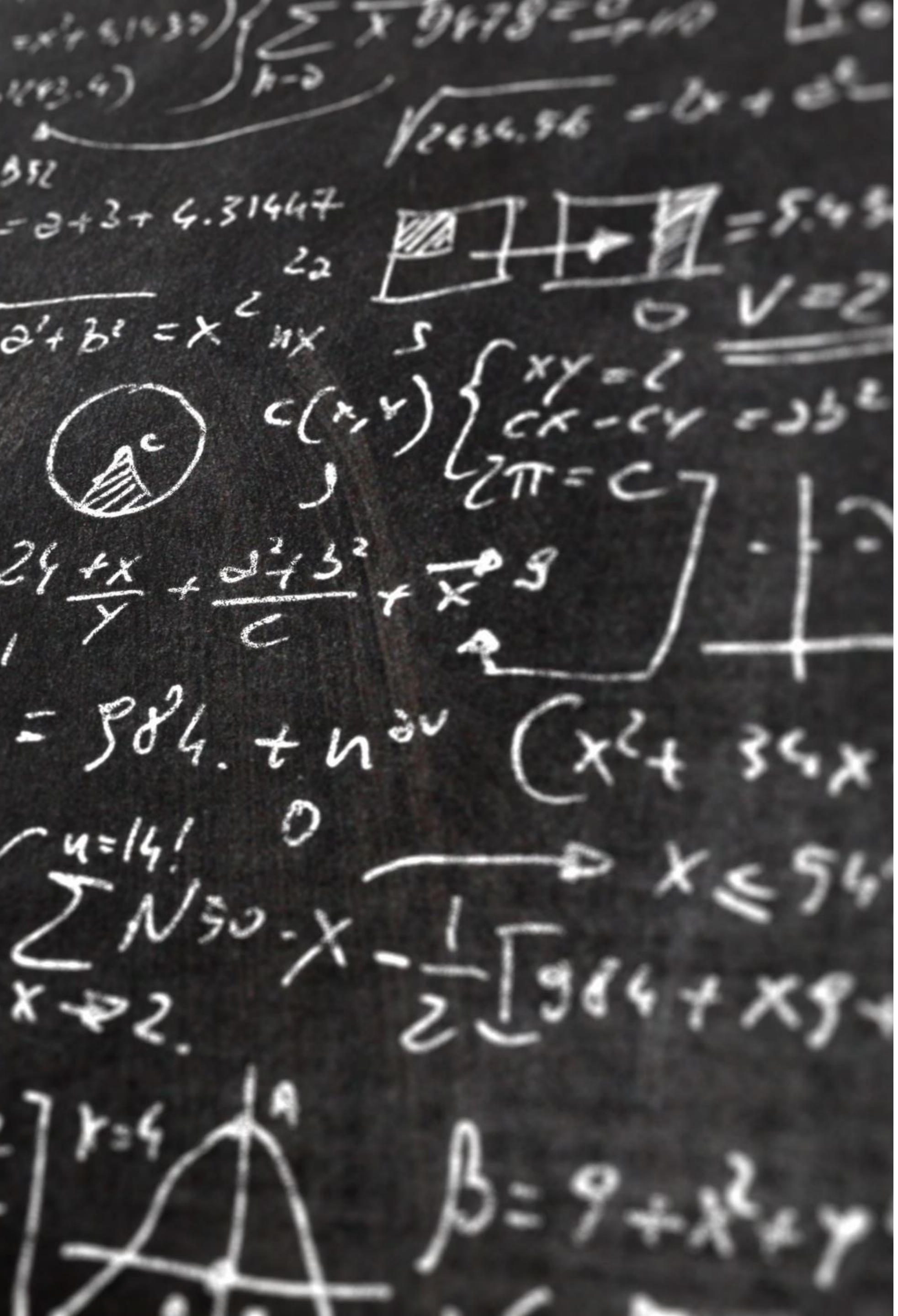
Telkom University

Implementasi Struktur Data 2022/2023

Pengantar Struktur Data

PROGRAM STUDI DIPLOMA III REKAYASA PERANGKAT LUNAK APLIKASI



Remember this?

Algorithm

- A clearly specified set of simple instructions to be followed to solve a problem

What is a good algorithm?

- The longest?
- The biggest?
- The scariest?
- The most efficient → running time, memory

Data Structure - Definition

When you stopped at the store this morning, you went to the back of a line to wait for the cashier.

Do you see a stack of books or a pile of papers on your desk? It’s easy to look at or remove the top. item of the stack or to add a new item to the top of the stack.

At your desk, you see your to-do list. Each entry in the list has a position that might or might not be important to you.

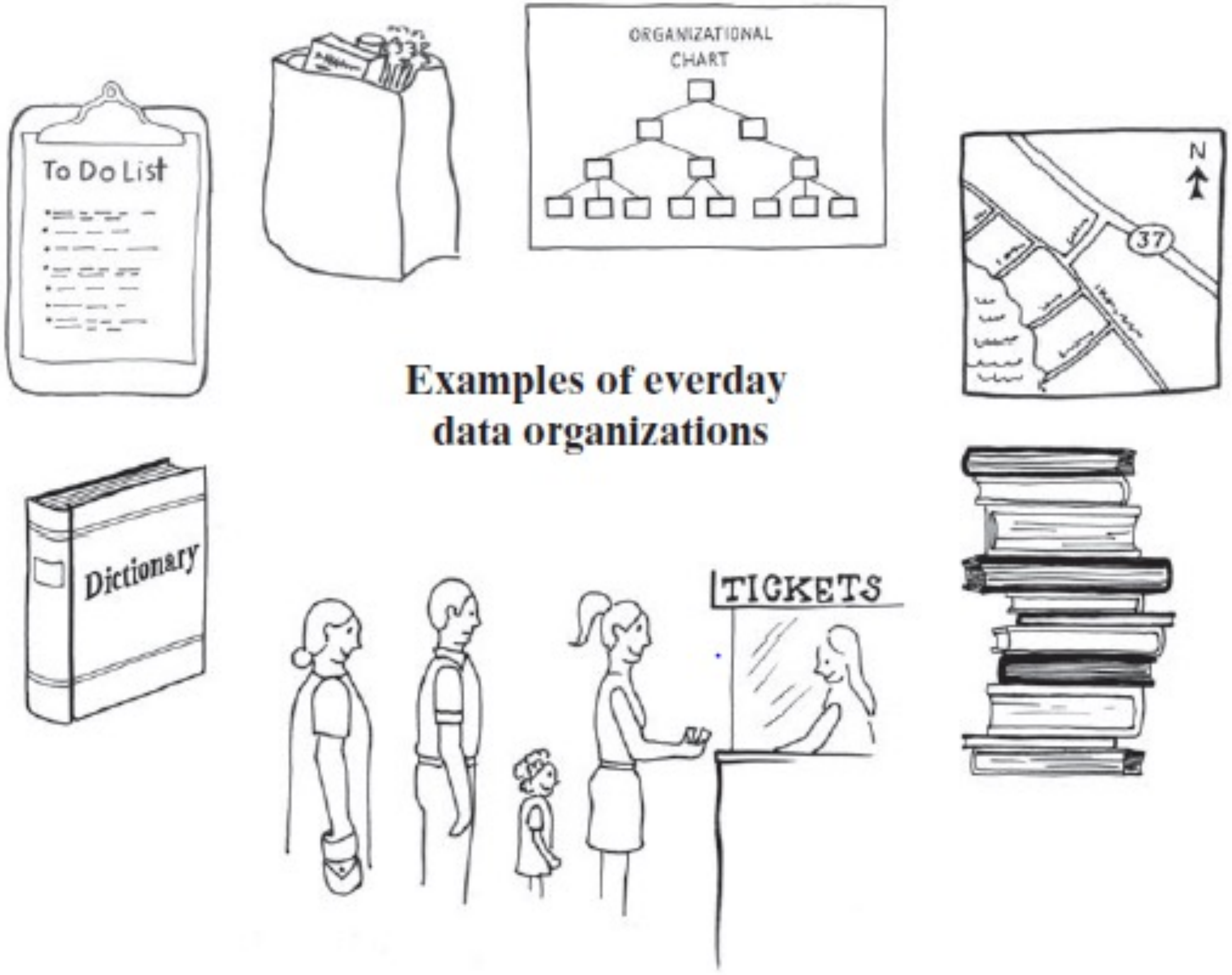
Data Structure - Definition

Your dictionary is an alphabetical list of words and their definitions.

