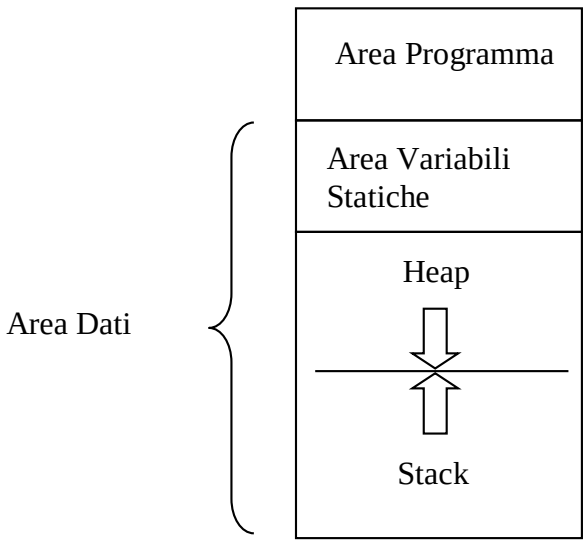
Parametri Formali di una Funzione e Record di Attivazione

Un record di attivazione rappresenta il "mondo" di una funzione, e contiene tutto ciò che ne caratterizza l'esistenza, tra cui ad esempio:

- le variabili locali
- i parametri formali
- il valore di ritorno (nel caso sia presente nella funzione)

Ad ogni attivazione di una funzione viene creato un nuovo record d'attivazione specifico per quella chiamata.

L'area di memoria dedicata alla memorizzazione dei record di attivazione è l'Area Stack. Si ricorda che a ciascun programma utente viene assegnata una porzione di memoria, che è automaticamente divisa in due parti; l'area del codice contiene il codice del programma, mentre l'area dati contiene: le variabili statiche, l'area heap disponibile per allocazioni dinamiche, e l'area stack che contiene i record di attivazione delle funzioni. La figura seguente mostra la divisione dell'area assegnata all'utente. Come è visibile dalla figura, le dimensioni delle aree programma e della porzione dell'area dati corrispondente alle variabili statiche sono fisse e sono decise in fase di compilazione. Infatti tali dimensioni corrispondono al numero di righe comando del programma e al numero di variabili statiche definite in esso. Le altre due aree (Heap e Stack) non hanno dimensione fissa, ma l'area complessiva (pari alla somma delle due) è anch'essa fissa. In particolare ciascuna area (Heap e Stack) ha un limite massimo entro cui poter crescere.



Il fatto che l'Heap e lo Stack non abbiano dimensione fissa significa che tale dimensione varia nel tempo a seconda dell'utilizzo delle due aree. In particolare per quanto riguarda l'area Stack, essa crescerà ogni qual volta una funzione viene chiamata e il suo relativo record di attivazione viene memorizzato nello stack. La sua dimensione si ridurrà, invece, quando una funzione termina, determinando la distruzione del relativo record di attivazione (in effetti l'area di memoria occupata dal record di attivazione viene solamente rilasciata).

L'inserimento di ciascun record di attivazione nello stack avviene secondo la politica LIFO (Last In First Out- Ultimo Arrivato Primo a Essere Estratto). In altri termini l'inserimento di ciascun record di attivazione è tale che il primo record di attivazione presente nello stack è relativo all'ultima funzione chiamata ed in corso di esecuzione.

In base a quanto detto, è possibile riassumere i seguenti concetti.

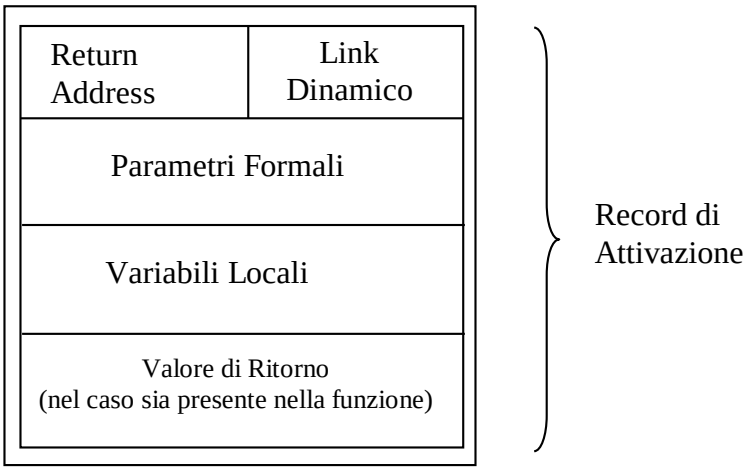
1. La dimensione del record di attivazione:
    - varia da una funzione all'altra
    - ma, per una data funzione, è fissa e calcolabile a priori.
2. Il record di attivazione:
    - viene creato nello stack dinamicamente nel momento in cui la funzione viene chiamata
    - rimane nello stack per tutto il tempo in cui la funzione è in esecuzione
    - viene deallocato (rilasciato) alla fine quando la funzione termina.
3. Funzioni che chiamano altre funzioni danno luogo a una sequenza di record di attivazione:
    - allocati secondo l'ordine delle chiamate
    - deallocati in ordine inverso

