TIB29 – Struktur Data dan Algoritma

PERINGATAN HAK CIPTA

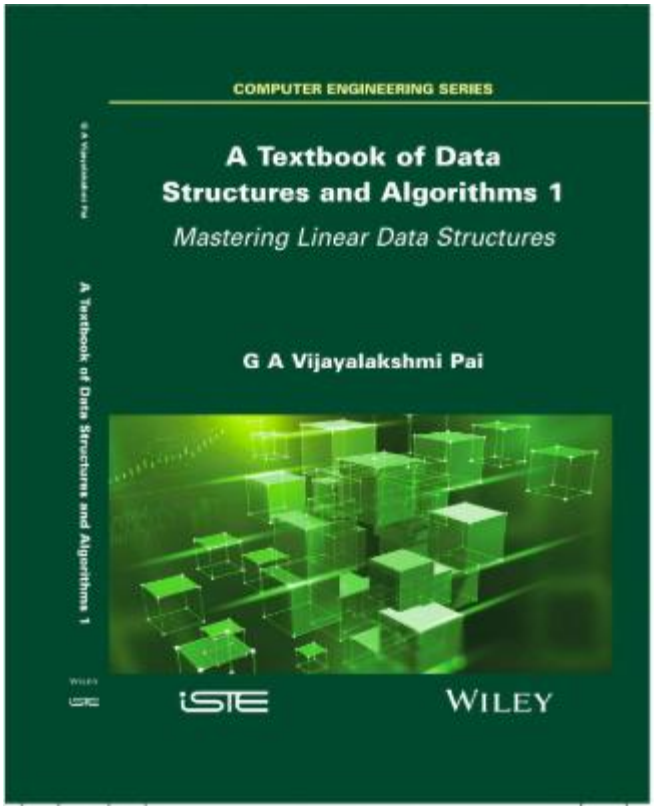
PERINGATAN HAK CIPTA

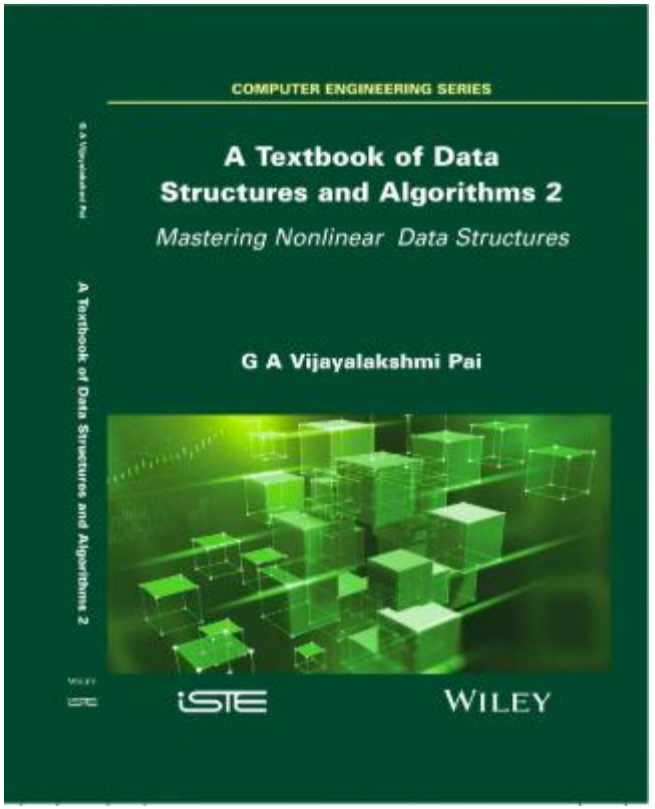
Capaian Pembelajaran Mata Kuliah

1. Mahasiswa mampu menjelaskan konsep dan teori mengenai struktur data dan algoritma. (C2, A2)
2. Mahasiswa mampu menerapkan konsep struktur data dan algoritma dalam pemrograman. (C3, A3)

Pengenalan Struktur Data

Diadopsi Dari Sumber:





Sub-CPMK

- Mahasiswa mampu menjelaskan konsep dasar struktur data beserta pembentukannya. (C2, A2)

Materi

1. Konsep dasar struktur data
2. Tipe Data Abstrak

1. Konsep Dasar Struktur Data

1.1 Pengertian Struktur Data

- Struktur data adalah cara penyimpanan, penyusunan dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien.
- Struktur data berarti tata letak data yang berisi kolom-kolom data, baik itu kolom yang tampak oleh pengguna (*user*) ataupun kolom yang hanya digunakan untuk keperluan pemrograman yang tidak tampak oleh pengguna.

1.1 Pengertian Struktur Data (Lanj.)

- Setiap baris dari kumpulan kolom-kolom tersebut dinamakan catatan (*record*).
- Lebar kolom untuk data dapat berubah dan bervariasi. Ada kolom yang lebarnya berubah secara dinamis sesuai masukan dari pengguna, dan juga ada kolom yang lebarnya tetap.

