Tipo di Dati Astratto

Tipi di Dati

Per tipo di dato si intende un insieme di valori e un insieme di operazioni che si possono applicare ad esso.

Ogni tipo di dato ha una propria rappresentazione in memoria, cioè viene rappresentato attraverso un'opportuna codifica sfruttando un certo numero di celle di memoria.

In C ed in altri linguaggi ogni variabile ha un tipo associato. Ciò permette di:

- associare un insieme di valori ammissibili per quella variabile
- determinazione a priori della quantità di memoria necessaria per la memorizzazione di quella variabile.

Tipo di Dati Astratto

Un **tipo di dati astratto** (ADT) è un tipo di dati organizzato in modo tale che la specifica egli oggetti e la specifica delle operazioni sugli oggetti sono distinte dalla rappresentazione degli oggetti e dall'implementazione delle operazioni.

Cosa vuol dire che la specifica dalle operazioni differisce dall'implementazione delle operazioni. ??

Indipendenza dall'implementazione

La specifica di una operazione è formata da:

- Prototipo della funzione che implementa l'operazione
- Pre e postcondizione della funzione

Requisito molto importante perchè implica:

Indipendenza dalla particolare implementazione (linguaggio usato) e dalla rappresentazione interna che tale linguaggio effettua del particolare tipo di dato

Classificazione delle operazioni su un ADT

Le funzioni associate ad un tipo di dato possono suddividersi nelle seguenti classi:

Funzioni per creare: Costruttori.

Creano una nuova istanza del tipo designato

Funzioni per distruggere: Distruttori.

Eliminano una particolare istanza del tipo designato

Funzioni per trasformare: Trasformano una istanza del tipo designato usando una o più istanze esistenti

Funzioni per informare/osservare: forniscono certe informazioni su una particolare istanza

Definizione di un ADT

Primo passo:

Definizione dell'ADT relativo all' oggetto che intendiamo studiare

-> acquisire elementi del tipo di oggetto senza riferimenti alla implementazione.

Tipo di dato e funzioni su oggetti di tale tipo. normalmente costruttori, trasformatori e informatori

Secondo passo

Analisi ed esame della particolare implementazione e rappresentazione dell ADT

ARRAY come ADT

Normalmente siamo abituati a vedere un array come un insieme di locazioni contigue di memoria che contengono dati omogenei.

Astraiamoci dalla particolare implementazione.

Un array lo possiamo pensare come un insieme di coppie <Indice, Valore>, tale che ad ogni indice definito viene associato univocamente un determinato valore

ARRAY come ADT

Operazioni su array:

