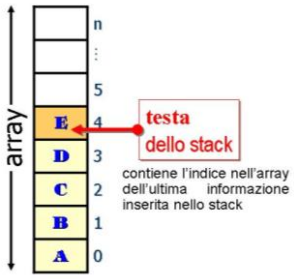
Esercizi di verifica

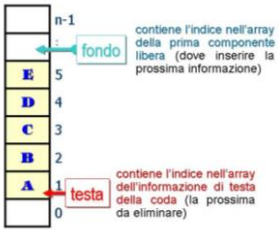
8 – Strutture dati dinamiche lineari: pila, coda e lista lineare

P2_08_03_T

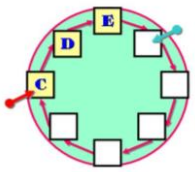
37. [liv.1] Simulare in *C* la gestione di una **pila** (**stack**) tramite array statico (può essere anche un array di struct) creando le funzioni di manipolazione `push()` [inserimento] e `pop()` [eliminazione]. Il programma deve prevedere un menù che consenta di scegliere l'operazione da eseguire.



38. [liv.1] Simulare in *C* la gestione di una **coda** (**queue**) tramite array statico (può essere anche un array di struct) creando le funzioni di manipolazione `enqueue()` [inserimento] e `dequeue()` [eliminazione]. Il programma deve prevedere un menù che consenta di scegliere l'operazione da eseguire. Le informazioni NON vanno spostate!



39. [liv.3] Simulare in *C* la gestione di una **coda** (**queue**) tramite array statico **circolare** (può essere anche un array di struct) creando le funzioni di manipolazione `enqueue()` [inserimento] e `dequeue()` [eliminazione]. Il programma deve prevedere un menù che consenta di scegliere l'operazione da eseguire.



P2_08_04_T

40. [liv.1] Simulare in *C* l'algoritmo di visita di una *lista lineare* già memorizzata mediante un array statico di `struct` (come nella tabella in basso) in cui il primo campo contiene l'informazione ed il secondo contiene il *link* al nodo successivo (in questo caso il *link* è l'indice di una componente dell'array). Memorizzando nell'array i dati come mostrato nella figura che segue, l'output del programma consiste nell'elenco di nomi ordinato alfabeticamente.

dati di input: ListaNomi

ListaNomi

INFO	pnext
Anna	5
Mario	8
Giuseppe	6
Angela	0
Valeria	-1
Fabrizio	7
Marianna	1
Giovanni	2
Patrizia	10
Valentina	4
Sara	9

p_Testa

41. [liv.3] Simulare in *C* la *gestione delle camere di un albergo* mediante liste lineari rappresentate su un array di `struct`: i principali campi sono le "informazioni" (numero di camera, cliente, etc.) ed i "link" (puntatori ai

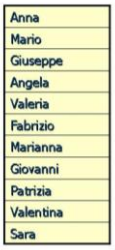
nodi della lista). [Suggerimento: L'array di struct corrisponde alla memoria in cui allocare la lista delle camere libere (`ListaLibera`) e la lista delle camere occupate (`ListaDati`). È necessario creare prima la `ListaLibera`, inizializzando l'array dei link in modo che ogni componente punti alla componente successiva. Ogni nodo da inserire nella `ListaDati`, quando una camera viene assegnata ad un cliente, è prelevato dalla testa della `ListaLibera` ed inserito nella testa della `ListaDati`; mentre il nodo da eliminare dalla `ListaDati`, quando una particolare camera viene liberata, è restituito alla `ListaLibera` (in testa) per poter essere riutilizzato in seguito.]

P2_08_06_C

42. [liv.1] Realizzare la gestione di una lista lineare mediante menù (visualizzazione mediante visita, inserimento in testa, inserimento in mezzo, eliminazione in testa, eliminazione in mezzo) implementando la lista lineare con una struttura autoriferente dinamica.

43. [liv.2] A partire dalla versione ricorsiva di costruzione di una lista in *C*, scriverne la versione iterativa.

44. [liv.2] Scrivere una *function C* per costruire una *lista ordinata in ordine alfabetico* a partire da un elenco di nomi in ordine casuale, come nel seguente.



P2_08_08_AT

45. [liv.1] Realizzare in *C* le funzioni per la gestione, mediante menù, delle strutture dati *pila* e *coda* mediante *lista lineare dinamica e generica* con [rispettivamente senza] nodo sentinella.

46. [liv.2] Realizzare in *C* le funzioni per la gestione, mediante menù, delle strutture dati *catena* e *lista bidirezionale* mediante *lista lineare dinamica e generica* con [rispettivamente senza] nodo sentinella.

47. [liv.3] Implementare in *C* il prodotto righe×colonne di due matrici sparse rappresentate tramite liste multiple. Scegliere per ciascuna matrice la rappresentazione più idonea.