Speaking of your computer, you have organized your files into folders, or directories. Each folder contains several other folders or files.

Finally, notice the road map that you are using to plan your weekend trip. The diagram of roads and towns shows you how to get from one place to another.

Data Structure - Definition



Examples of everday data organizations

A data structure is a ***way of organizing*** *input data and operations which can be performed on this data* (e.g. add, delete, find an element).

So, why data structure?

The data structures govern the space and time consumed by your running program.

In addition, large programs take time to write. Using different structures can have an impact on how long it takes to *write* your program.

Choosing the wrong structures can cause your program to run poorly or be difficult or impossible to implement effectively.

Algorithm + data structure = program

Example of Data Structure

ARRAY

LINKED LIST

STACK

QUEUE

TREE

GRAPH

Abstract Data Type (ADT)

DATA STRUCTURE IMPLEMENTATION

ADT - Overview

Computer programs also need to organize their data. Programs can use a list, a stack, a dictionary, and so on. These ways of organizing data are represented by abstract data types.

An **abstract data type**, or **ADT**, is a specification that describes a data set and the operations on that data.

Each ADT specifies what data is stored and what the operations on the data do. ADTs independently of any programming language it doesn’t indicate how to store / how to implement!

In contrast, a **data structure** is an implementation of an ADT within a programming language.

A **collection** is a general term for an ADT that contains a group of objects.

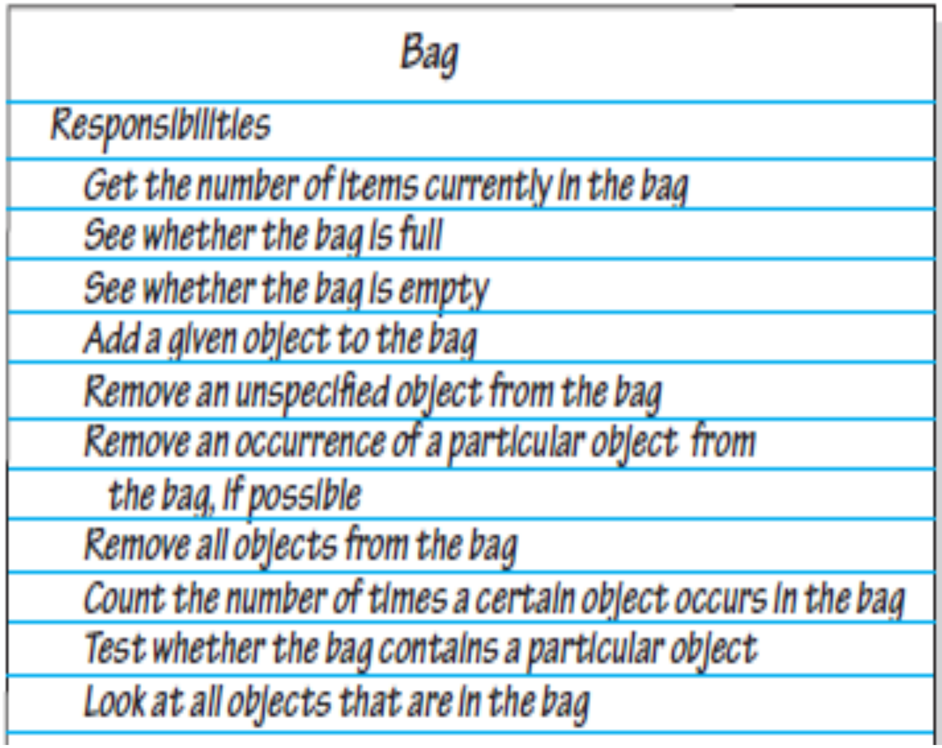
A **container** is a class that implements a collection. Some people use the terms “container” and “collection” interchangeably.

ADT - Overview

ADT, is a specification for a **group of values** and the **operations** on those values that is **defined conceptually and independently** of any programming language.

Imagine a paper bag, a reusable cloth bag, or even a plastic bag.

People use bags when they shop, pack a lunch, or eat potato chips. Bags contain things, but you don’t really care if things order exactly!



Bag
Responsibilities
Get the number of items currently in the bag
See whether the bag is full
See whether the bag is empty
Add a given object to the bag
Remove an unspecified object from the bag
Remove an occurrence of a particular object from the bag, if possible
Remove all objects from the bag
Count the number of times a certain object occurs in the bag
Test whether the bag contains a particular object
Look at all objects that are in the bag

Bag’s Behavior

Bag contains a finite number of objects, reporting how many objects it contains could be one of a bag’s behaviors:

- *Get the number of items currently in the bag*

Two related behaviors detect whether a bag is full or empty:

- *See whether the bag is full*
- *See whether the bag is empty*

We should be able to add and remove objects:

- *Add a given object to the bag*
- *Remove an unspecified object from the bag*
- *Remove an occurrence of a particular object from the bag, if possible*
- *Remove all objects from the bag*

So..?

Abstract data type (ADT) adalah kumpulan objek beserta operasinya. ADT merupakan suatu pemodelan, jadi pada pendifinisiannya tidak dituliskan bagaimana cara implementasi dari operasi ADT.

- Ingat perbedaan tipe data (primitive) dengan tipe data objek (reference)?

Tipe data memiliki operasi tersendiri (+, -, * dst), demikian juga ADT memiliki operasi-operasi tersendiri.

Then…

Pembuatan ADT terbagi dua:

- Pembuatan interface, yang berisi spesifikasi atau dokumentasi dari ADT.
- Implementasi ADT: bagaimana operasi tersebut dilakukan menggunakan struktur data dan perintah yang ada pada bahasa pemrograman tertentu (dalam hal ini, Java)

ADT hanya berisi spesifikasi atau detil dari interface, bukan implementasinya. Pemisahan ini penting, karena user hanya perlu tahu apa saja operasinya, bukan bagaimana cara operasi tersebut dilakukan.

If it’s still confusing…

Using ADT is Like Using Vending Machine!

