corsi di laurea in Ingegneria Chimica,Elettronica,Telecomunicazioni,Informatica

Fondamenti di Informatica: complemento didattico su Attivazione di funzione e record di attivazione

Dispensa a puntate: questa è la prima puntata. Comment e correzioni sono ben accetti. Quando sarà possibile e utile, ci saranno altre puntate ... mt

1. Engage (Attivazione)

Sia data la seguente funzione che, ricevendo due numeri interi, `n` ed `m`, restituisce il massimo comun divisore tra `n` ed `m`,

```
int mcd(int n, int m) {
  int risultato;

  while (n!=m)
     if (m>n)
       m=m-n;
     else n-=m;

  risultato = n;   /* (o m ...) */

return risultato;
}
```

in essa riconosciamo

- i **parametri formali**: `n`, `m`;
- una **variabile locale**: `risultato`;
- il **valore di ritorno** (valore prodotto dalla funzione e "restituito" in luogo della sua chiamata), che in questo caso è la valutazioone dell'espressione "`risultato`")

Il programma successivo, usa la funzione per calcolare l'mcd di una sequenza di coppie di numeri dati in input (vedi file `mcd.c`):

```
#include <stdio.h>

int mcd(int, int);            /* dichiarazione della funzione mcd */

int main ()
{
  int primo,secondo;            /* i numeri di cui trovare il MCD */

                                          /* lettura prima coppia */
  printf("fornire i primi due numeri: ");
  scanf("%d %d", &primo, &secondo);

  while ( (primo!=0) && (secondo!=0) ) {
    printf("mcd tra %d e %d e' %d\n", primo,secondo, mcd(primo,secondo));

                /* lettura coppia successiva a quella appena gestita */
    printf("fornire due numeri (se uno e' 0, termino): ");
    scanf("%d %d", &primo, &secondo);
  } /* fine while */
```

```
printf("FINE\n");
return 0;
}
```

Come si vede, a seconda delle sequenze di input, la funzione `mcd` viene chiamata più o meno numerose volte; ogni chiamata ha l'aspetto seguente:

```
... mcd(primo,secondo) ...
```

dove i puntini simboleggiano l'ambiente in cui la chiamata è immersa (una `printf`), che in questo momento non ci interessa.

Come in ogni chiamata, c'è una *funzione chiamante* (`main`, nel nostro esempio) e c'è una *funzione chiamata* (`mcd`). Inoltre, le espressioni inserite tra le parentesi della chiamata costituiscono i **parametri attuali**, (`primo` e `secondo` nel nostro caso) corrispondenti ai valori che verranno usati come parametri nell'esecuzione della funzione.

Ogni chiamata di funzione viene gestita, come è noto, secondo il seguente schema:

1) l'esecuzione della funzione chiamante viene interrotta;
2) la funzione chiamata viene attivata; ciò corrisponde a
    a. assegnare ai parametri formali i valori dei parametri attuali, cioè eseguire il passaggio dei parametri
    b. eseguire il codice della funzione, in cui i simboli associati ai parametri formali sono ora variabili che contengono i valori assegnati con il passaggio dei parametri.
3) Quando il codice della funzione giunge al termine, il risultato (cioè il valore specificato nella `return()`), viene fornito alla funzione chiamante (si dice anche "ritornato", con una traduzione sportiva)
4) la funzione chiamante viene riattivata e la sua esecuzione può proseguire, trovando il valore ritornato dalla chiamata nel punto esatto della chiamata.

In altre parole, supponendo di star eseguendo il programma (cioè la funzione `main()`) e di stare per eseguire l'istruzione

```
printf("mcd tra %d e %d e' %d\n", primo,secondo, mcd(primo,secondo));
```

e supponendo che i valori assegnati in `primo` e `secondo` siano, rispettivamente 15 e 5, accade che

- prima di eseguire la `printf`[*] i suoi argomenti vengono valutati e per valutare `mcd(primo,secondo)` si esegue la chiamata di `mcd` con parametri attuali `primo`=15 e `secondo`=5;
- l'attivazione termina ritornando il valore 5,
- la `main` torna in esecuzione da dove si era interrotta e utilizza il 5 come terzo valore da stampare nella `printf`.