Quando la funzione termina, il controllo torna al chiamante, che deve riprendere la sua esecuzione dall'istruzione successiva alla chiamata della funzione e trovare il suo "mondo" inalterato. Per "mondo" si intende l'insieme delle informazioni contenute nel record di attivazione dello stesso chiamante (ad esempio variabili locali e parametri formali). Al fine di permettere al programma chiamante di riprendere dalla prima istruzione successiva alla chiamata a funzione e di recuperare l'intero record di

attivazione, dopo la conclusione della funzione chiamata, si inseriscono nel record di attivazione della funzione anche:

- l'**indirizzo di ritorno**, ossia l'indirizzo della prossima istruzione del chiamante che andrà eseguita quando la funzione termina;
- il **link dinamico**, ossia l'indirizzo di memoria del record di attivazione del chiamante, in modo da poter ripristinare l'ambiente del chiamante quando la funzione termina.

La seguente figura mostra tutti i contenuti presenti tipicamente in un record di attivazione.



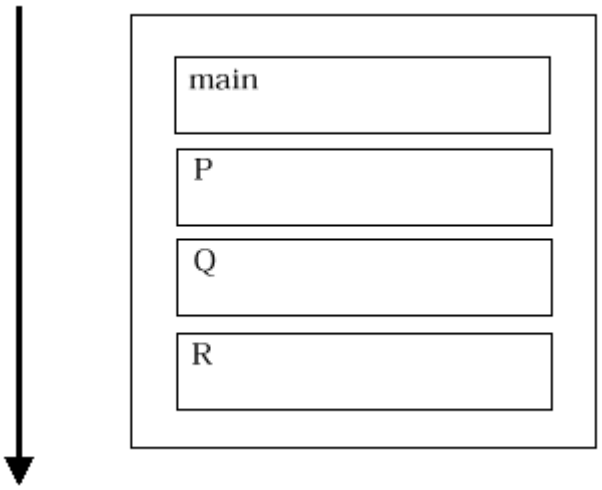
La sequenza dei link dinamici costituisce la cosiddetta **catena dinamica**, che rappresenta la storia delle attivazioni ("chi ha chiamato chi"). Per le **funzioni** che tornano un valore, il record di attivazione prevede anche un'ulteriore cella, destinata a contenere il risultato della funzione. Tale risultato viene copiato dal chiamante prima (ovviamente!) di deallocare il record di attivazione della funzione appena terminata. Altre volte, il risultato viene restituito dalla funzione al chiamante semplicemente lasciandolo in un registro della CPU.

Attenzione: è importante ricordare che anche il main è una funzione e per tanto anche per essa viene creato un record di attivazione. Ovviamente esso sarà distrutto solo alla conclusione del programma stesso.

Si consideri il seguente programma:

```
#include <stdio.h>

int x;

void R (int a) {
      x=a;
}

void Q (int x) {
      R(x);
}

void P() {
      int a;

      a=10;
      Q(a);
}

int main(void)
{
      P();
}
```

L'esecuzione del main comporterà la creazione del record di attivazione del main. Inoltre verranno creati i record di attivazione delle funzioni P, Q, R nell'ordine, come si evince dalla seguente figura. Infatti il main chiama la funzione P. Viene così creato il record di attivazione della funzione P. la funzione P chiama a sua volta la funzione Q, che per poter concludersi deve chiamare la funzione R. Ciò comporta che il primo record nello stack è R, ossia l'ultimo ad essere stato inserito. I record di attivazione verranno eliminati nell'ordine inverso, ossia R, Q, P e main. Infatti non appena la funzione R viene completata, anche la funzione Q viene completata. Infine anche P termina, determinando la conclusione dello stesso programma main.


main
P
Q
R

1 Passaggio dei Parametri

Scopo di questa sezione è quello di comprendere come i parametri di una funzione sono passati nel record di attivazione. Possono esistere: Parametri Passati per Valore e per Indirizzo (Reference o Riferimento). I primi sono anche chiamati parametri formali valore, mentre gli altri sono conosciuti come parametri formali variabile.