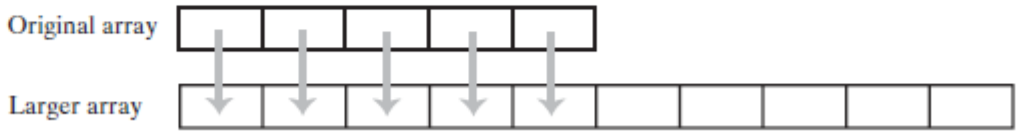
List

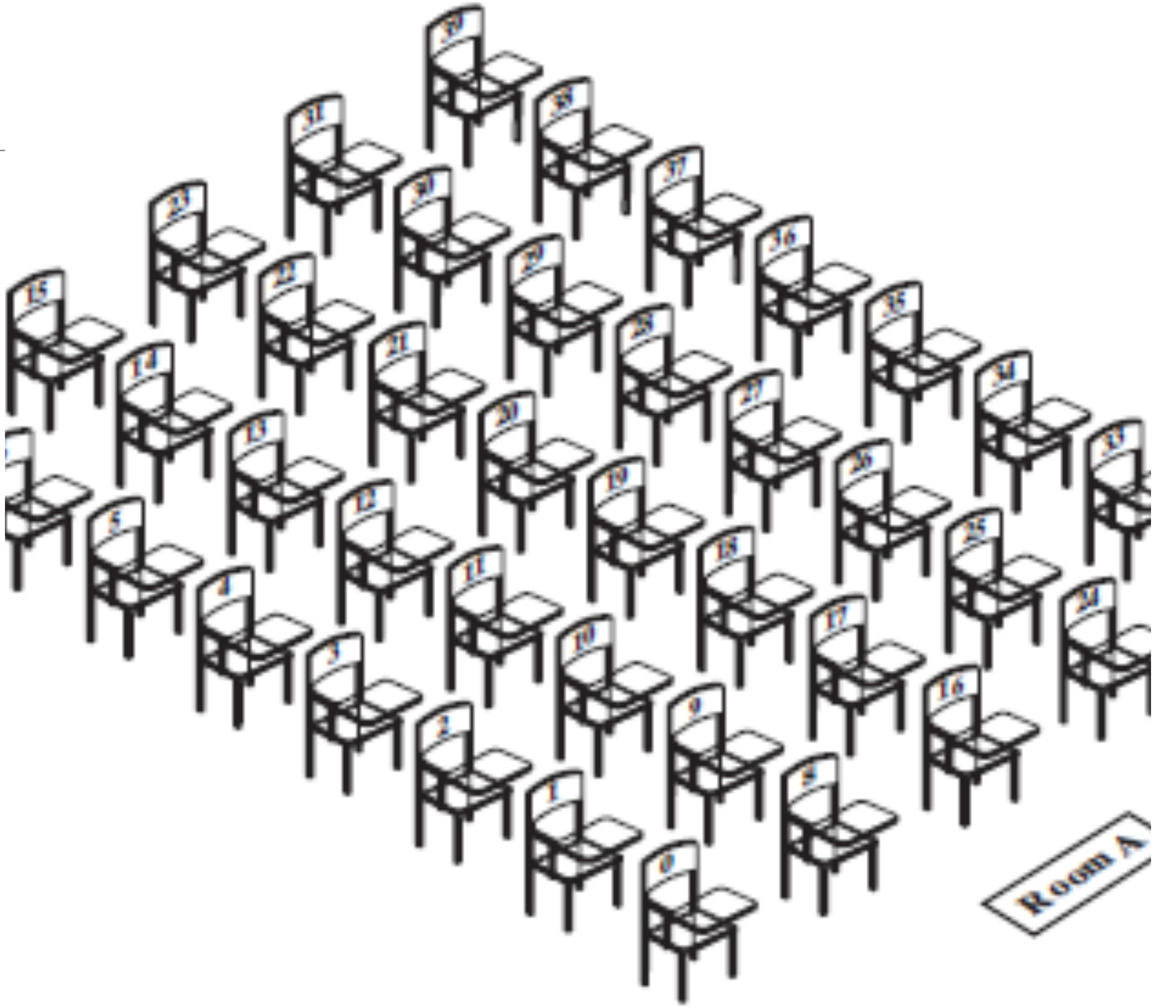
IMPLEMENTASI STRUKTUR DATA

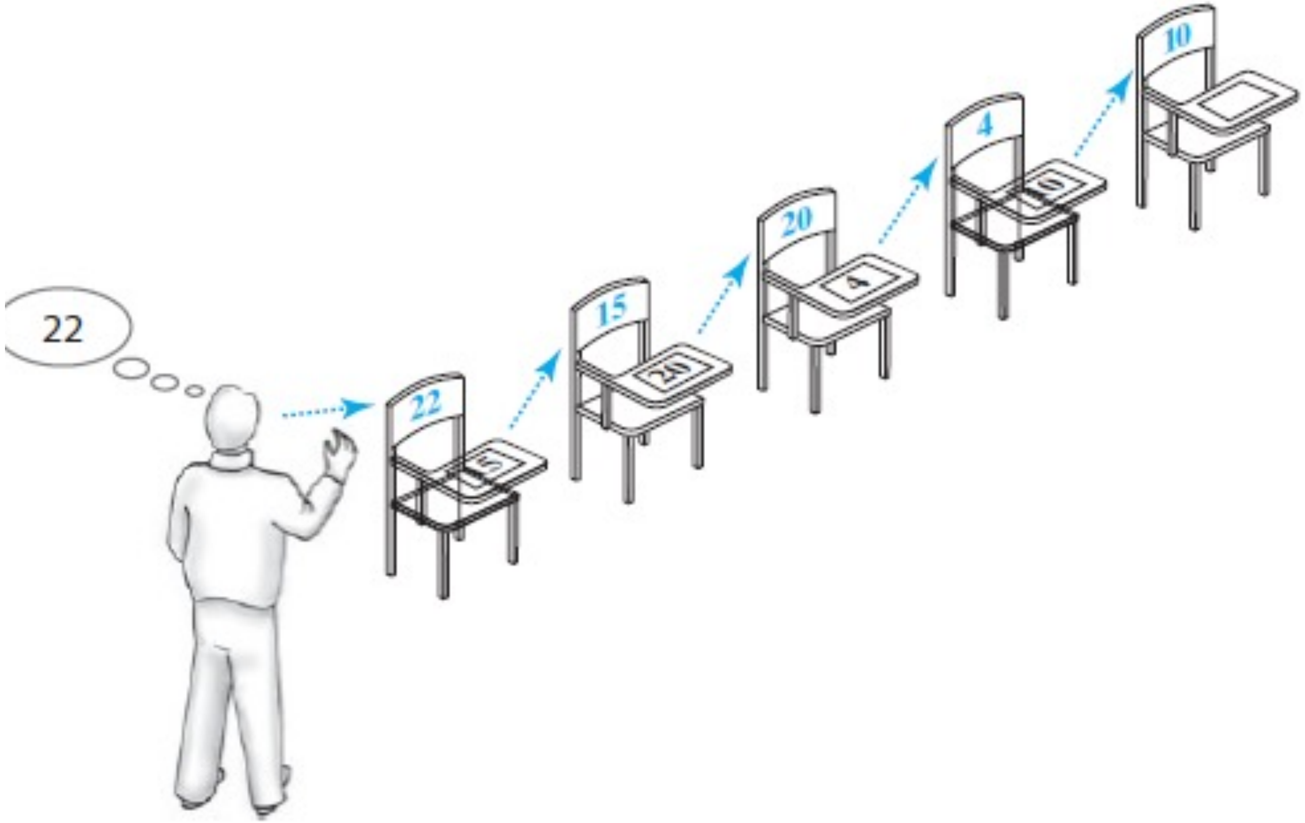
ADT with Array

Array has a fixed size, and so it can either become full or have several unused elements.

Resizing an array -> move data each time to expand an array







ADT LIST

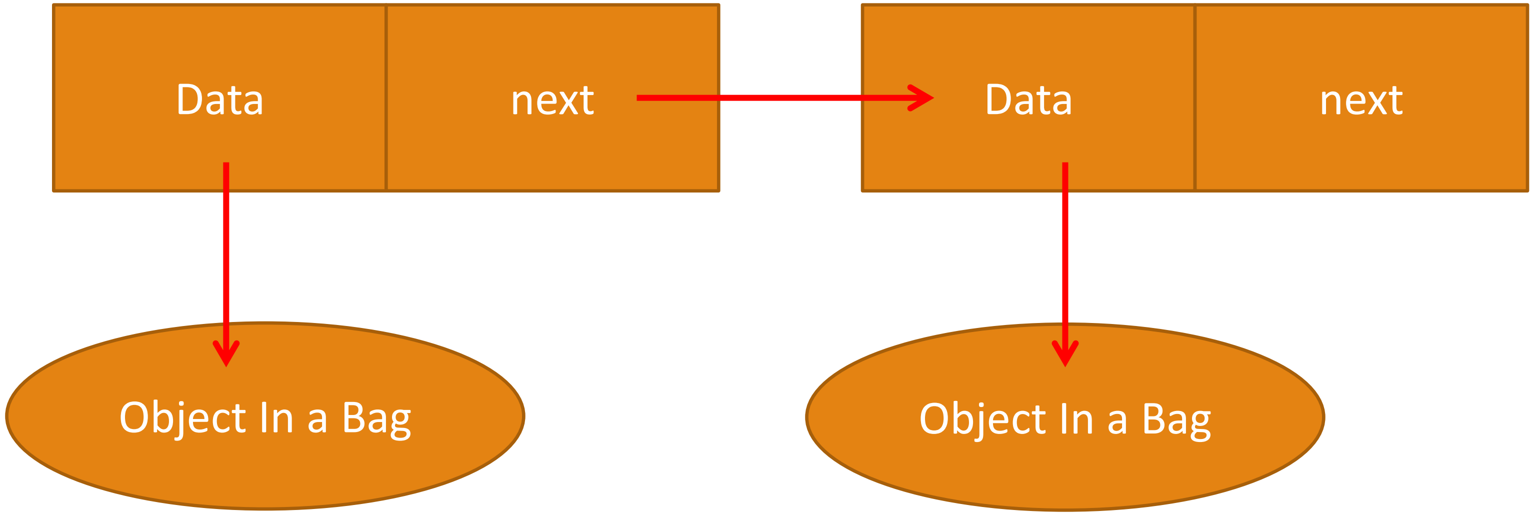
Data organization that uses memory only as needed for a new entry and returns the unneeded memory to the system after an entry is removed.