2. Il record di attivazione

Quel che ci importa in questo complemento didattico è il fatto che l'esecuzione del codice della funzione `mcd` avviene in realtà all'interno di una zona riservata di memoria, nella quale trovano spazio, tra l'altro, le variabili locali e i parametri formali da usare durante l'attivazione della funzione. Questa zona di memoria viene occupata (*allocata*) come primo atto dell'attivazione della funzione; si tratta di memoria che precedentemente era libera, adesso è riservata e tornerà libera non appena l'attivazione sarà terminata (perché, dopo il che la funzione chiamante ha ricevuto il risultato, non serve più).

[*] che in effetti è un'altra chiamata di funzione ...

Questa zona di memoria, si chiama **record di attivazione**: essa

- viene allocata per eseguire un'attivazione di funzione,
- è destinata a contenere le locazioni di memoria necessarie per le variabili locali e per i parametri formali della funzione durante la sua esecuzione
- viene deallocata quando l'attivazione è terminata.

Cosa c'è nel record di attivazione? Sostanzialmente tre cose:

- una sezione in cui sono allocate le locazioni di memoria necessarie per i parametri formali: queste locazioni vengono riempite al momento del passaggio dei parametri, con i valori passati come parametri attuali;
- una sezione in cui sono allocate le locazioni di memoria necessarie per le variabili locali definite nella funzione
- un'indicazione del punto di ritorno, cioè del punto in cui la funzione chiamante era stata interrotta (per attivare la funzione chiamata), dal quale riprenderà l'esecuzione dal ritorno dalla chiamata.

Questo blocco di memoria viene allocato al momento della chiamata e scompare al; termine della chiamata medesima; cioè per ogni chiamata viene allocato (e deallocato) un record di attivazione dedicato; per cui le variabili e i parametri di una funzione hanno un cosiddetto *ciclo di vita* limitato al tempo dell'attivazione della funzione (salvo eccezioni): esistono solo durante l'attivazione; non esistono al di fuori dell'attivazione.

In Fig.1 e nelle successive, si vede come si presenta la memoria centrale durante l'esecuzione del programma d'esempio visto prima.

In effetti anche la `main` è una funzione ... e qualcuno l'avrà pure attivata ... ma per essa rinunciamo a mostrare il record di attivazione (sarebbe degenere e poco istruttivo in questo momento).

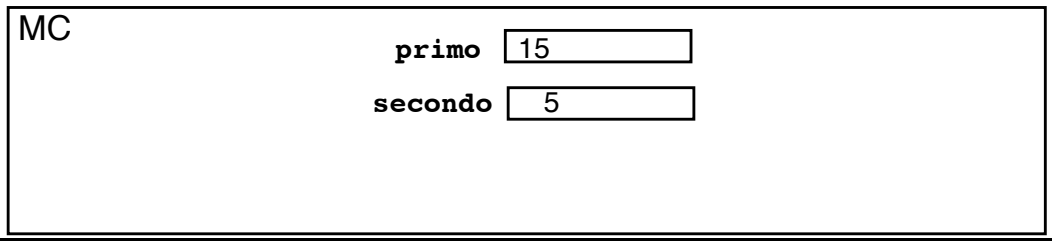
Quel che è certo è che stiamo eseguendo il codice della `main`: supponiamo di aver già eseguito le istruzioni di lettura e di trovarci nella situazione di **Fig.1** (`primo` e `secondo` sono state assegnate e contengono 15 e 5); adesso avviene la chiamata alla funzione mcd: si tratta della chiamata `mcd(primo, secondo)`; lo schema di funzionamento è brevemente richiamato subito qui sotto e amplia quello visto precedentemente (una discussione più prolissa, applicata al caso del nostro esempio, è in **Tab. 1**):

1) l'esecuzione della funzione chiamante viene interrotta;
2) la funzione chiamata viene attivata; ciò corrisponde a
   i. allocare un record di attivazione, in cui l'attivazione verrà eseguita;
   ii. eseguire il passaggio dei parametri (copia dei valori dei parametri attuali nelle locazioni dei parametri formali;
   iii. eseguire il codice della funzione.
3) Quando l'esecuzione del codice termina,
   i. il risultato (cioè il valore specificato nella `return()`), viene fornito alla funzione chiamante;
   ii. il record di attivazione viene deallocato
4) la funzione chiamante viene riattivata e la sua esecuzione può proseguire, trovando il valore ritornato dalla chiamata nel punto esatto della chiamata.