1.1 Parametri Passati per Valore.

Nel caso di parametri formali valore, nella cella del record di attivazione corrispondente al parametro formale, viene copiato il valore assunto dal parametro attuale all'atto della chiamata. La corrispondenza tra parametri formali ed attuali è di tipo posizionale.

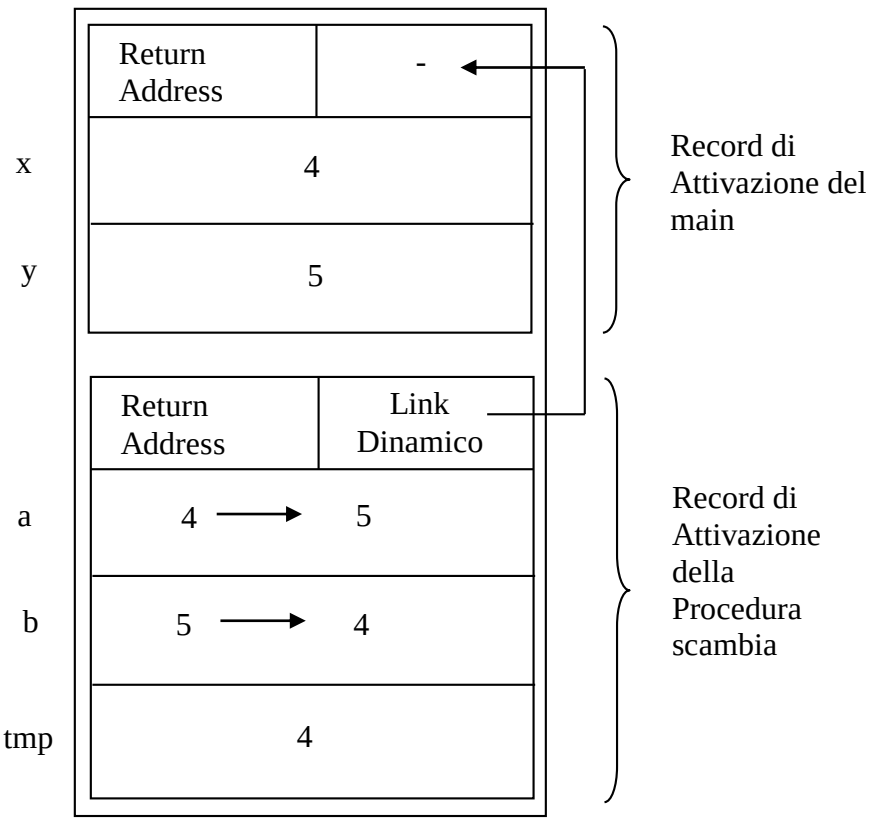
Si consideri il seguente esempio:

```
#include <stdio.h>

void scambia ( int a, int b)
{     int tmp;

      tmp=a;
      a=b;
      b=tmp;
}

int main(void){
      int x, y;

      x=4;
      y=5;
      scambia(x,y);
      printf("%d %d \n", x, y);
}
```

Quando il main viene eseguito verrà allocato il relativo record di attivazione, composto dalle variabili locali x e y (inizializzate a 4 e 5). In tal caso l'Address Return è relativo al Sistema Operativo che ha chiamato il main. Inoltre manca il link dinamico, che in questo caso non è necessario perché la conclusione del main corrisponde alla conclusione dell'intero programma e dunque non è necessario ripristinare alcun record

di attivazione precedente. Infine mancano i parametri formali, visto che sono assenti nella chiamata *int main(void)*.