By linking data, its avoids moving data when adding or removing entries.

Linked list and array (conventional)

Array	List
Alokasi memory bersifat statis	Alokasi memory bersifat dinamis
Lokasi memory continue (fisik dan logik terurut)	Lokasi pada memori random (fisik dapat terpisah, logik berkaitan)
Operasi pengubahan susunan data relatif memakan waktu	Operasi pengubahan susunan data lebih mudah dan ringkas
Akses data lebih mudah (menggunakan indeks)	Akses data lebih sulit (menggunakan bantuan)

LInkED LIST



A **linked list** is a data structure used for collecting a sequence of objects that allows efficient addition and removal of elements in the middle of the sequence.

A linked list consists of a number of nodes, each of which has a reference to the next node. We call this the next link. The last cell's next link references null.

Example – ADT Buku(1)

Spesifikasikan terlebih dahulu operasi yang dapat dilakukan pada buku (dengan mengingat karakteristik buku)

- Misal, karakteristik buku: Memiliki judul, pengarang, tahun terbit, dan penerbit.

Berdasarkan karakteristik yang ada, tentukan operasi yang bisa dilakukan pada data buku

- Memasukkan buku baru
- Menghapus buku
- Menghitung banyaknya data buku
- Mencari buku tertentu

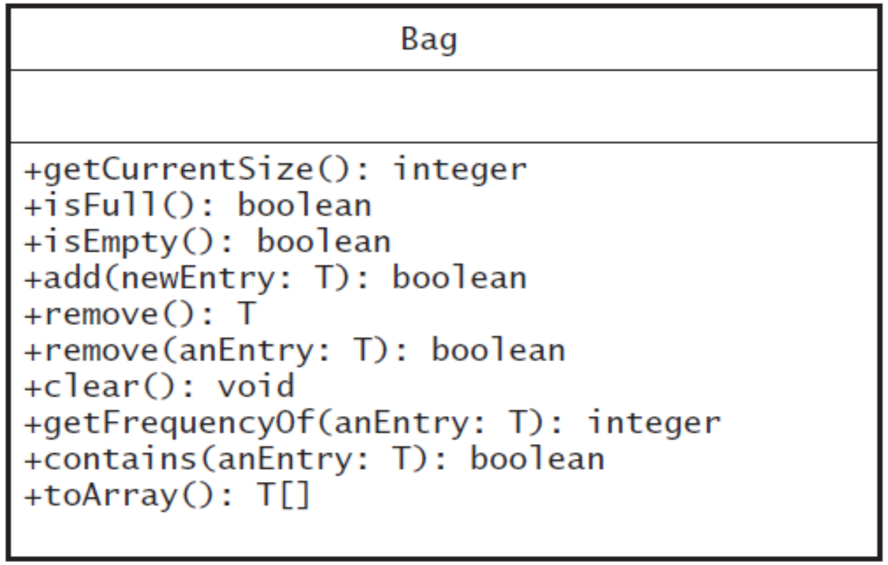
Example – ADT Buku (2)

Sebelum implementasi, tentukan method berdasarkan spesifikasi yang ada

- Beri nama method tersebut, parameter masukannya, tipe data kembalian, dan komentar jika diperlukan
- Bisa menggunakan notasi UML untuk memudahkan

Ketika mengimplementasikan dalam program kelak, spesifikasi bagi ADT dapat dipisahkan pada interface.

Masih ingat?



```
Bag

+getCurrentSize(): integer
+isFull(): boolean
+isEmpty(): boolean
+add(newEntry: T): boolean
+remove(): T
+remove(anEntry: T): boolean
+clear(): void
+getFrequencyOf(anEntry: T): integer
+contains(anEntry: T): boolean
+toArray(): T[]
```

Source: Carrano, 2001

Bagaimana dengan ADT Buku yang sudah kita buat sebelumnya?

Specification of ADT List

- Penambahan data (biasanya dilakukan pada akhir list, tapi pada dasarnya dapat dilakukan di awal, di akhir dan di tengah-tengah)
- Menghapus semua data bku
- Mengganti suatu data
- Melihat data tertentu
- Melihat isi keseluruhan data
- Mencari data tertentu pada list
- Mencari tahu apakah list kosong atau tidak

Linked List (Cont’l)

SINGLE LINKED LIST

DOUBLE LINKED LIST

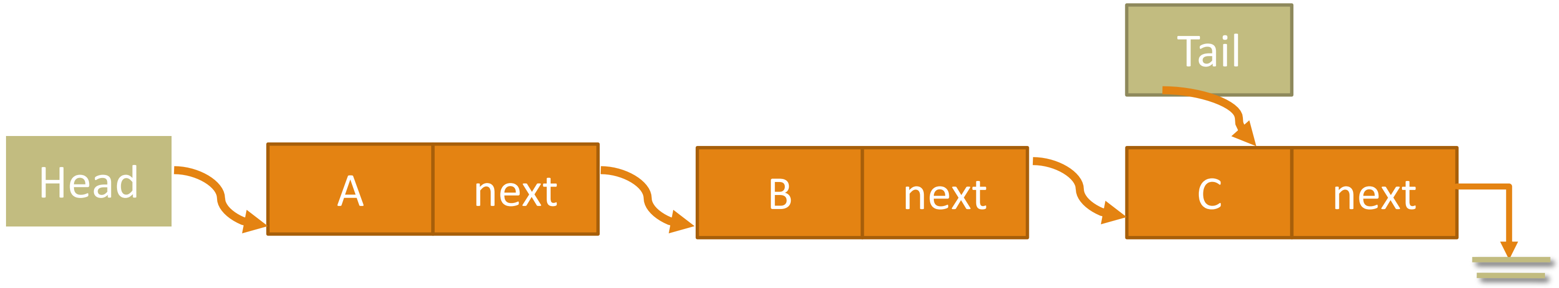
CIRCULAR LINKED LIST

Singly Linked List

Bentuk linked list yang paling sederhana

Objek linked list (head/ first) memiliki reference ke node pertama dari linked list

Merupakan list satu arah: setiap node memiliki reference ke node selanjutnya pada linked list



Operasi pada Singly Linked List

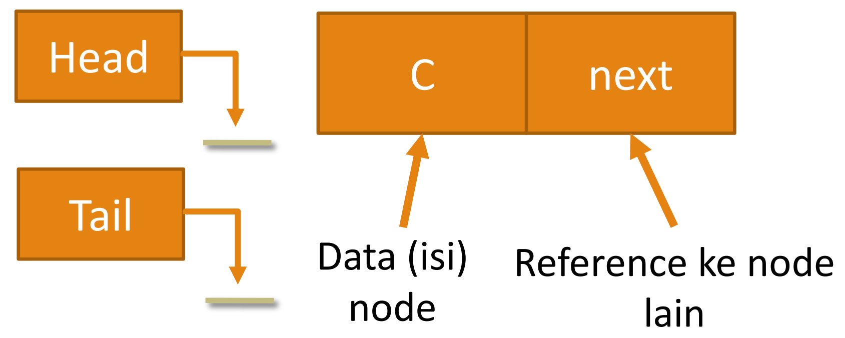
Insert (Sisip) :

- Sisip Depan (Insert First)
- Sisip Belakang (Insert Last)
- Sisip Tengah (Insert At Specific Position)

Delete (Hapus):

- Hapus Depan (Delete First)
- Hapus Belakang (Delete Last)
- Hapus Tengah (Delete At Specific Position)

