Linked List – Introduction

Team Emertxe

Linked List -Introduction

Introduction

Why?

Array

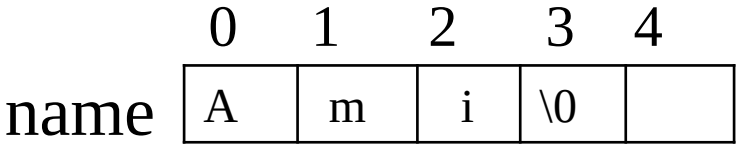
Static Array

Dynamic Array

Introduction

Static Array

- Memory ⇒ Stack



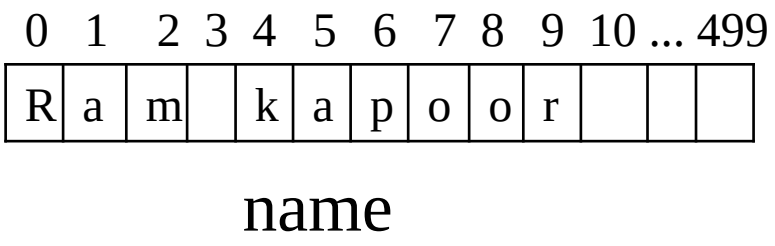
- Ex: char name[5] ;
- “Ami”
- “Ram Kapoor”
- **Drawbacks**
- Shortage of memory

Introduction

Static Array

- Memory ⇒ Stack

- Ex: char name[500] ;
- “Ami”
- “Ram Kapoor”
- **Drawbacks**
- Wastage of memory
- Shortage of memory



Introduction

Static Array

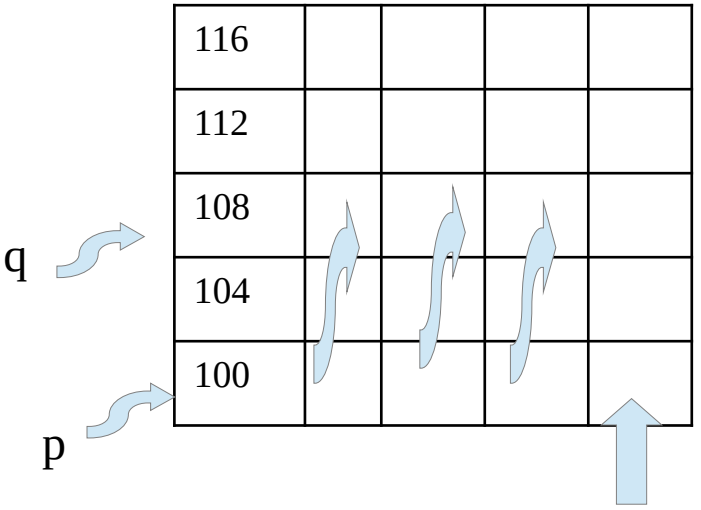
- Memory  Stack

- Ex: char name[n] ; --> Size of array
- Based on **n** value memory can be allocated.
- **Drawbacks**
- Cannot extend or shrink the memory.

Introduction

Dynamic Array

- Memory → Heap
- **Functions :** Malloc, Calloc , Realloc
- char *p = malloc(3 * sizeof(char))
- char *q = realloc(p, 7*sizeof(char))
- **Drawback**
- Reallocating/Copying the old data to a new location is time consuming.



