Record di Attivazione
Cenni sulla ricorsione

Maurizio Palesi
Salvatore Serrano

Il modello a RUN-TIME

- Ogni volta che viene invocata una funzione o procedura
  - ➔ si crea una nuova attivazione (istanza) del servitore
  - ➔ viene allocata la memoria per i parametri e per le variabili locali
  - ➔ si effettua il passaggio dei parametri
  - ➔ si trasferisce il controllo al servitore
  - ➔ si esegue il codice della funzione

Il modello a RUN-TIME: AMBIENTE

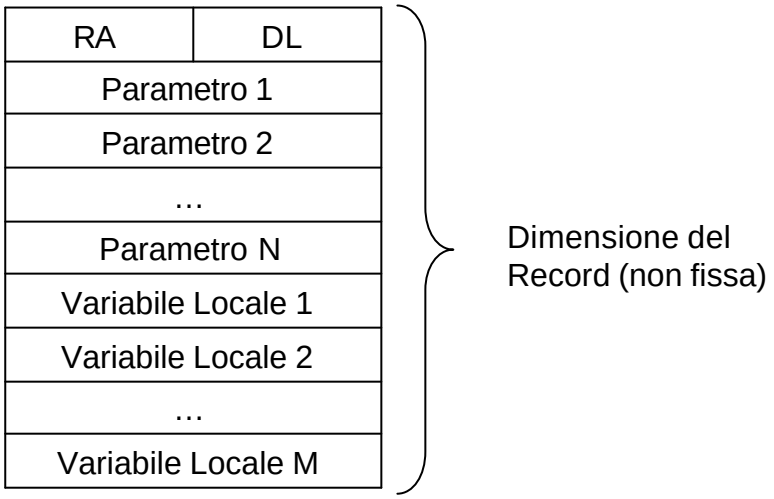
- Al momento dell' *invocazione*, si crea un *nuovo ambiente*
  - ➔Viene creata una struttura dati che contiene il **binding** dei **parametri** e degli **identificatori** definiti localmente alla funzione detta

RECORD DI ATTIVAZIONE

Record di Attivazione

- È il "mondo della funzione": contiene tutto ciò che ne caratterizza l'esistenza:
  - ➔**Parametri**
  - ➔**Variabili**
  - ➔**Indirizzo di ritorno (Return Address RA)**
    - ✓Indica il punto a cui tornare (nel codice della funzione chiamante) al termine della funzione, per permettere alla funzione chiamante di proseguire una volta che la funzione chiamata termina
  - ➔**Collegamento al record di attivazione della funzione chiamante (Dynamic Link DL)**

Record di Attivazione



Record di Attivazione

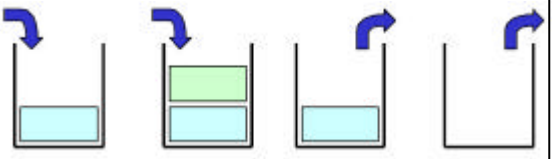
- Rappresenta il “mondo della funzione”: **nasce e muore con essa**
  - ➔ È creato al momento della invocazione di una funzione
  - ➔ Permane per tutto il tempo in cui la funzione è in esecuzione
  - ➔ È distrutto (deallocato) al termine dell’esecuzione della funzione
- Ad ogni chiamata di funzione viene creato un **nuovo record**, specifico per **quella chiamata** di **quella funzione**
- La dimensione del record di attivazione
  - ➔ Varia da una funzione all’altra
  - ➔ Per una data funzione, è fissa e calcolabile a priori

Record di Attivazione

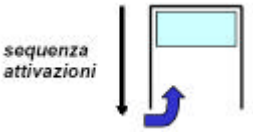
- Funzioni che chiamano altre funzioni danno luogo a una sequenza di record di attivazione
  - ➔Allocati secondo l'ordine delle chiamate
  - ➔Deallocati in ordine inverso
- La sequenza dei **link dinamici** costituisce la cosiddetta **catena dinamica** che rappresenta la **storia delle attivazioni** ("chi ha chiamato chi")
- L'area di memoria in cui vengono allocati i record di attivazione deve essere gestita **come una pila**

Stack

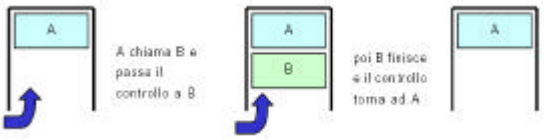
- Struttura dati gestita con politica **LIFO** (**L**ast **I**n, **F**irst **O**ut)



- Normalmente lo **STACK** dei record di attivazione si disegna nel seguente modo



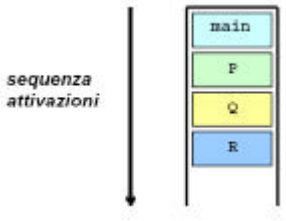
- Quindi, se la funzione A chiama la funzione B lo stack evolve nel seguente modo



