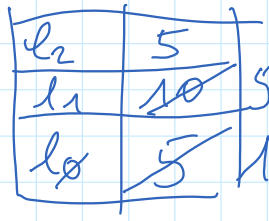
No!






```
{ int x = 5;
  int y = 10;
  int temp = x;
  x = y;
  y = temp;
```





a m