Il main chiama la procedura scambia, passando i parametri attuali x e y. Ciò comporta la creazione del relativo record di attivazione. Esso contiene: l'indirizzo di ritorno all'istruzione che il main dovrà eseguire dopo la conclusione della procedura scambia (tale istruzione è printf("%d %d \n", x,y);), il link dinamico al record di attivazione del main, la variabile locale tmp e i parametri formali a e b. I valori assunti da a e b coincidono con i valori dei corrispondenti parametri attuali passati alla procedura ossia il valore di x, cioè 4 e il valore di y, ossia 5. La procedura scambia inizializza tmp=4 e modifica il valore di a e b, che diventano 5 e 4, rispettivamente. Successivamente, la procedura scambia termina ritornando al chiamante, ossia al main. Tutto il record di attivazione della procedura scambia viene disallocato. Il ritorno al main comporta il ripristino del record di attivazione del main, ossia delle variabili x e y, che conservano i loro valori prima della chiamata alla procedura. La seguente figura mostra ciò che accade ai record di attivazione del main e della procedura scambia. Come si vede i valori di x e y non vengono modificati, mentre i parametri formali a e b vengono scambiati nella procedura e poi deallocati una volta che la procedura si conclude.



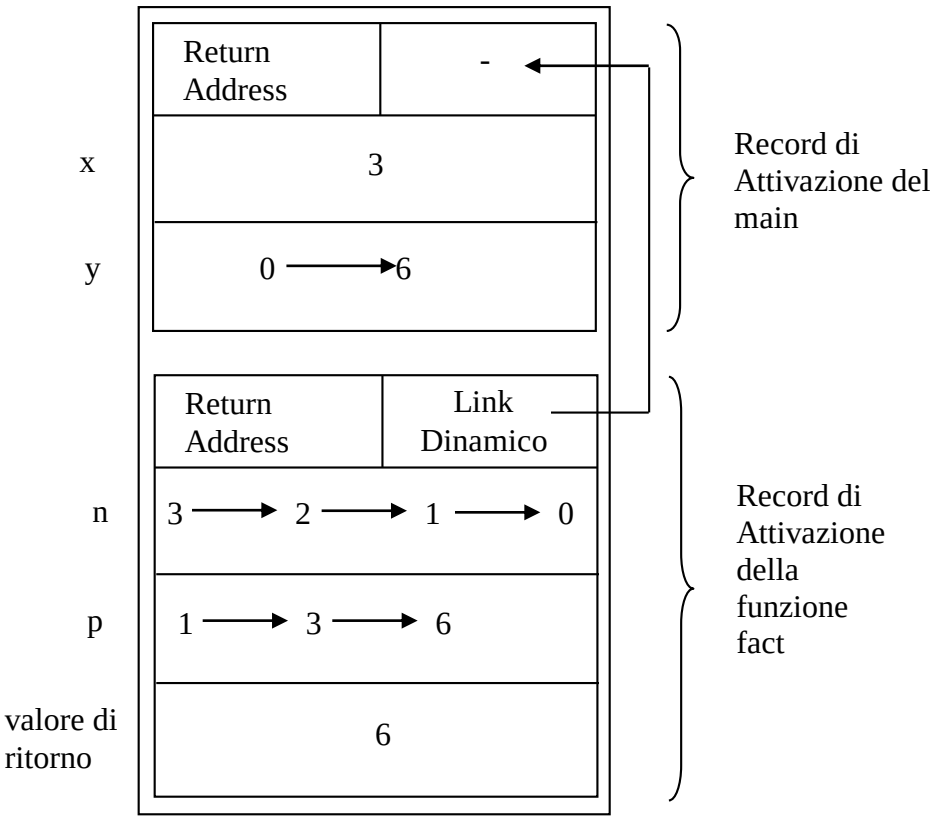
Si consideri questo altro programma:

```
#include <stdio.h>

int fact ( int n)
{     int p;

      p=1;
      while (n>0) p*=n--;
      return p;
}

int main(void)
{
      int x, y;

      x=3;
      y=fact(x);
}
```

Quando il main viene eseguito verrà allocato il relativo record di attivazione, composto dalle variabili locali x e y. In tal caso l'Address return è relativo al Sistema Operativo che ha chiamato il main. Inoltre manca il link dinamico, che in questo caso non è necessario, come spiegato in precedenza. Infine mancano i parametri formali del main.