1) l'esecuzione della funzione chiamante viene interrotta;	la main() si ferma, in attesa di avere il risultato di mcd(primo, secondo)
2) la funzione chiamata viene attivata; ciò corrisponde a i. allocare un record di attivazione in cui la sezione VAR contiene le (locazioni per le) variabili locali, la sezione PAR contiene (le locazioni per) i parametri formali e il punto di ritorno è l'indirizzo del punto nel codice della funzione chiamante dove verrà ripresa l'esecuzione di quest'ultima; ii. eseguire il passaggio dei parametri: i valori dei parametri attuali vengono copiati nelle locazioni dei parametri formali; iii. eseguire il codice della funzione	l'attivazione comporta i. **allocazione del record di attivazione (Fig.2)** nella sezione PAR ci sono due locazioni, associate ai simboli usati per i parametri formali (n ed m); nella sezione VAR c'è una sola locazione, associata alla variabile locale risultato; il punto di ritorno è il punto della chiamata alla funzione `mcd`, nella main; ii. il **passaggio dei parametri (Fig.3)** consiste nel valutare le espressioni `primo e secondo` (i due *parametri attuali*) e nel copiare i loro valori (15 e 5) nelle locazioni dei parametri formali (`n` ed `m`): si segue l'ordine di definizione dei parametri formali e di passaggio dei parametri attuali; il primo parametro attuale viene passato al primo formale e così via... iii. **l'esecuzione del codice della funzione**: - in **Fig.4** si vede cosa succede dopo la prima iterazione del while: `n` è diventato 10; - in **Fig.5** si vede cosa succede dopo la seconda iterazione del while: `n` è diventato 5; - in **Fig.6** si vede che il ciclo termina (`n==m`) e la variabile `risultato` è stata assegnata a 5*. Poi termina anche il l'attivazione, con `return(risultato)`.
3) Quando il codice della funzione giunge al termine, i. il risultato (cioè il valore specificato nella `return()`), viene fornito alla funzione chiamante ii. il record di attivazione viene deallocato	insomma, come si vede in **Fig.7** i. il 5 viene ritornato alla `main`, ii. il record di allocazione scompare
4) la funzione chiamante viene riattivata e la sua esecuzione può proseguire, trovando il valore ritornato dalla chiamata nel punto esatto della chiamata.	la `main` torna in esecuzione; il 5 ritornato dalla chiamata di `mcd` è disponibile, nel punto in cui `mcd` era stata chiamata (quindi il terzo valore stampato dalla `printf` sarà 5)

Tabella 1 : schema di attivazione (sin.) e sua applicazione al caso dell'esempio (des.)

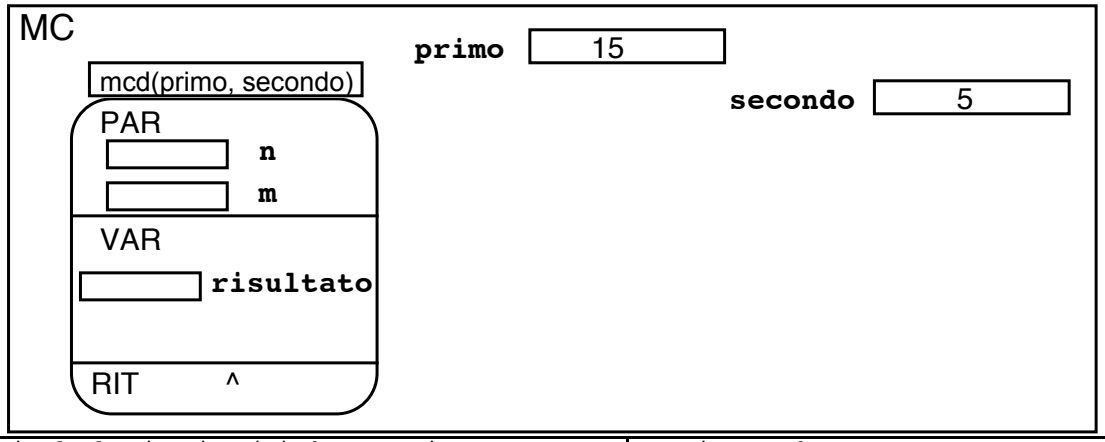
* sarebbe stato perfettamente plausibile che la funzione restituisse quin il valore di n (o di m ...): la variabile locale risultato è in effetti ridondante; l'abbiamo usata solo a scopo di esempio, per avere una qualsiasi variabile locale da usare nei nostri esempi ...




```
...
printf("fornire i primi due numeri: ");
  scanf("%d %d", &primo, &secondo);

  while ( (primo!=0) && (secondo!=0) ) {
    printf("mcd tra %d e %d e' %d\n", primo,secondo, mcd(primo,secondo));

                  /* lettura coppia successiva a quella appena gestita */
    printf("fornire due numeri (se uno e' 0, termino): ");
    scanf("%d %d", &primo, &secondo);
  } /* fine while */

printf("FINE\n");
return 0;
```

