Differences between array and Linked List

Arrays:

- Fixed size
- Same data type
- starts from 0
- contiguous memory location

Linked Lists:

- Consists two component: data,next(link)
- Dynamic size
- Non-contiguous memory location
- Sequential access

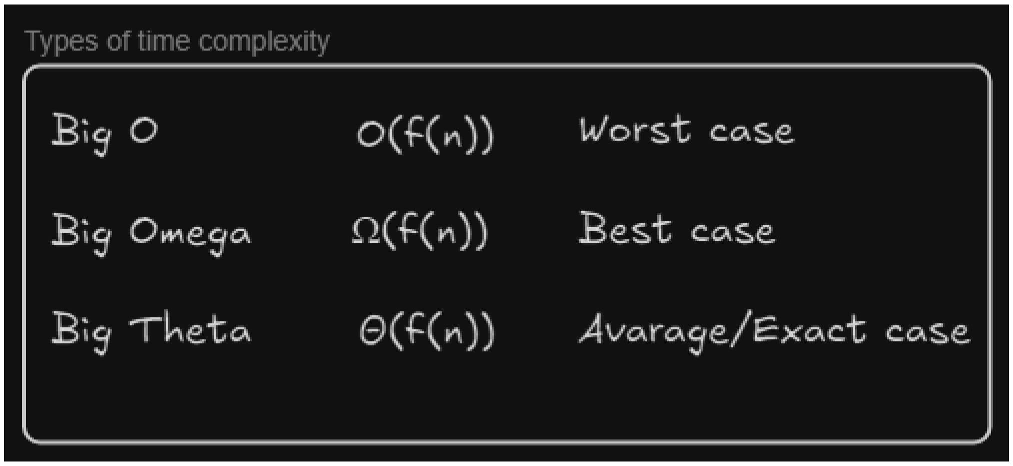
Stack is a linear data structure that operates on the LIFO
Key operations:

- push (adding elements to the top).
- pop (removing elements from the top)
- peek (viewing top elements without removing)

Big O, Big omega, Big theta
Algorithm Complexity
1.Worst Case

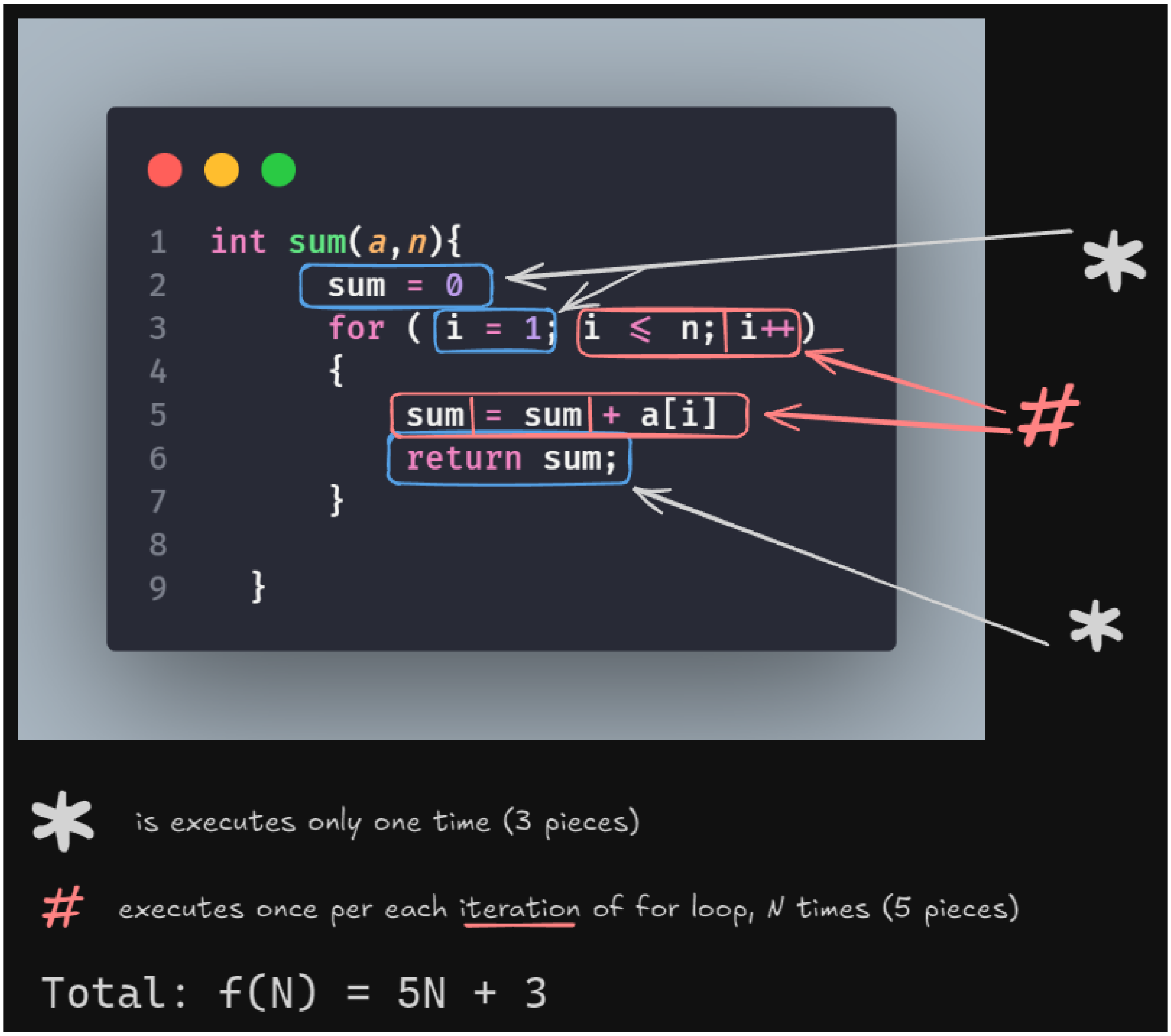
- Maximum running time of an Algorithm.
- provides guarantee on running time
  2.Best Case:
- Minimum running time
- lower bound (limit)
  3.Avarage Case:
- Expected running time over all inputs

