Strutture dati

- Organizzano e memorizzano insiemi di dati
- Permettono accesso e manipolazione efficienti
- Fondamentali per algoritmi e programmazione
- Differenti tipi a seconda dell’esigenza

Array

- Collezione di elementi dello stesso tipo
- Accesso tramite indice numerico
- Dimensione fissa in molti linguaggi
- Esempio Python: `numeri = [1, 2, 3, 4]`

Liste

- Collezione ordinata di elementi
- Dimensione dinamica
- Permette inserimento, rimozione e modifica
- Esempio Python: `lista = ["a", "b", "c"]`

Dizionari / Mappe

- Collezione di coppie chiave-valore
- Accesso tramite chiave univoca
- Ideale per dati associativi
- Esempio Python: `diz = {"nome": "Luca", "eta": 20}`

Stack (pila)

- Accesso LIFO (Last In, First Out)
- Operazioni principali: push e pop
- Utilizzato in algoritmi di backtracking
- Gestione funzioni e chiamate ricorsive

Queue (coda)

- Accesso FIFO (First In, First Out)
- Operazioni principali: enqueue e dequeue
- Ideale per gestione code e processi
- Utilizzata nei sistemi operativi

Set

- Collezione di elementi unici
- Non mantiene ordine
- Utile per eliminare duplicati
- Operazioni di unione, intersezione, differenza

Importanza delle strutture dati

- Migliorano efficienza dei programmi
- Base per algoritmi complessi
- Permettono gestione organizzata di informazioni
- Fondamentali in ogni linguaggio di programmazione