Figura 1 la main() è in esecuzione: le due variabili sono state assegnate con una scanf

==========




```
   printf("fornire i primi due numeri: ");
     scanf("%d %d", &primo, &secondo);

     while ( (primo!=0) && (secondo!=0) ) {
       printf("mcd  tra  %d  e  %d  e'  %d\n",
                                          ^
       primo,secondo, mcd(primo,secondo));

       printf("fornire due numeri (se uno e' 0,
       termino): ");
       scanf("%d %d", &primo, &secondo);
     } /* fine while */
...
```

```
Funzione mcd

int mcd(int n, int m) {
  int risultato;

  while (n!=m)
     if (m>n)
       m=m-n;
     else n-=m;

  risultato = n;
return risultato;    }
```

Figura 2 main() è stata interrotta; il record di attivazione poer la chiamata `mcd(primo, secondo)` è stato allocato; il punto di ritorno nella main corrisponde alla printf

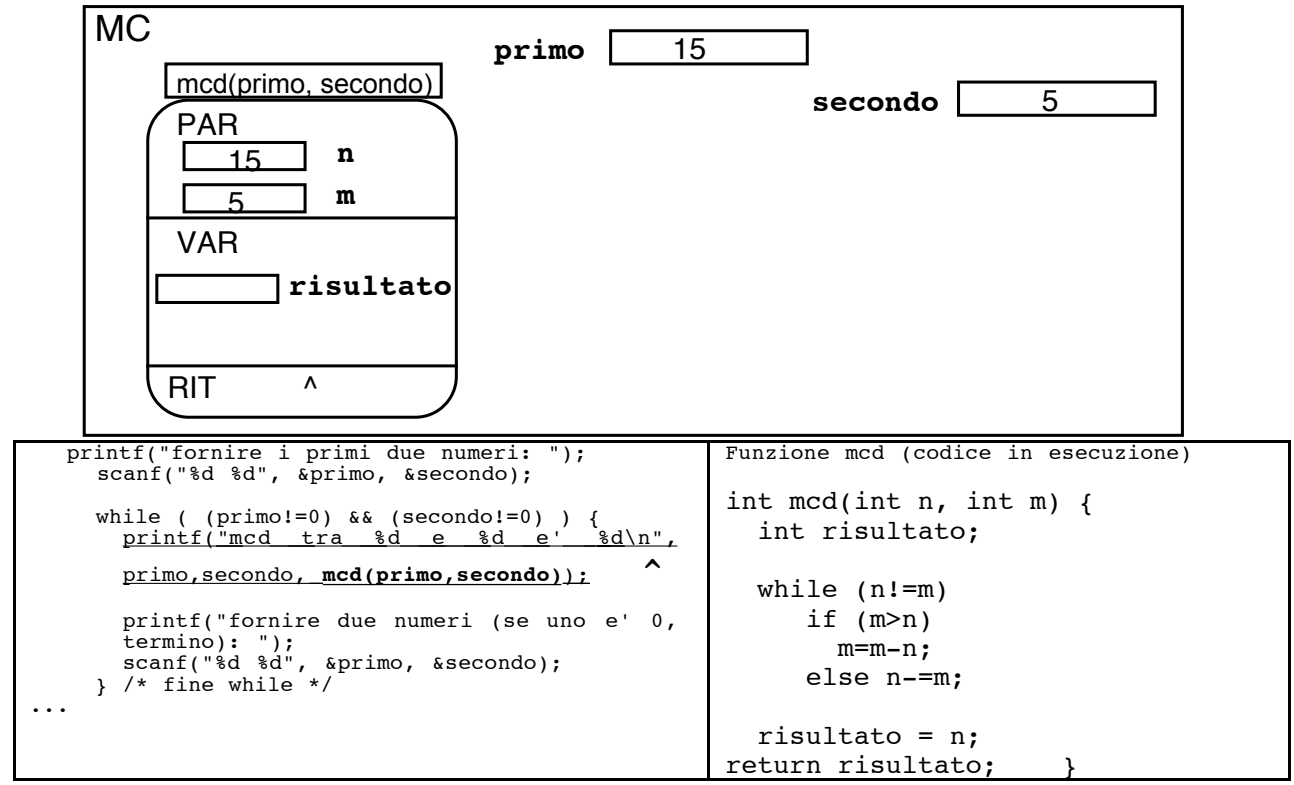


Figura 3 è stato effettuato il passaggio dei parametri: il primo parametro attuale è un'espressione (`primo`) che valuta a 15; quindi il 15 viene copiato nel parametro formale `n` nel record di attivazione; analogamente il secondo parametro attuale (che vale 5) viene copiato in `m`

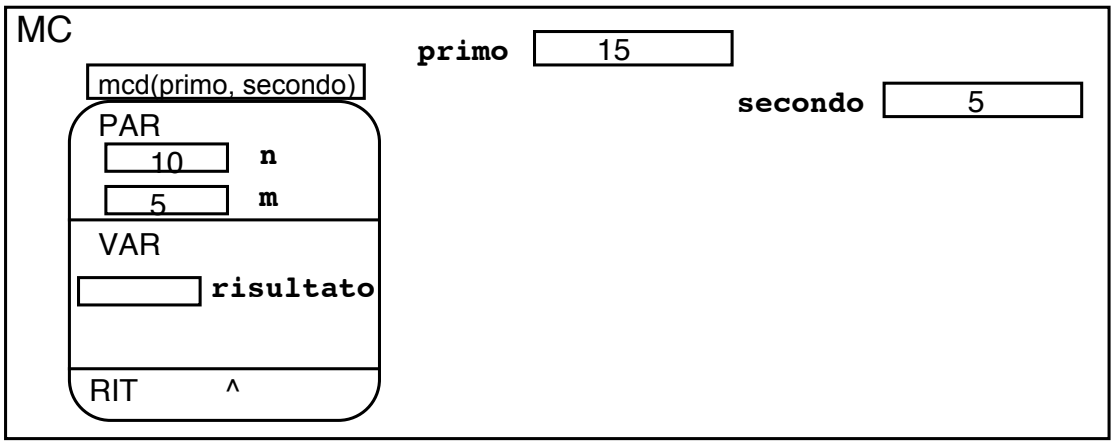


Figura 4 la funzione è in esecuzione: dopo la prima iterazione del while `n` è diventato 10