Il main chiama la funzione fact. Ciò comporta la creazione del relativo record di attivazione. Esso contiene: l'indirizzo di ritorno all'istruzione che il main dovrà eseguire dopo la conclusione della funzione fact (tale istruzione è y=valore di ritorno della funzione fact(x)), il link dinamico al record di attivazione del main, la variabile locale p, il parametro formale n e la locazione di memoria dove verrà messo il valore di ritorno che la funzione restituirà al main. Il valore di n coincide con il valore del parametro attuale passato alla funzione ossia il valore di x, cioè 3. La funzione fact inizializza p=1 e attraverso il ciclo while modifica il valore di n, che inizialmente è 3 e che viene decrementato fino a diventare 0. Contemporaneamente il valore p viene ottenuto dal prodotto p*=n. Quando il ciclo while termina, il valore di n è 0 e p vale 6. Dopo la conclusione del ciclo while, la funzione fact termina ritornando il valore di p, al chiamante, ossia al main. Come detto, il valore di ritorno verrà messo in una locazione del record di attivazione della funzione fact. Tale valore verrà letto dal main, poco prima che il record di attivazione della funzione fact venga distrutto.

.



Il ritorno al main comporta il ripristino del record di attivazione del main, ossia della variabile x, che conserva il suo valore prima della chiamata alla funzione. Tale ripristino può essere realizzato grazie al link dinamico contenuto nel record di attivazione della funzione fact.

1.1.1 Parametri Formali Valore di tipo vettore

Scopo di questa sezione è quello di descrivere il passaggio di parametri nel caso in cui uno dei parametri formali è rappresentato da un vettori.

Si ricorda che, in linguaggio C i parametri formali passati per valore relativamente ad un vettore (**anche per quello dinamico, creato con malloc**), si esprimono nelle tre forme equivalenti:

- tipobase * nome_parametro;
- tipobase nome_parametro[];
- tipobase nome_parametro[dimensione];

Come detto, tale regola vale sia nel caso in cui il vettore è definito secondo la sintassi del linguaggio C:

tipobase nome_vettore[dimensione],

sia nel caso in cui il vettore sia stato allocato in modo dinamico nell'heap, tramite il comando:

nome_vettore=(cast)malloc(dimensione vettore);

Ad esempio si consideri il seguente programma:

```
#include<stdio.h>
#define N 5
#define M 10

float vett1[N], vett2[M];

void riempi(float v[], int dim){
   unsigned long i;

   for (i=0; i<dim; i++) {
        printf("Inserisci l'elemento di indice %u ",i);
        scanf("%f",v+i); //oppure scanf("%f",&v[i]);
   }
}

void stampa(float v[N], int dim){
   unsigned long i;

   for (i=0; i<dim; i++)
        printf("\nElemento di indice %u = %f ",i,v[i]);
}

float max(float *v, int dim){
   unsigned long i;
   float max=v[0];

   for (i=1; i<dim; i++)
        if (v[i]>max) max=v[i];

   return max;
}

int main(void)
{
  riempi(vett1,N);
  stampa(vett1,N);
  printf("\nMassimo elemento = %f ",max(vett1,N));
  riempi(vett2,M);
  stampa(vett2,M);
  printf("\nMassimo elemento = %f ",max(vett2,M));
}
```

Come si vede, le due procedure (*riempi* e *stampa*) e la funzione *max* presentano un parametro formale valore che corrisponde ad un vettore, definito come:

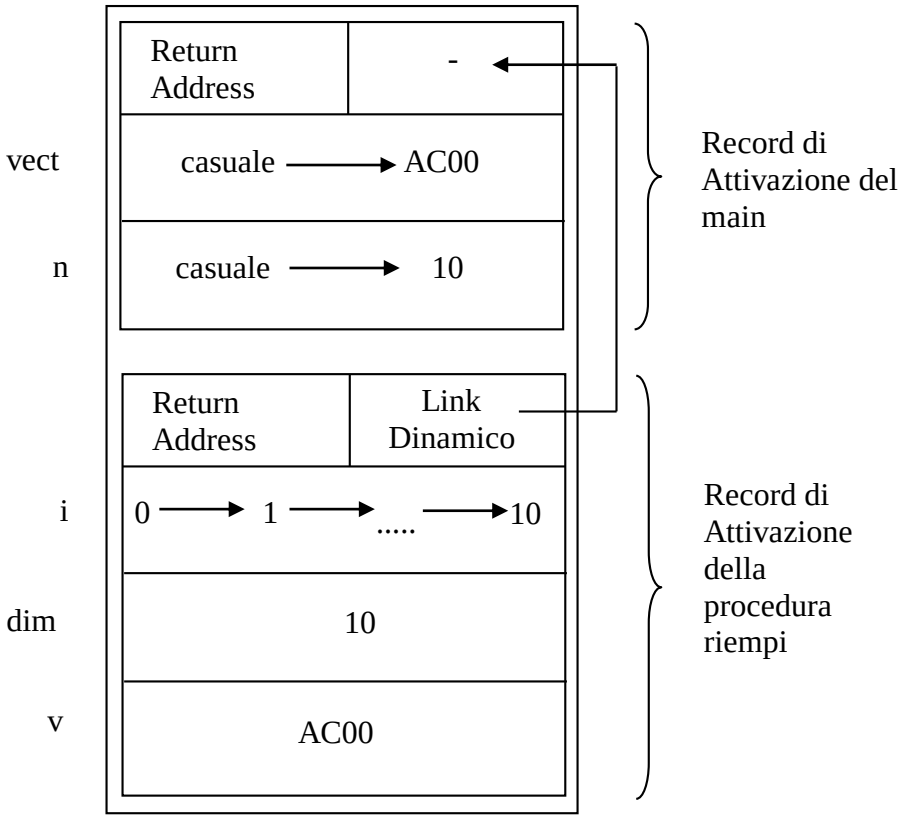
float vett[N]

Il parametro formale corrispondente a tale vettore, viene espresso, nelle tre funzioni del programma appena mostrato, utilizzando tutte e tre le forme precedentemente descritte, che sono del tutto equivalenti.

Le stesse notazioni si possono applicare anche a vettori allocati in modo dinamico nell'heap. Ad esempio, si consideri il seguente programma:

```
#include<stdio.h>
#include<stdlib.h>

void riempi(float [], int );

int main(void)
{
      float *vect;
      int n;

      n=10;
      vect=(float *)malloc(n*sizeof(float));
      riempi(vect,n);
}

void riempi(float v[], int dim){
   int i;

   for (i=0; i<dim; i++) {
        printf("Inserisci l'elemento di indice %u ",i);
        scanf("%f",&v[i]);
   }
}
```

Supponendo che il vettore dinamico venga allocato nell'heap a partire dall'indirizzo AC00, la seguente figura mostra ciò che avviene nello stack:



Come si vede il parametro formale v nella procedura riempi viene inizializzato con lo stesso valore di vect, ossia AC00. In tal modo è possibile riempire realmente ciascun elemento del vettore (tramite il comando scanf("%f",v+i)).

Per quanto detto prima, erano possibili le due seguenti definizioni alternative della procedura riempi:

```
void riempi(float *v, int dim){
   int i;

   for (i=0; i<dim; i++) {
        printf("Inserisci l'elemento di indice %u ",i);
        scanf("%f",&v[i]);
   }
}
```

```
void riempi(float v[10], int dim){
   int i;

   for (i=0; i<dim; i++) {
        printf("Inserisci l'elemento di indice %u ",i);
        scanf("%f",&v[i]);
   }
}
```

E' da notare che quest'ultima forma è in genere da evitare per **i vettori dinamici**, in quanto legata eccessivamente alla dimensione effettiva del vettore, **nota solo durante l'esecuzione**; **dunque per i vettori dinamici è consigliato limitarsi alle sole due notazioni:**

- tipobase * nome_parametro;
- tipobase nome_parametro[];

1.2.Parametri Passati per Indirizzo.