Esempio di chiamate annidate

```
int R(int A) {
  return A+1;
}
int Q(int x) {
  return R(x);
}
int P() {
  int a=10;
  return Q(a);
}
main() {
  int x = P();
}
```

Sequenza chiamate:
S.O. ® main() ® P() ® Q() ® R()



Esempio di ricorsione: FATTORIALE

```
#include <stdio.h>
int fact(int n) {
  if (n<=0)
     return 1;
  else
     return n*fact(n-1);
}
main() {
  int n;
  printf("Inserisci n:");
  scanf("%d", &n);
  printf("n! = %d", fact(n));
}
```

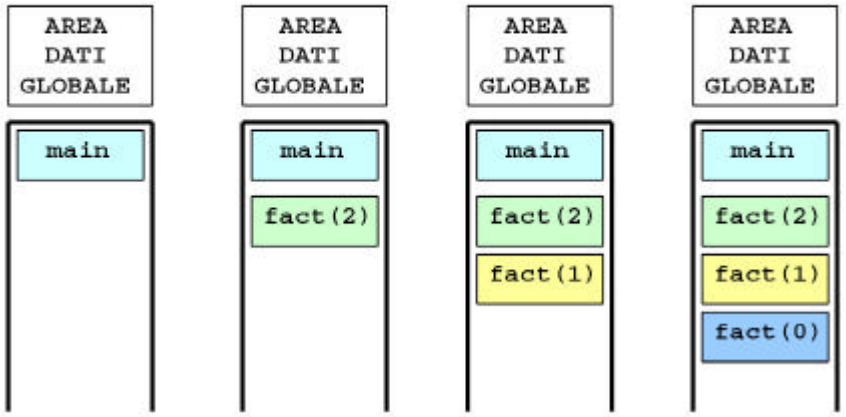
FACT(2): sequenza attivazioni

Situazione all'inizio dell'esecuzione del `main()`

Il `main()` chiama `fact(2)`

`fact(2)` chiama `fact(1)`

`fact(1)` chiama `fact(0)`

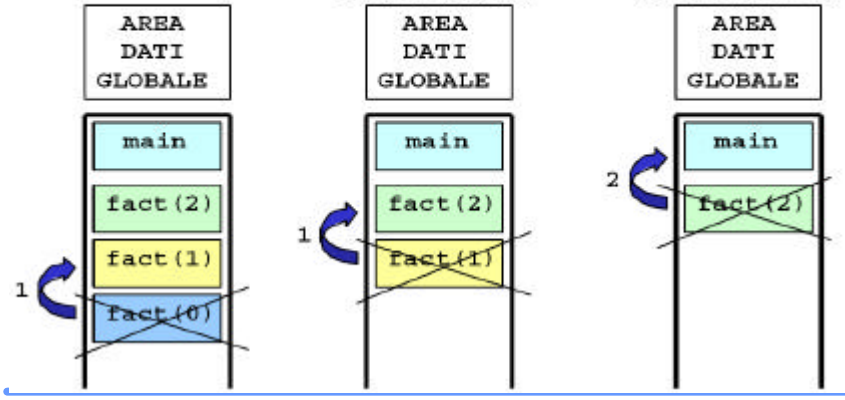


FACT(2): sequenza attivazioni

`fact(0)` termina restituendo il valore 1. Il controllo torna a `fact(1)`

`fact(1)` effettua la moltiplicazione e termina restituendo il valore 1. Il controllo torna a `fact(2)`

`fact(1)` effettua la moltiplicazione e termina restituendo il valore 2. Il controllo torna al `main()`



Esercizio

- Calcolare la somma dei primi N interi utilizzando la ricorsione

Soluzione

```
#include <stdio.h>
int sommaFinoA(int n) {
  if (n==1)
     return 1;
  else
     return n+sommaFinoA(n-1);
}
main() {
  int n;
  printf("Inserisci n:");
  scanf("%d", &n);
  printf("Ris = %d", sommaFinoA(n));
}
```

Esercizio

- Risolvere in maniera ricorsiva il problema di Fibonacci

Il problema di Fibonacci, o problema dei conigli, consiste nel determinare quante coppie di conigli ci saranno dopo N mesi, nelle seguenti ipotesi:

- ➔Al mese 0 c'è una coppia di conigli neonati
- ➔Un coniglio diventa fertile dopo un mese dalla nascita
- ➔Ogni coppia di conigli fertile genera ogni mese una nuova coppia di conigli
- ➔Non c'è mortalità di conigli

Soluzione

```
#include <stdio.h>
unsigned Fibonacci(unsigned n) {
  if (n<2)
     return 1;
  else
     return Fibonacci(n-1)+Fibonacci(n-2);
}

main() {
  unsigned m;
  printf("Inserisci mese:");
  scanf("%d", &m);
  printf("Coppie = %d", Fibonacci(m));
}
```