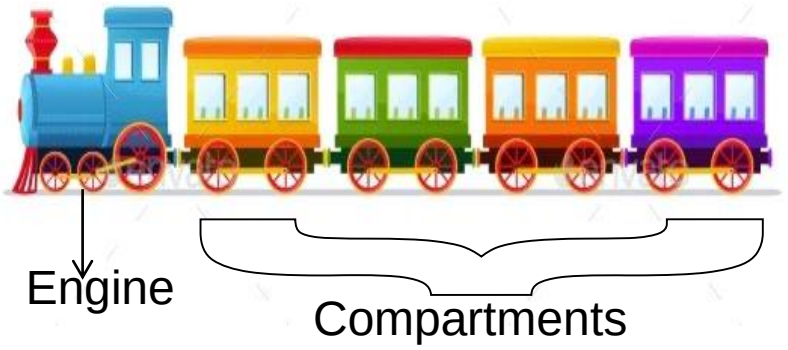
Memory Segments

What is Linked List

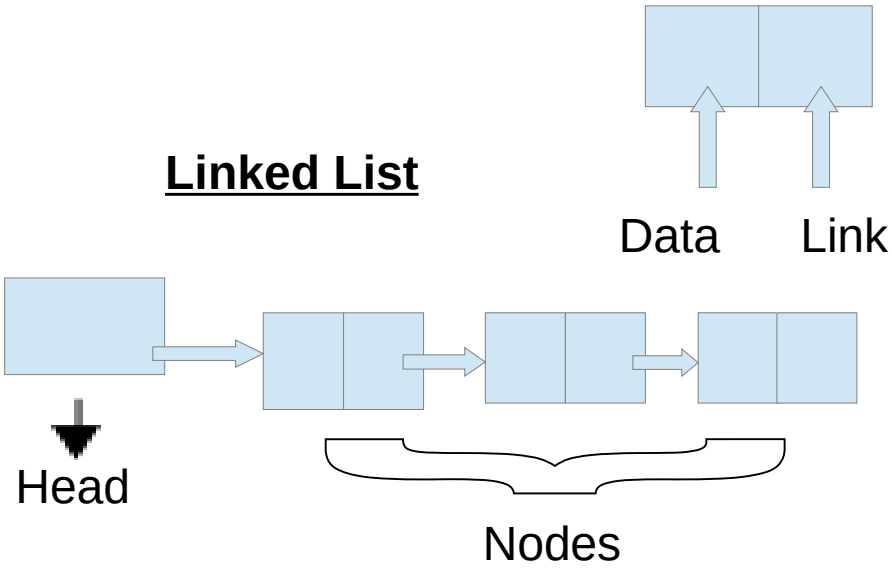
Introduction

What?

Train



Linked List



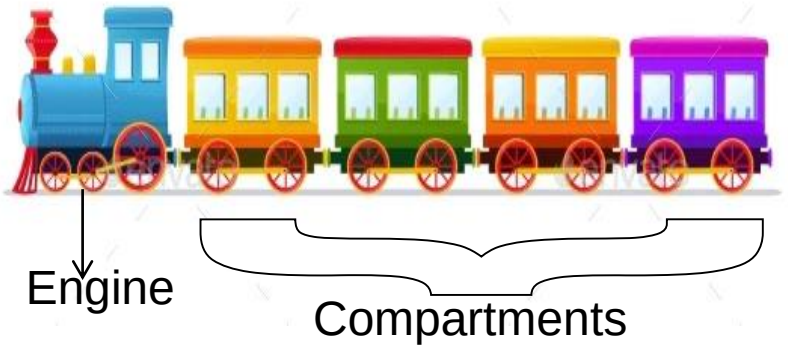
Definition

It is a Linear data structure that consist of sequence of nodes which are connected to each other to form a list.

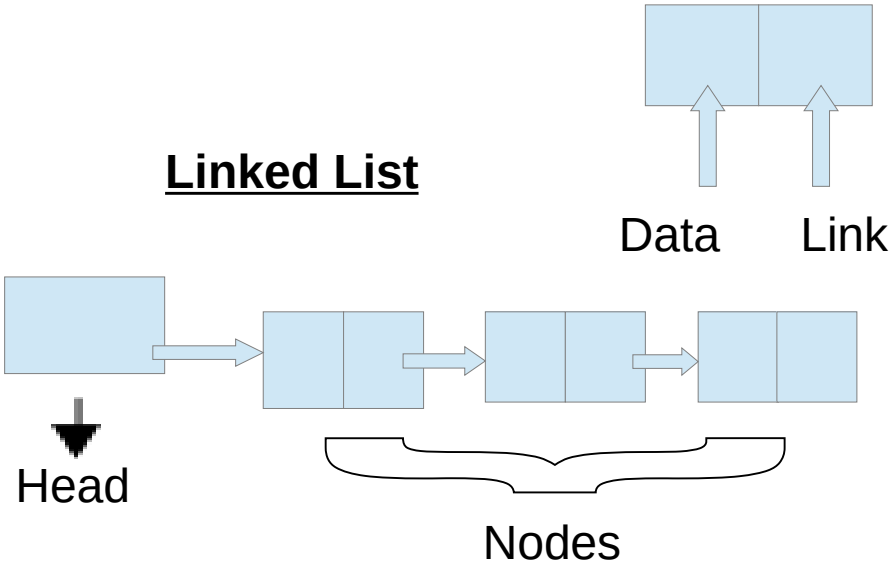
Introduction

What?

Train



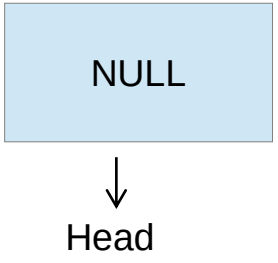
Linked List



Head -> Pointer to the first node.
Data -> Data to be inserted.
Link -> Link to the next node