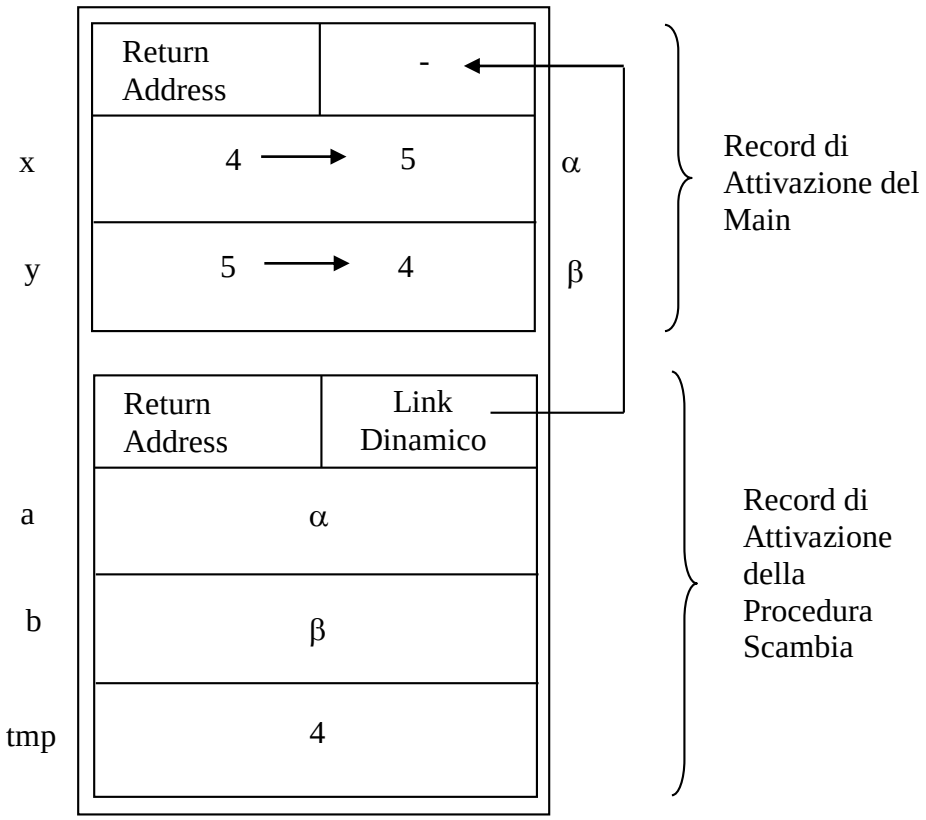
Nel caso di parametri formali passati per riferimento (utilizzando il tipo puntatore), nella cella del record di attivazione relativa al parametro formale viene copiato l'indirizzo della variabile che costituisce, all'atto della chiamata della procedura o funzione, il parametro attuale corrispondente al parametro formale.

Si consideri il seguente esempio:

```
#include <stdio.h>

void scambia ( int *a, int *b)
{     int tmp

      tmp=*a;
      *a=*b;
      *b=tmp;
}

int main(void)
{
      int x,y;

      x=4;
      y=5;

      scambia(&x,&y);
      printf("%d %d \n",x,y);
}
```

Quando il main viene eseguito verrà allocato il relativo record di attivazione, composto dalle variabili locali x e y. In tal caso l'Address return è relativo al Sistema Operativo che ha chiamato il main. Inoltre manca il link dinamico, che in questo caso non è necessario, come spiegato in precedenza. Infine mancano i parametri formali del main. Il main chiama la procedura scambia, passando i parametri attuali &x e &y, ossia gli indirizzi di x e y. Ciò comporta la creazione del relativo record di attivazione. Esso contiene: l'indirizzo di ritorno all'istruzione che il main dovrà eseguire dopo la conclusione della procedura scambia (tale istruzione è printf("%d %d \n",x,y);), il link dinamico al record di attivazione del main, la variabile locale tmp e i parametri formali a e b. Questi parametri non contengono una copia di valori, ma **l'indirizzo (qui indicato con α e β) delle corrispondenti variabili x e y del main !**

Quindi i valori di a e b coincidono con gli indirizzi dei corrispondenti parametri attuali passati alla procedura ossia l'indirizzo di x, indicato con α, e l'indirizzo di y, ossia β. La procedura scambia inizializza tmp al valore assunto dalla locazione il cui indirizzo è contenuto in x, ossia 4. L'assegnazione *a=*b comporta che la cella il cui indirizzo è presente in a, ossia l'indirizzo α, viene modificata assumendo il valore pari alla cella il cui indirizzo è contenuto in b, ossia β. Dunque, tramite gli indirizzi contenuti in a e b, le variabili x e y vengono modificate dalle istruzioni *a=*b e *b=tmp, diventando rispettivamente 5 e 4. La procedura scambia termina ritornando al chiamante, ossia al main. Tutto il record di attivazione della procedura scambia viene disallocato. Il ritorno al main comporta il ripristino del record di attivazione del main. **Questa volta, però, le variabili x e y non hanno più i valori originali perché sono stati modificati dalla procedura scambia.**


Return Address
-
x
4 → 5
α
y
5 → 4
β
Record di Attivazione del Main
Return Address
Link Dinamico
a
α
b
β
tmp
4
Record di Attivazione della Procedura Scambia