- Creare un array
- Leggere un valore
- Registrare un valore

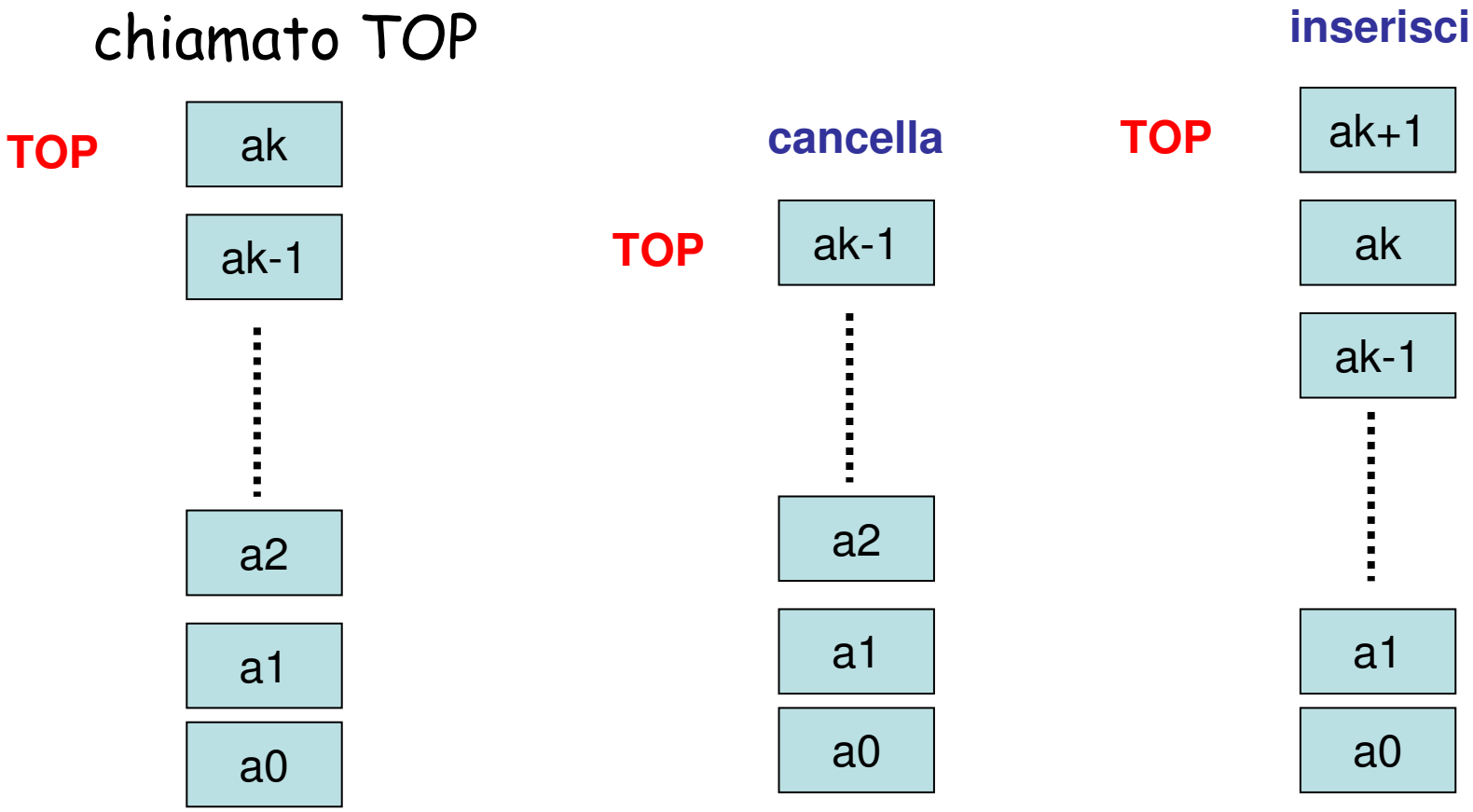
ARRAY come ADT

Oggetto: Insieme di coppie <indice, valore>, dove indice è un insieme ordinato e finito per esempio {0,1,...,n-1} e ad ogni valore dell'indice c'è un valore univocamente associato

- Crea(j,set): j è il numero di indice e set è una lista di j valori. Restituisce un array di j elementi formati dagli elementi di set in
- Preleva(A,i): A è un array, i è un indice. Restituisce il valore contenuto nella posizione i in A
- Scrivi(A,i,x): scrive in A nella posizione i l'elemento x e restituisce il nuovo array così ottenuto

STACK – PILE: LIFO

Uno stack (pila di dati) è una lista ordinata di elementi dove gli inserimenti e le cancellazioni vengono effettuate solo ad un estremo della lista chiamato TOP



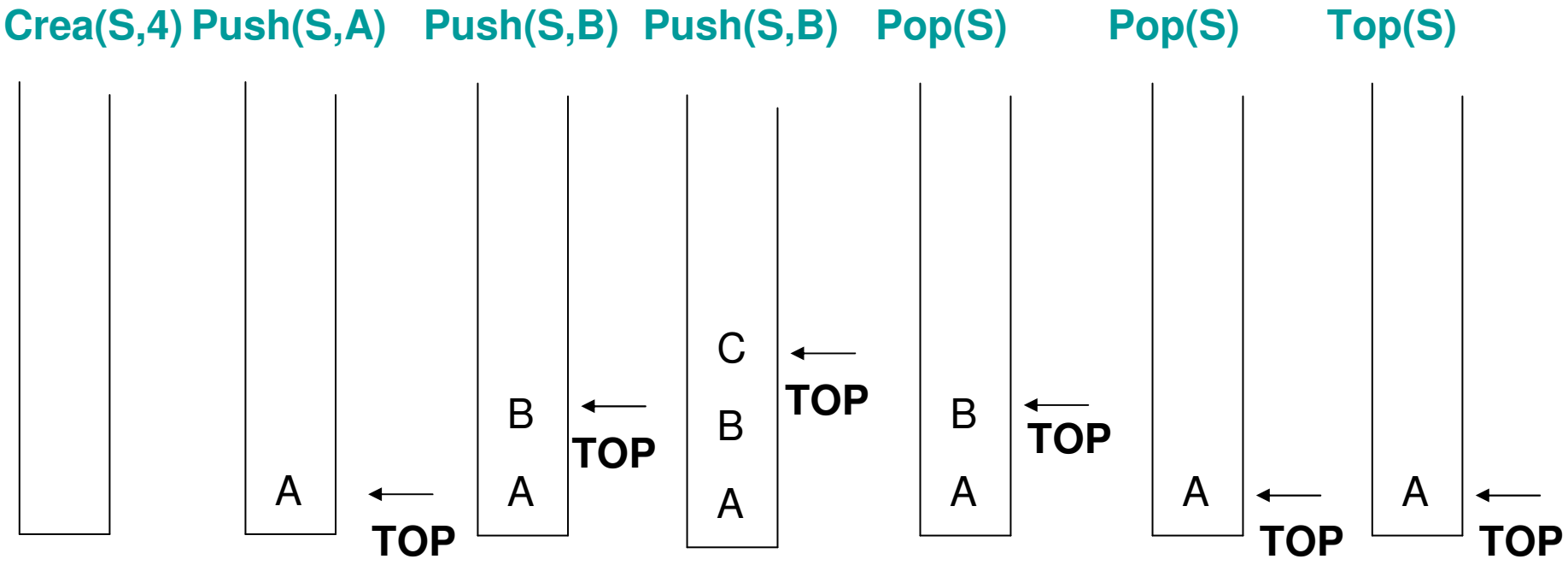
STACK come ADT

Oggetto: una lista ordinata con zero o più elementi

Funzioni:

- Crea(maxsize): crea uno stack vuoto la cui capienza massima è data da max_size. Restituisce uno stack
- Pieno(stack, max_size). Restituisce un booleano, vero se e solo se lo stack contiene esattamente max_size elementi.
- Push(Stack, elem). Restituisce un nuovo stack dove l'elemento elem è posto in cima allo stack (se lo stack non è già pieno)
- Pop(stack). Restituisce un nuovo stack da cui viene eliminato l'elemento presente nel suo top, se esiste.
- Top(stack). Restituisce , se esiste, l'elemento nel top dello stack, senza modificarlo.

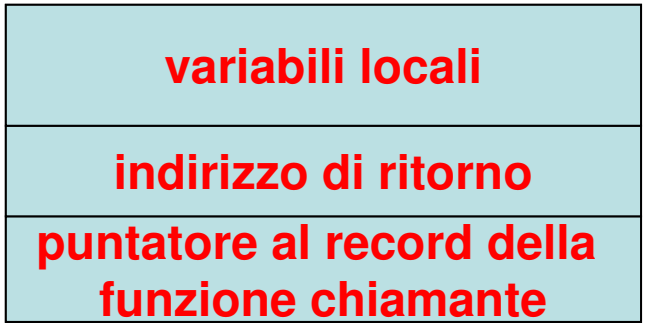
STACK come ADT



Un Esempio:
Stack di Sistema

È uno stack speciale che viene usato per gestire ed elaborare le chiamate delle funzioni

Ogni volta che una funzione viene chiamata il programma crea una struttura chiamata ACTIVATION RECORD e la pone in cima allo stack di sistema

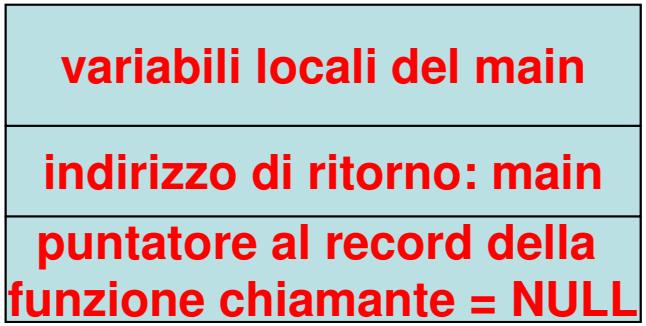


ACTIVATION RECORD

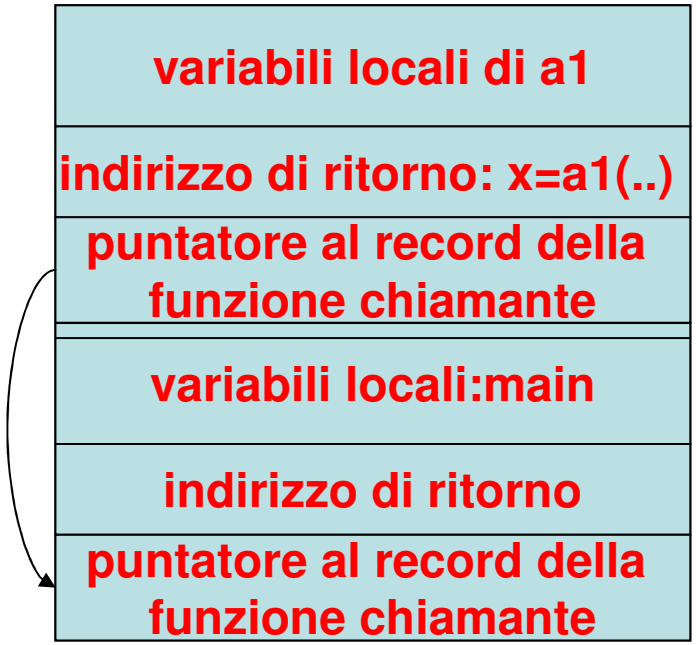
Stack di Sistema

```
int main(){
.....
x=a1(...);
...
}
int a1 (.....){
......
}
```

prima della chiamata di a1



dopo la chiamata di a1



dopo l'esecuzione di a1