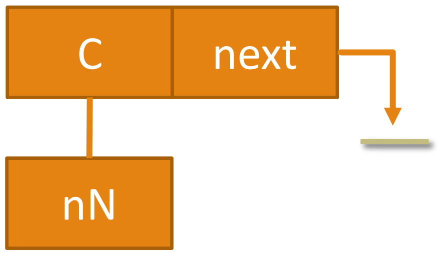
Sisip Depan



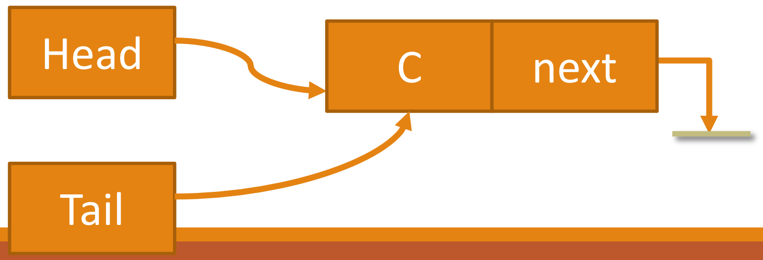
Head = Tail = Null

Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya

Node dengan isi "C" akan dimasukkan pada linked list



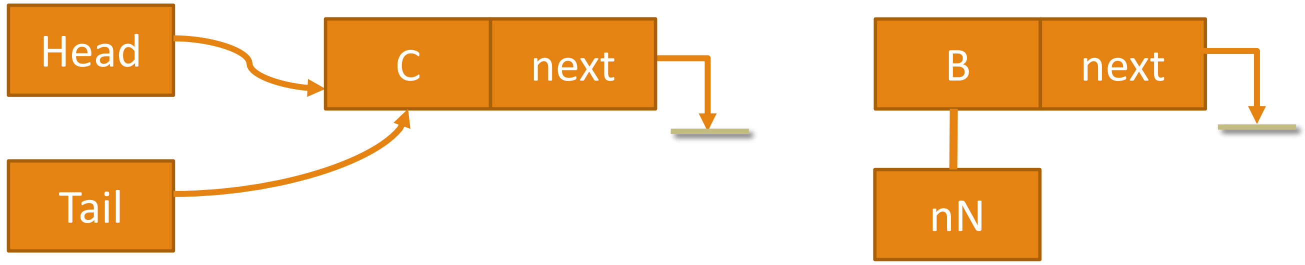
Keadaan 1:
Linked List masih kosong, dilakukan sisip Depan