Introduction

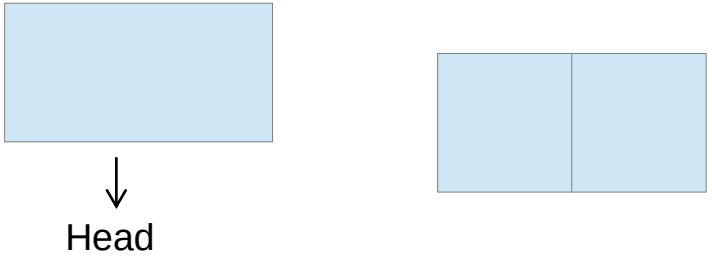
Example



Data Structure -Linked List

Introduction

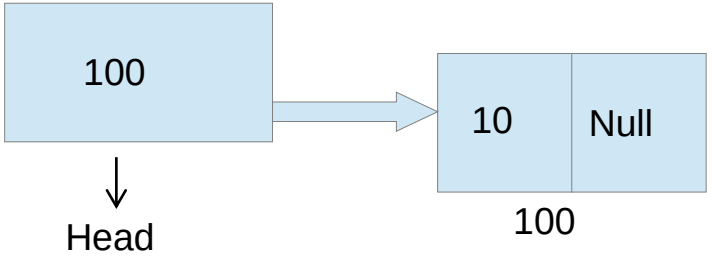
Example



Data Structure -Linked List

Introduction

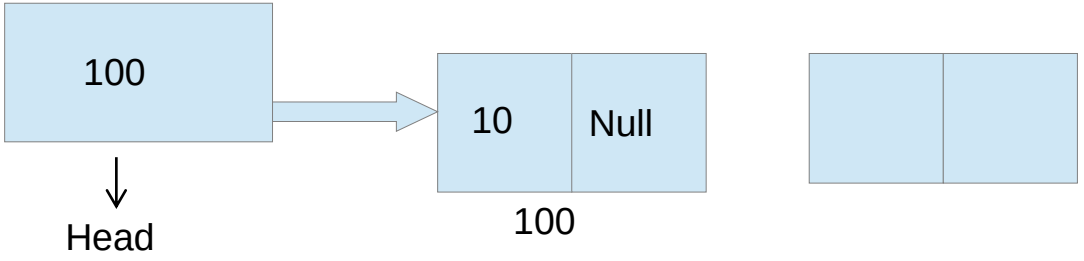
Example



Data Structure -Linked List

Introduction

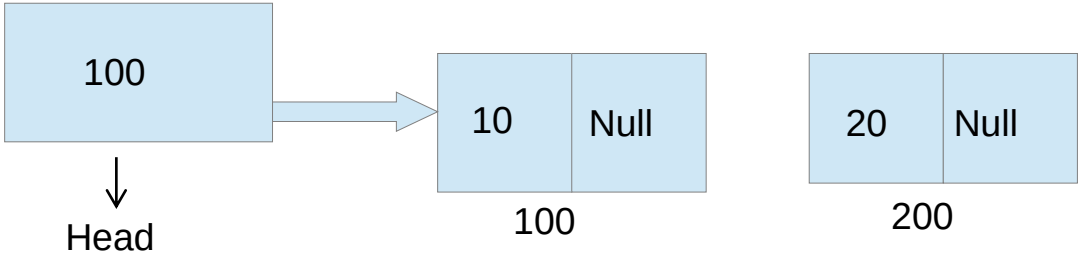
Example



Data Structure -Linked List

Introduction

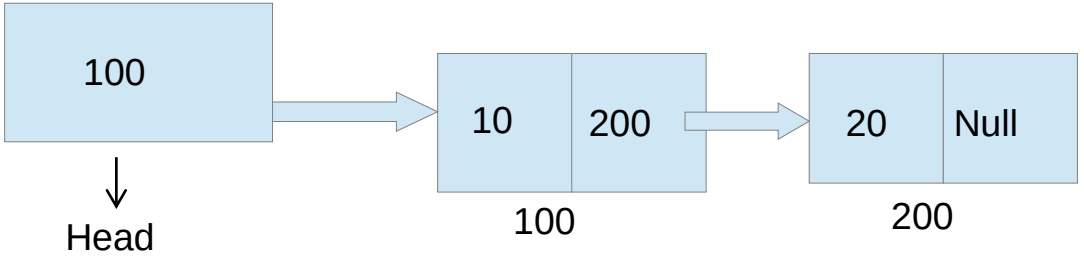
Example



Data Structure -Linked List

Introduction

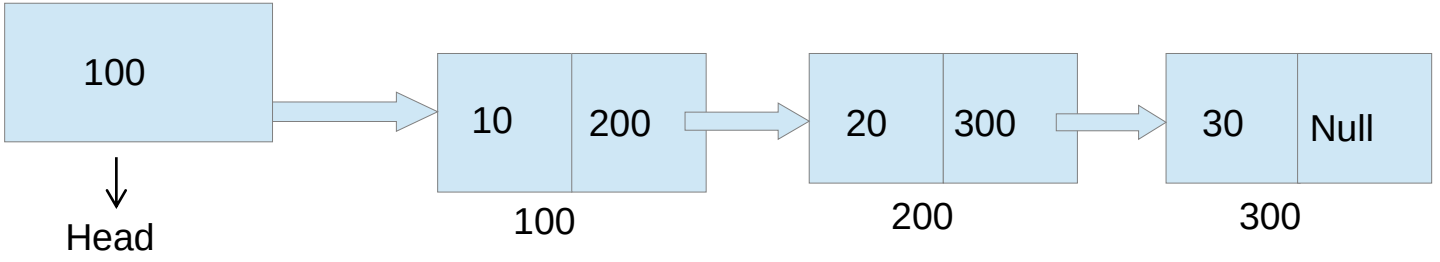
Example



Data Structure -Linked List

Introduction

Example



Introduction

Operations

- **Insertion**
- At last
- At First
- After a given element
- Before a given Element
- **Print List**
- **Deletion**
- At Last
- At first
- List
- **Reverse List**

Introduction

Types

1. Single Linked List
2. Double Linked List