- Analysis all possible input
- Compute expected running time



Types of time complexity

Big O	$O(f(n))$	Worst case
Big Omega	$\Omega(f(n))$	Best case
Big Theta	$\Theta(f(n))$	Avarage/Exact case



```
int sum(a,n){
      sum = 0
      for ( i = 1; i ⩽ n; i++)
      {
          sum = sum + a[i]
          return sum;
      }

  }
```

* is executes only one time (3 pieces)
# executes once per each iteration of for loop, N times (5 pieces)
Total: f(N) = 5N + 3

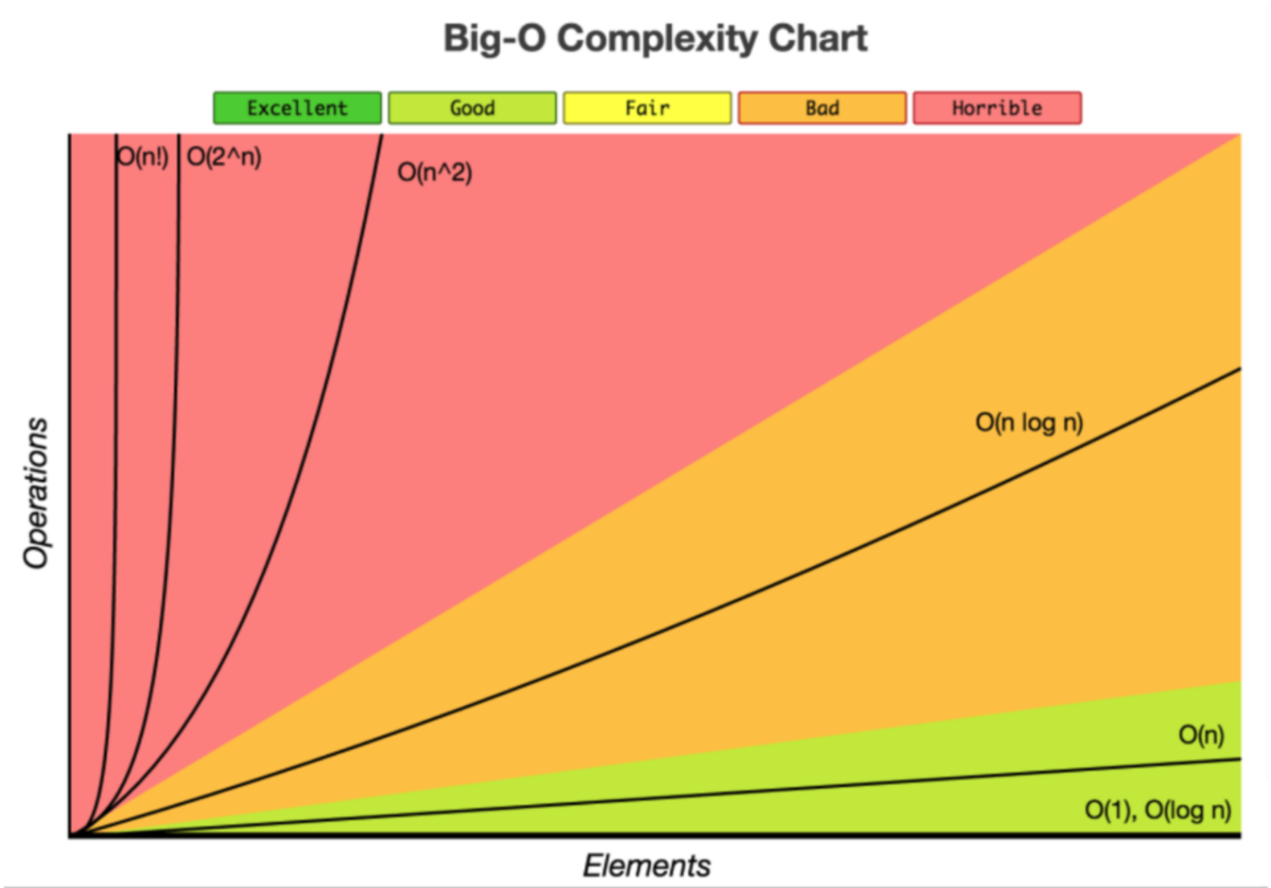