==========

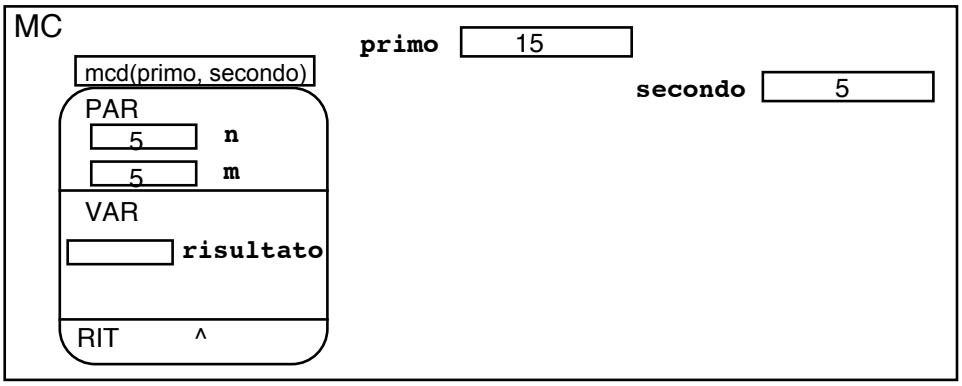


Figura 5 dopo la seconda iterazione del while n è diventato 5

==========

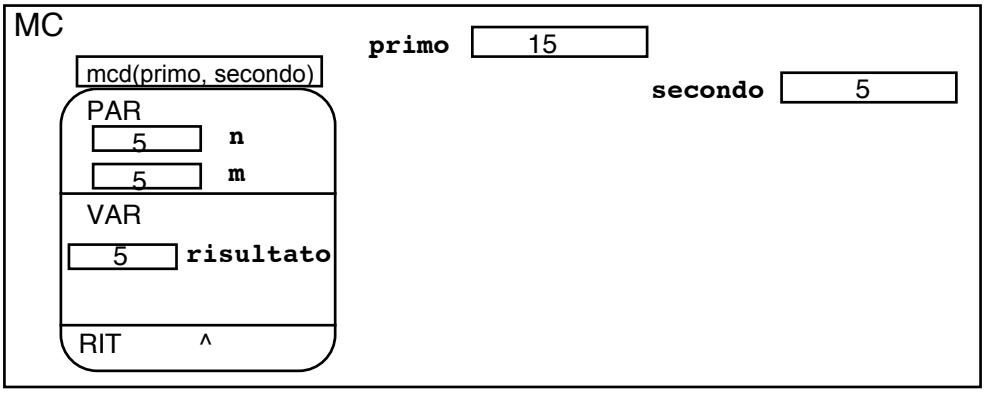
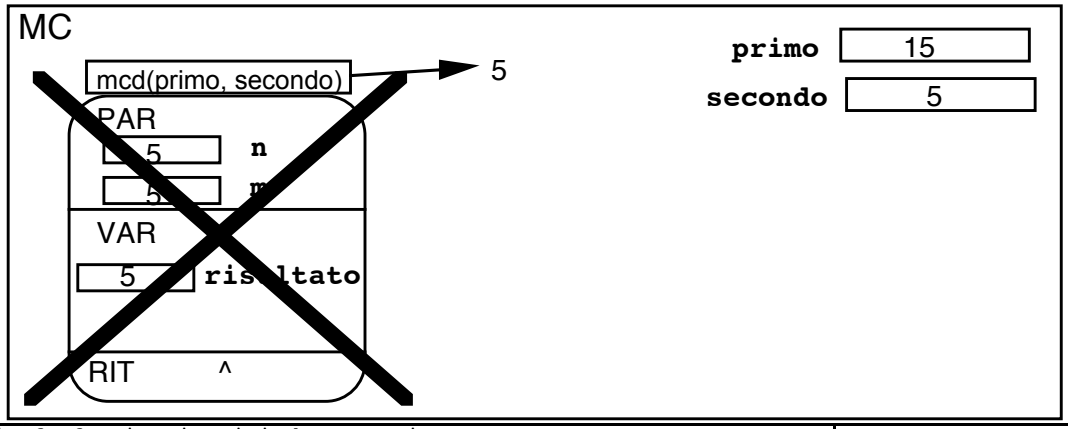


Figura 6 il ciclo termina e la variabile risultato viene assegnata

==========




```
   printf("fornire i primi due numeri: ");
     scanf("%d %d", &primo, &secondo);

     while ( (primo!=0) && (secondo!=0) ) {
       printf("mcd tra %d e %d e' %d\n", primo,secondo,
       mcd(primo,secondo));
                    ^
       printf("fornire due numeri (se uno e' 0, termino): ");
       scanf("%d %d", &primo, &secondo);
     } /* fine while */
...
```

Figura 7 con return(risultato) il valore 5 viene "ritornato" alla funzione chiamante (indicato dalla freccia) come risultato della chiamata: il 5 verrà usato dalla main in corrispondenza del punto di ritorno; il record di attivazione non serve più e viene deallocato

==========

Il fatto che `primo` e `secondo` (i parametri attuali della chiamata di `mcd()`) mantengano il loro valore (e non cambino come fanno `n` ed `m`) è una conseguenza delle modalità del passaggio dei parametri nelle funzioni C.

3. Passaggio di indirizzo

Alla prossima puntata ...