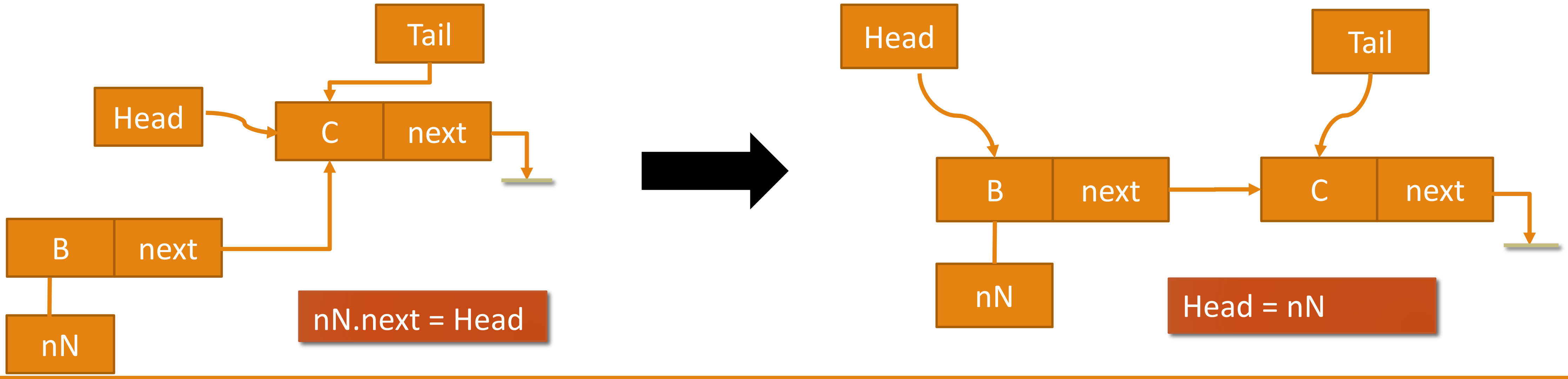


Head = Tail = nN

Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya

Keadaan 2:
Sisip depan pada linked list yang memiliki isi

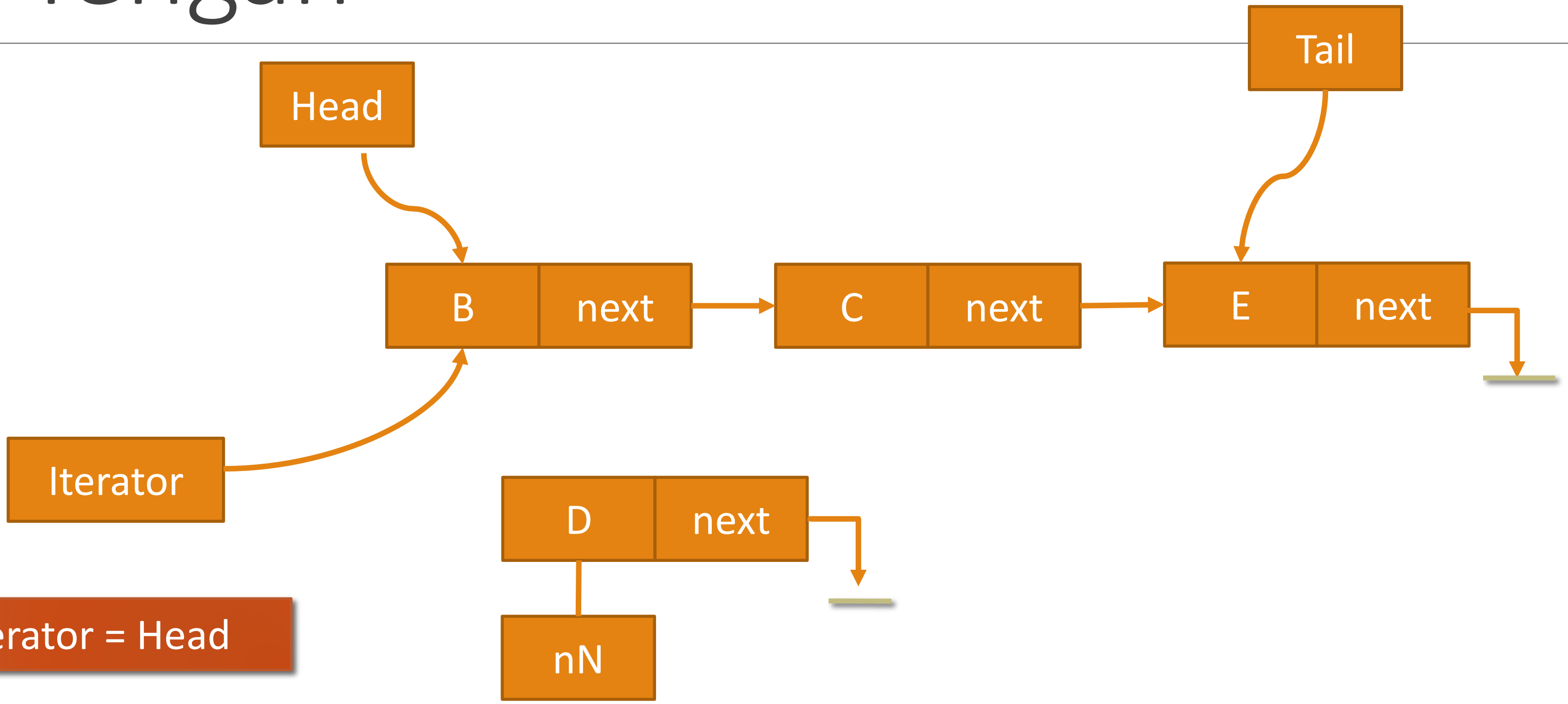




nN.next = Head

Head = nN

Sisip Tengah



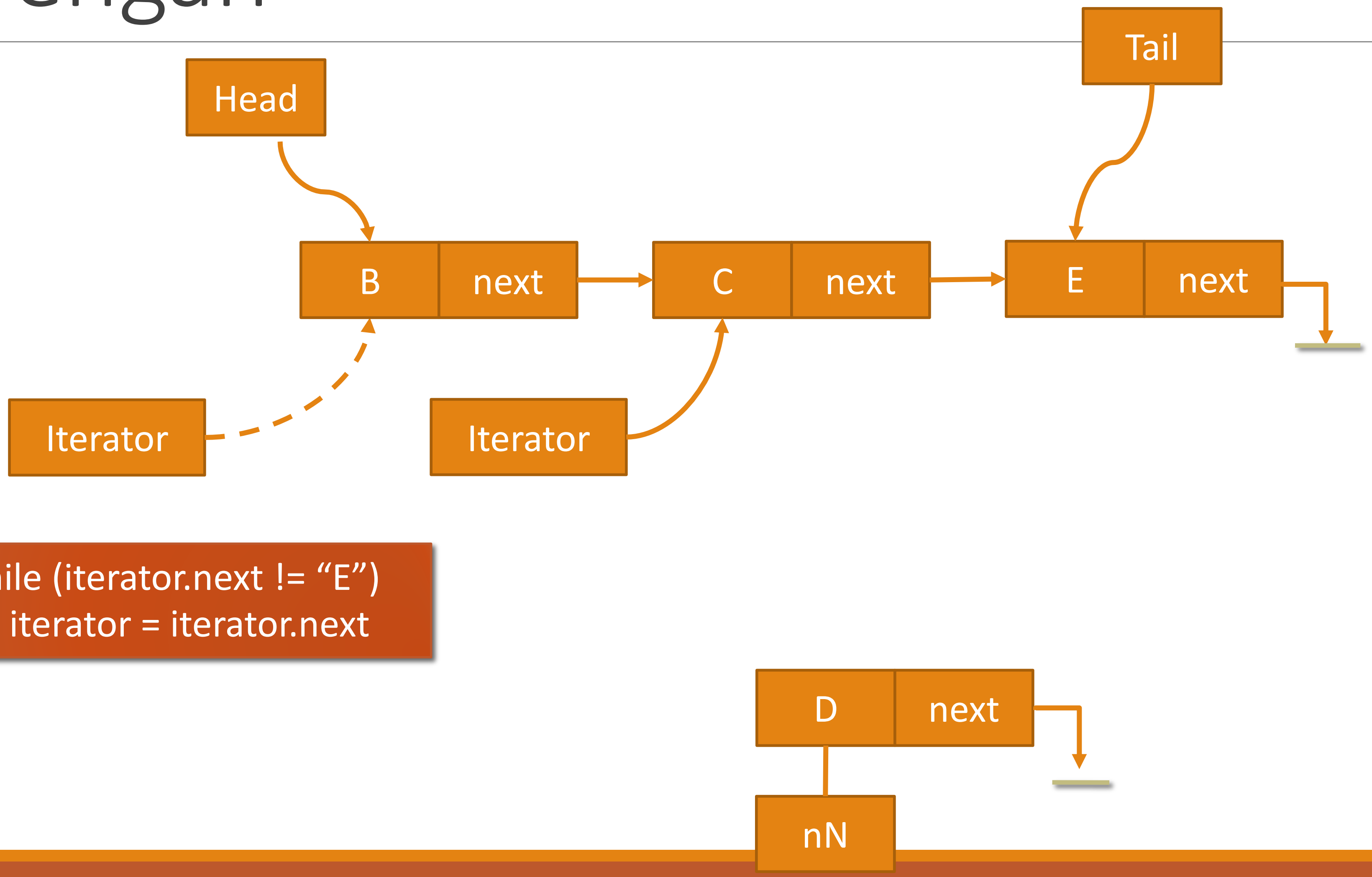
Iterator = Head

Sisipkan D antara C dan E

Posisi yang tepat??