Aspect	Time Complexity	Space Complexity
Measures	Execution time / number of steps	Memory usage
Typical unit	Operations	Bytes or variables
Example	O(n²) → nested loops	O(n) → additional array

Type of space complexity

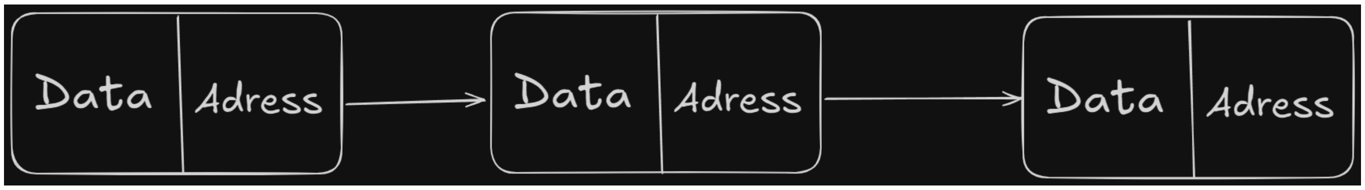
Type1:

- A fixed part to store certain data and var's, that are independent size of the problem
- Example: const int MAX = 100;
  Type 2:
- A variable part is a space required by var's

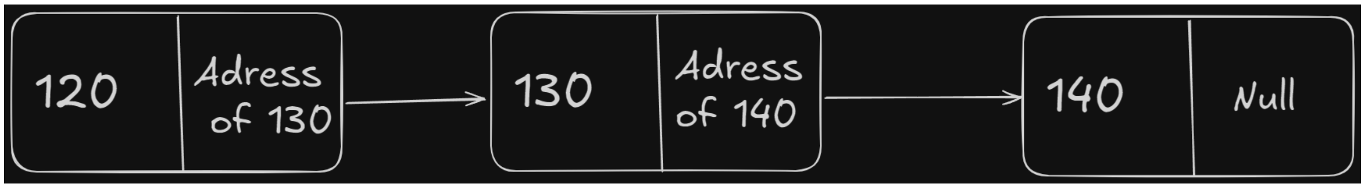
- Example: malloc(), recursive stack space