Bagian next (iterator) adalah node E

Sisip Tengah



Head

Tail

B next

C next

E next

Iterator

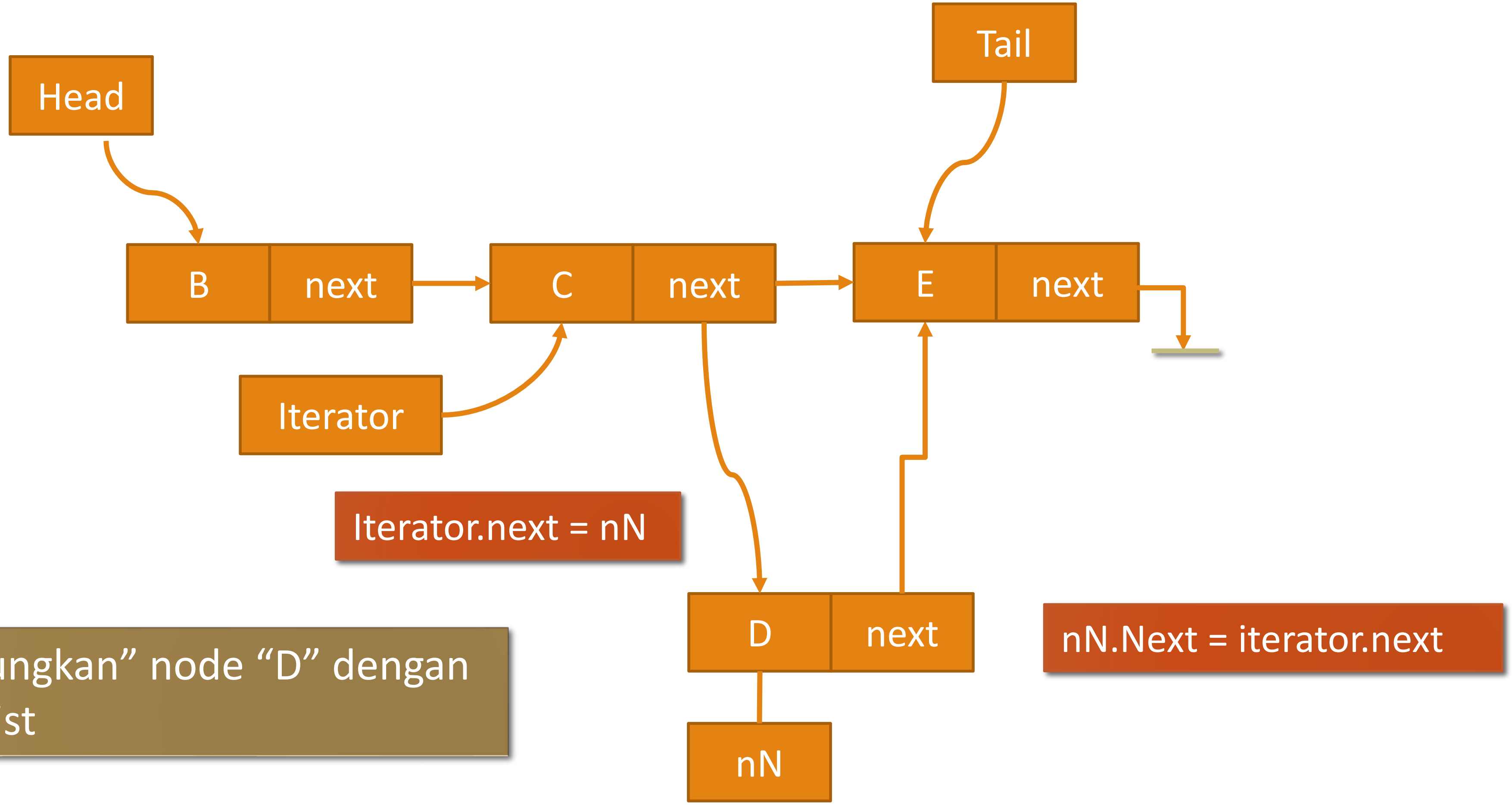
Iterator

D next

nN

```
while (iterator.next != “E”)
    iterator = iterator.next
```

Sisip Tengah

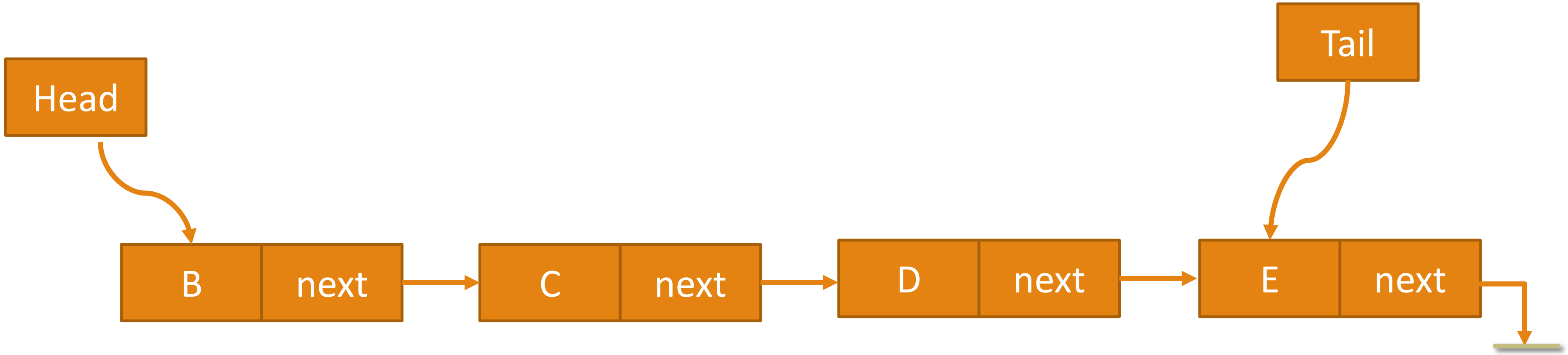


Iterator.next = nN

nN.Next = iterator.next

“Sambungkan” node “D” dengan linked list

Sisip Tengah

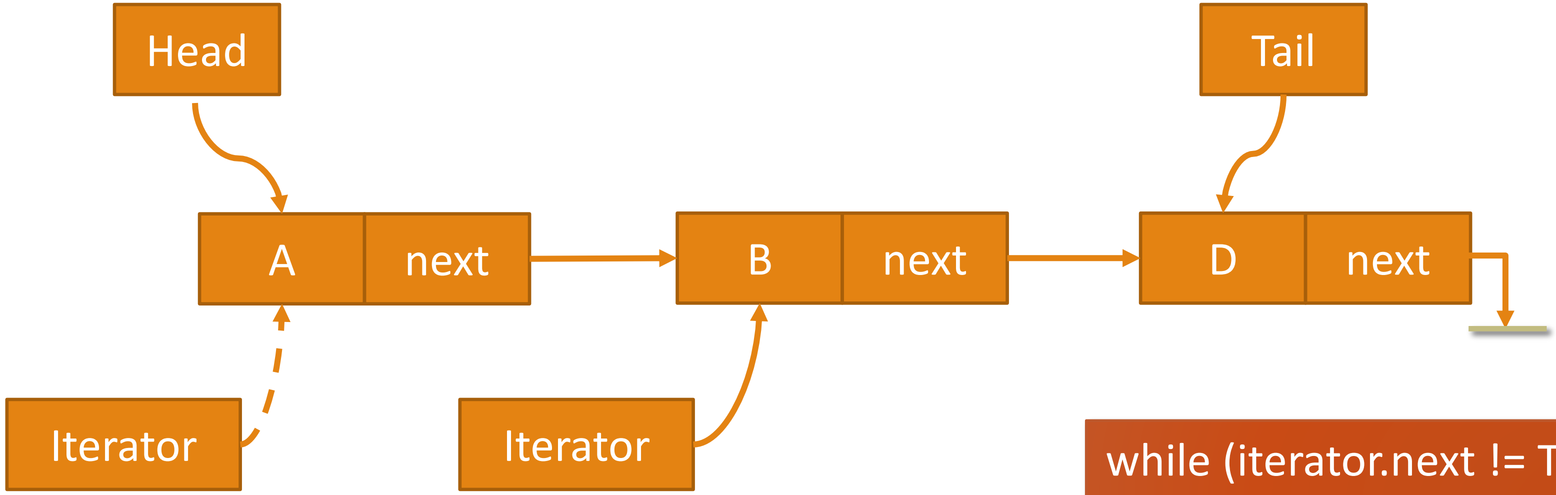


Hapus Belakang

Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya

Pindahkan Tail dari node paling belakang

Gunakan iterator untuk "memegang" node sebelum Tail



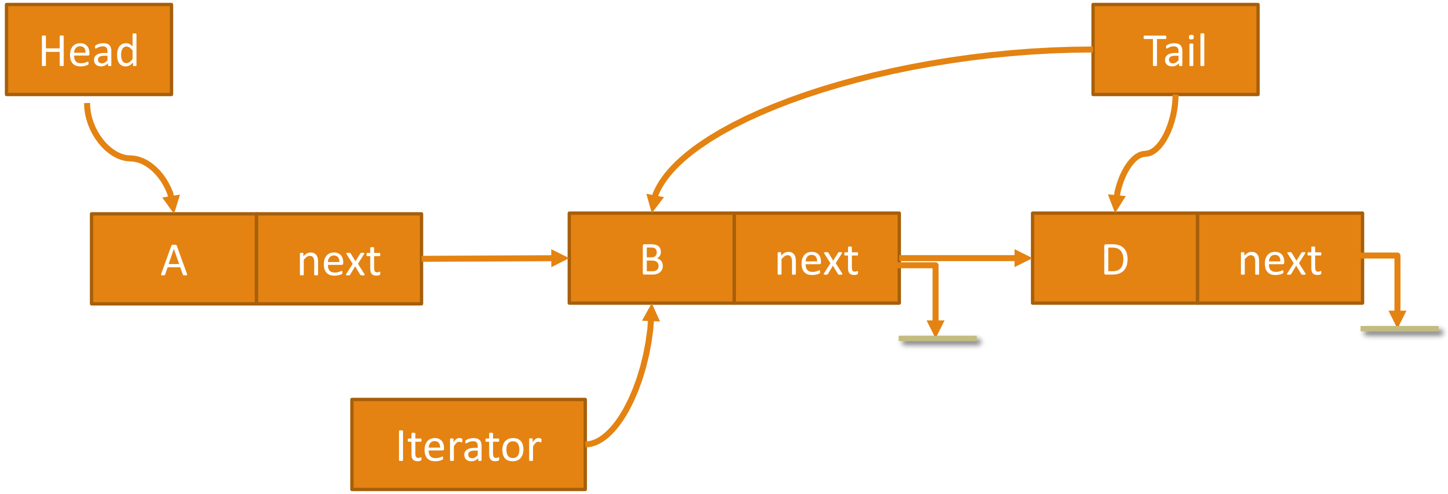
```
while (iterator.next != Tail)
    iterator = iterator.next
```

Hapus Belakang

Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya

Pindahkan Tail dari node paling belakang

Gunakan iterator untuk "memegang" node sebelum Tail



Tail = iterator

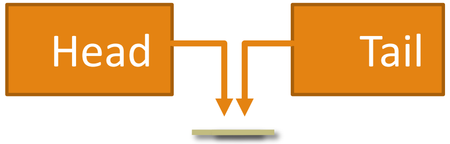
Tail.next = null

DOUBLY LINKED LIST

Sisip depan

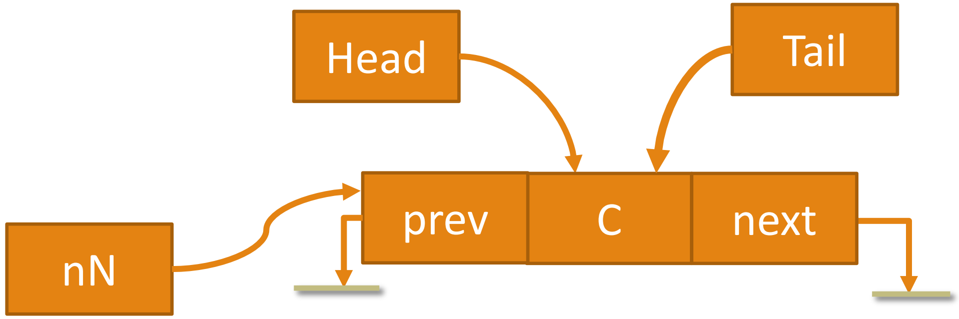
Keadaan 1:
Linked List masih kosong, dilakukan sisip Depan