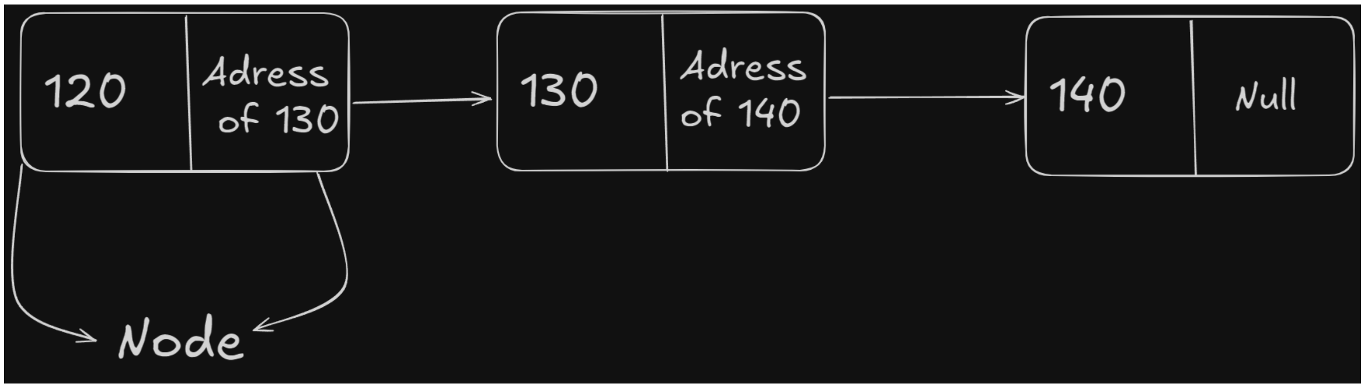


Linked Lists

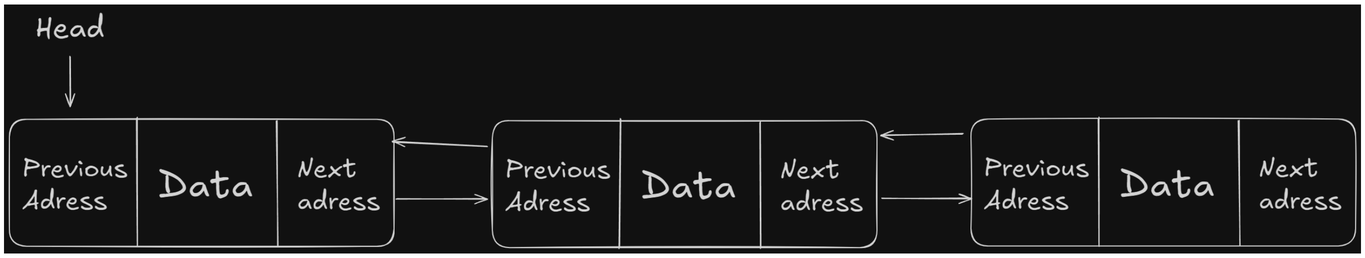


Examples:

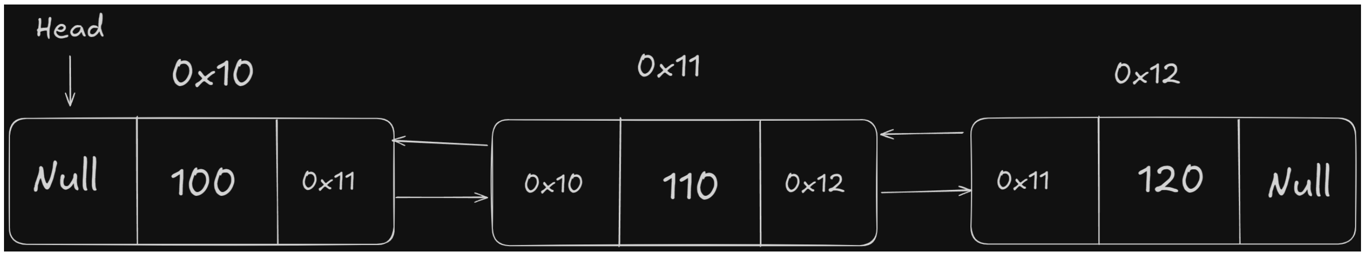




Double Linked List

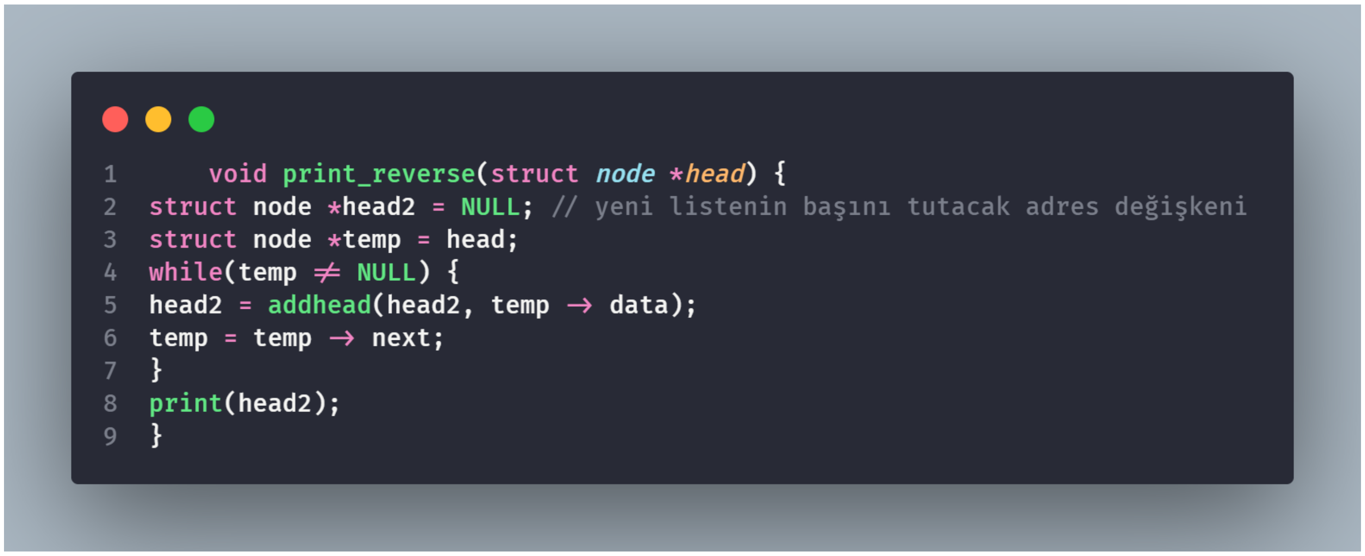


Example:



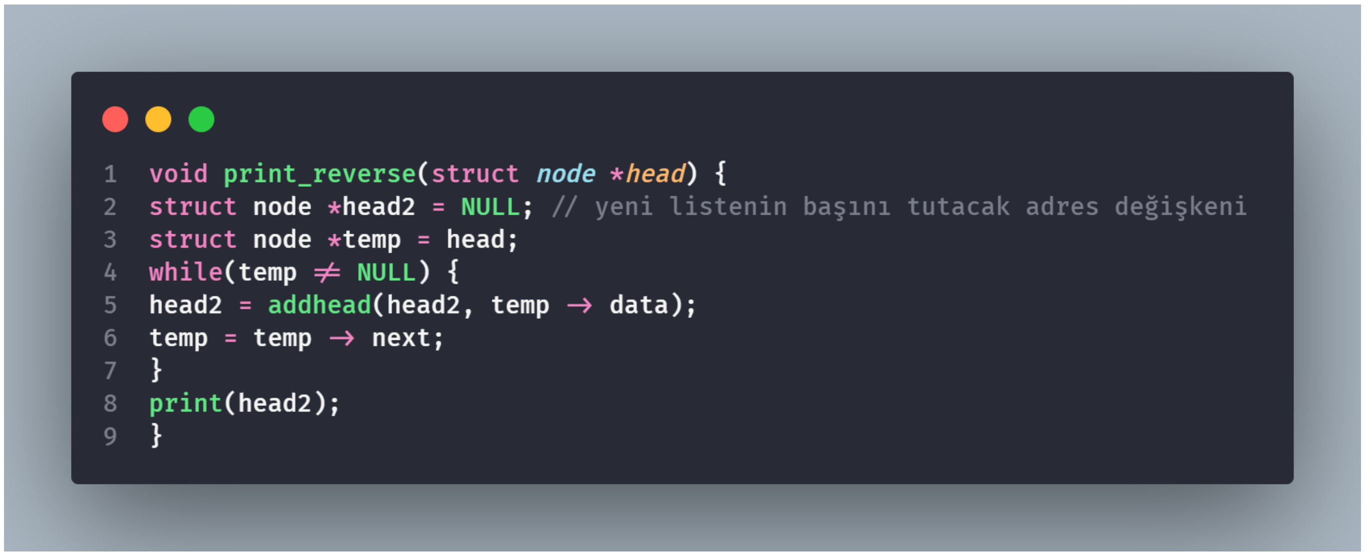
Linked list işlemleri

Tek Bağlı Doğrusal Listelerde Verileri Tersten Yazdırmak



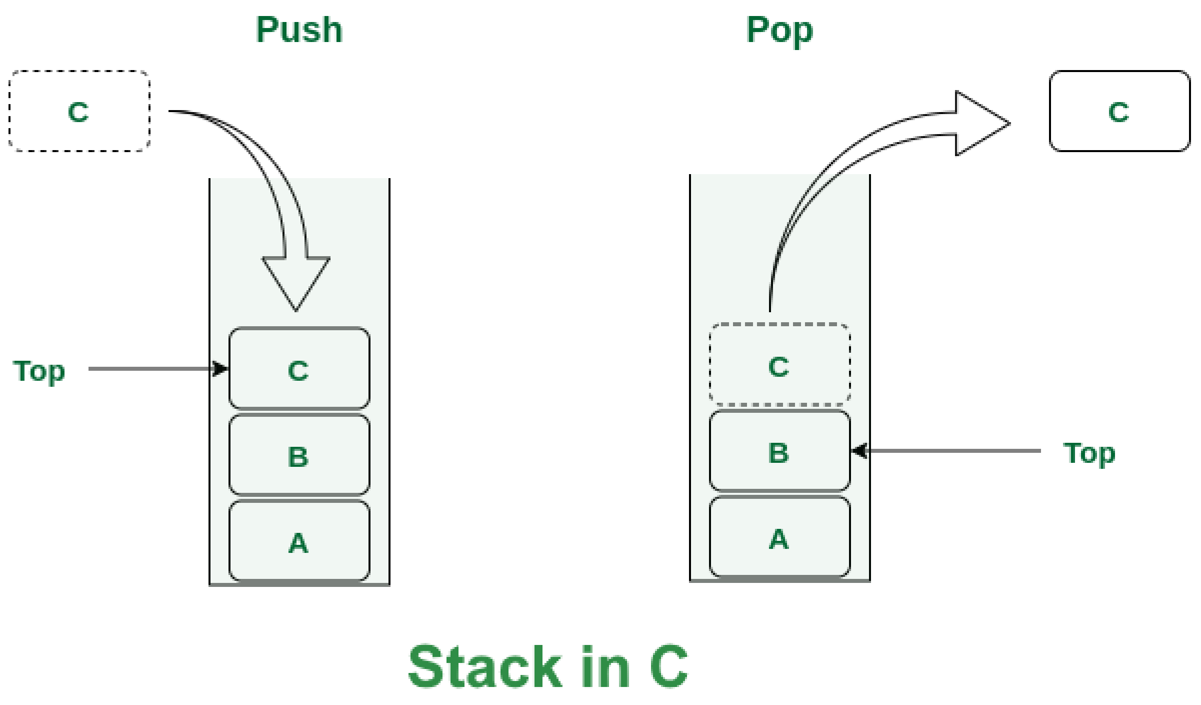
```c
    void print_reverse(struct node *head) {
struct node *head2 = NULL; // yeni listenin başını tutacak adres değişkeni
struct node *temp = head;
while(temp != NULL) {
head2 = addhead(head2, temp -> data);
temp = temp -> next;
}
print(head2);
}
```

Tek Bağlı Doğrusal Listenin Kopyasını Oluşturmak



```c
void print_reverse(struct node *head) {
struct node *head2 = NULL; // yeni listenin başını tutacak adres değişkeni
struct node *temp = head;
while(temp != NULL) {
head2 = addhead(head2, temp -> data);
temp = temp -> next;
}
print(head2);
}
```

Stack Yapısı


Push
Pop
C
Top
C
B
A
C
C
B
A
Top
Stack in C