Linked List masih kosong, head dan tail menunjuk ke NULL



Head = Tail = Null

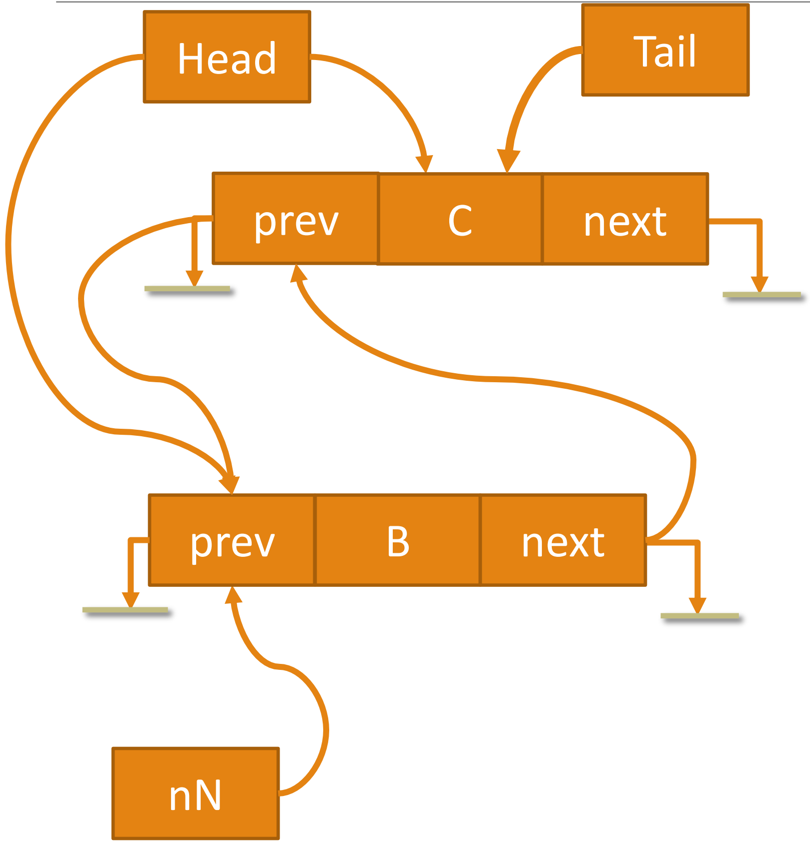
Ketika disisipkan node baru, maka head dan tail akan menunjuk node tersebut.



Head = Tail = nN

Sisip depan

Keadaan 2:
Sisip depan pada saat linked list memiliki isi

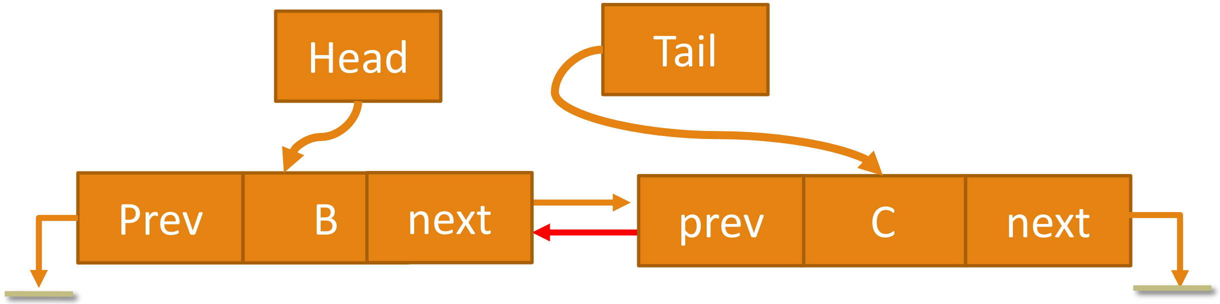


nN.next = Head

Head.prev = nN

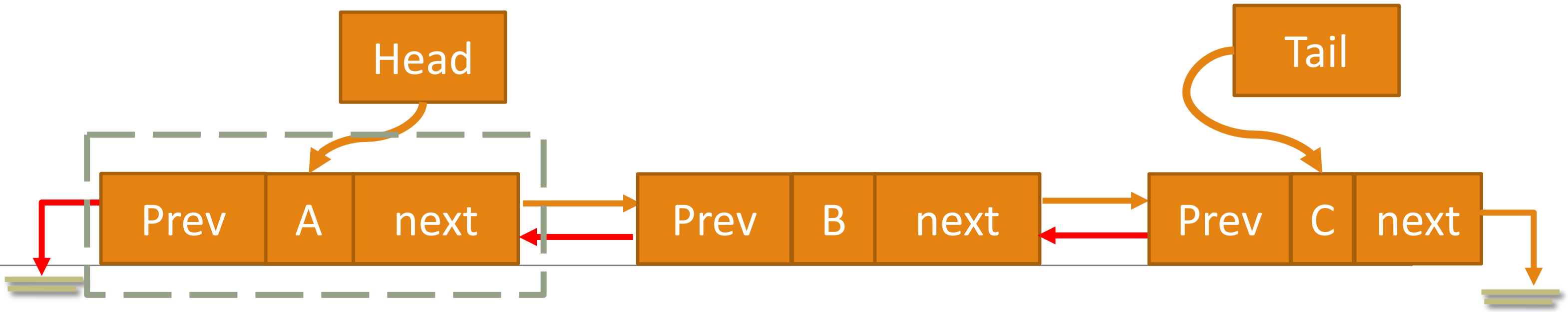
Head = nN

Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya



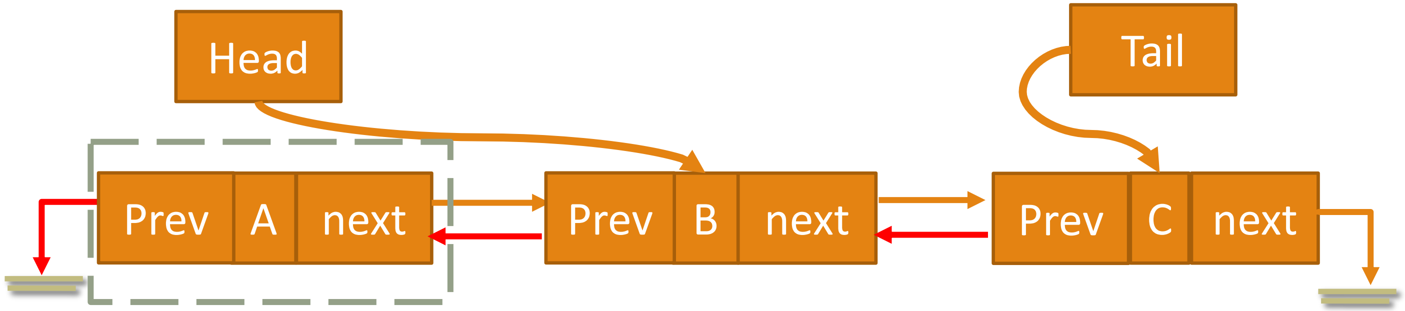
Hapus depan

1. Buat sebuah temporary yang menunjuk pada head



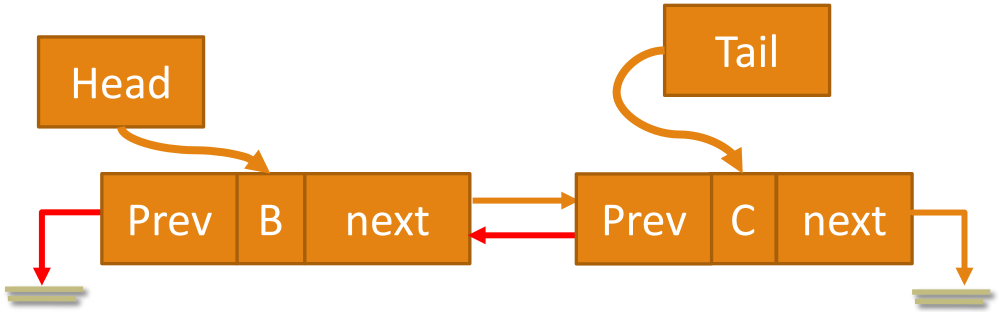
2. Pindahkan head ke node kedua

```
Head = Head.next
```



3. Buat bagian head previous menjadi NULL,
Temporary next menjadi null

```
Head.prev = null
```



Pastikan agar Head selalu berada di awal linked list, dan Tail di akhirnya

Reference

Carrano, Frank., *Data Structure and Abstraction,* Prentice Hall

Weiss ,M. A., *Data Structures and Algorithm Analysis in Java 3rd Ed,* Pearson Education Inc.

Adams, B. G., *Introduction To Computer Science: An Object-oriented Approach Using Java 5,* BlueJ and BeanShell Edition.