1.2 Manfaat Struktur Data

- Manfaat dari struktur data antara lain:
  - Mengoptimalkan pengorganisasian data dalam *memory*
  - Membantu dalam proses yang rumit

1.3 Bentuk Umum Struktur Data

Struktur Data disusun dalam dua basis yaitu

- *Array Base*

  Array Base merupakan representasi struktur data menggunakan *Array*

- *Linked List Base*

  Linked-List Base merupakan representasi struktur data menggunakan *Linked-list*

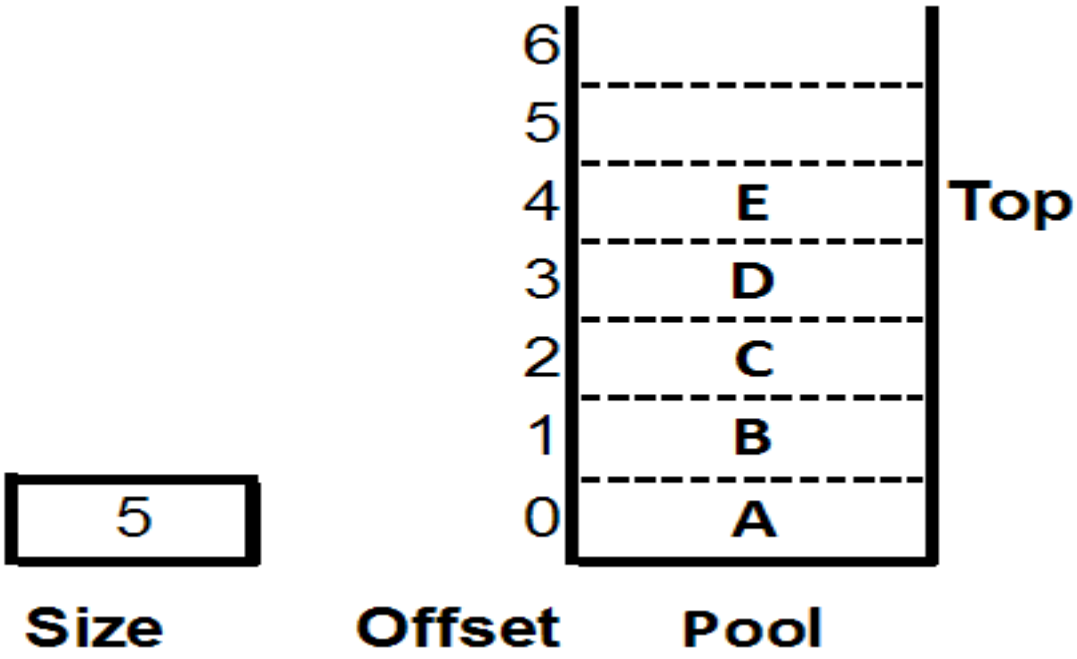
1.3.1 Struktur Data *Array Based*

- Struktur data yang diterapkan dalam bentuk *array*
- Penerapan *Array Based*:
  - *Stack*
  - *Queue*
  - *Graph*

  Contoh dapat dilihat pada masing-masing pembahasan pada materi berikutnya

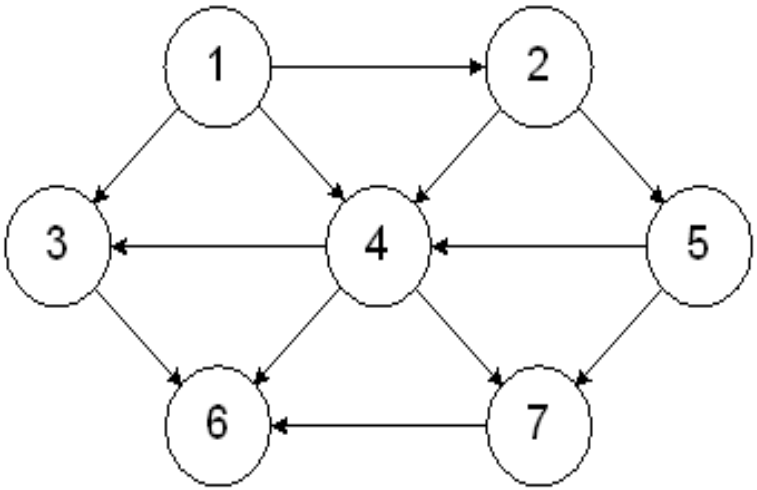
1.3.1 Struktur Data *Array Based (Lanj.)*

Contoh Array Base Stack:



1.3.1 Struktur Data *Array Based (Lanj.)*

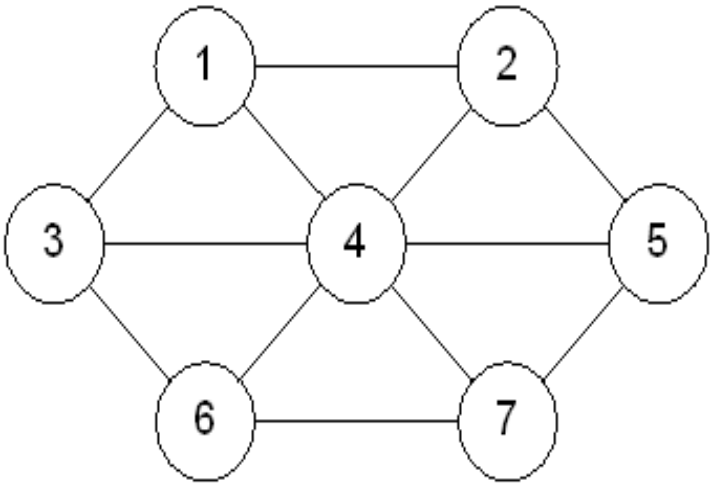
Contoh Array Base Graph untuk Directed Graph:



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[1]	0	1	1	1	0	0	0
[2]	0	0	0	1	1	0	0
[3]	0	0	0	0	0	1	0
[4]	0	0	1	0	0	1	1
[5]	0	0	0	1	0	0	1
[6]	0	0	0	0	0	0	0
[7]	0	0	0	0	0	1	0

1.3.1 Struktur Data *Array Based (Lanj.)*

Contoh Array Base Graph Untuk Undirected Graph:



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[1]	0	1	1	1	0	0	0
[2]	1	0	0	1	1	0	0
[3]	1	0	0	1	0	1	0
[4]	1	1	1	0	1	1	1
[5]	0	1	0	1	0	0	1
[6]	0	0	1	1	0	0	1
[7]	0	0	0	1	1	1	0

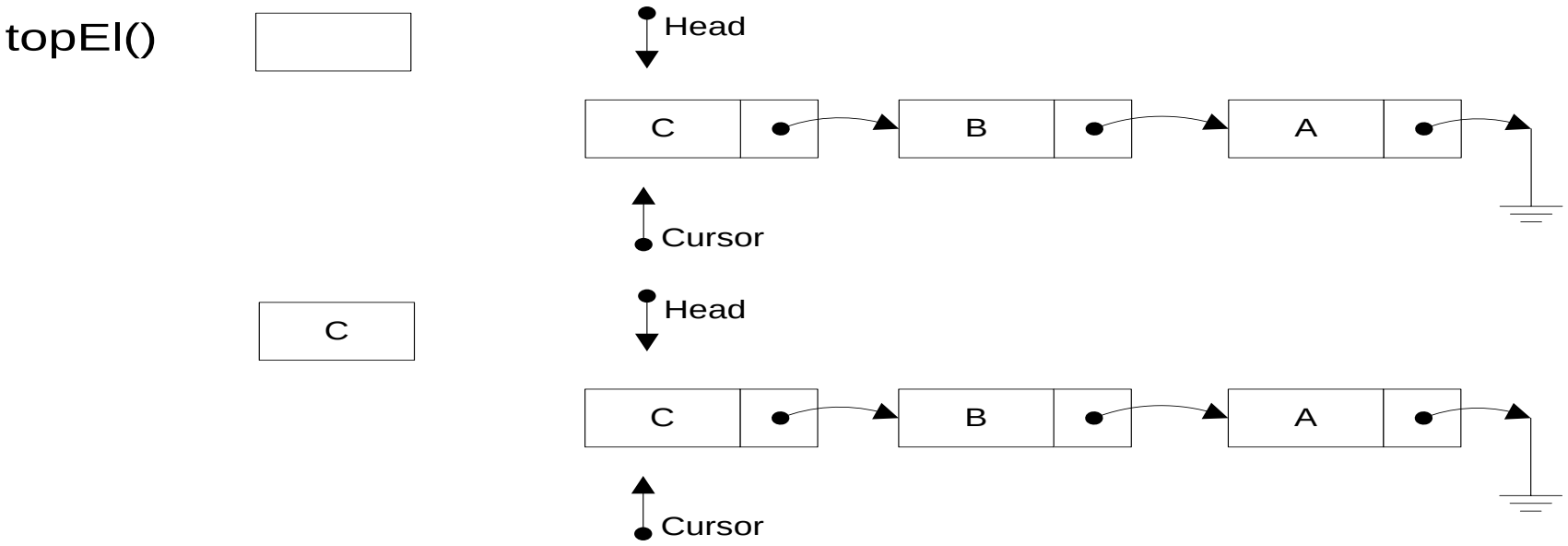
1.3.2 Struktur Data *Linked-List Based*

- Penerapan struktur data dalam bentuk *linked list*
- Penerapan *Linked-List Based*
  - *Stack* / Tumpukan
  - *Queue* / Antrian
  - *Tree* / Pohon
  - *Graph*

  Contoh dapat dilihat pada masing-masing pembahasan pada materi berikutnya

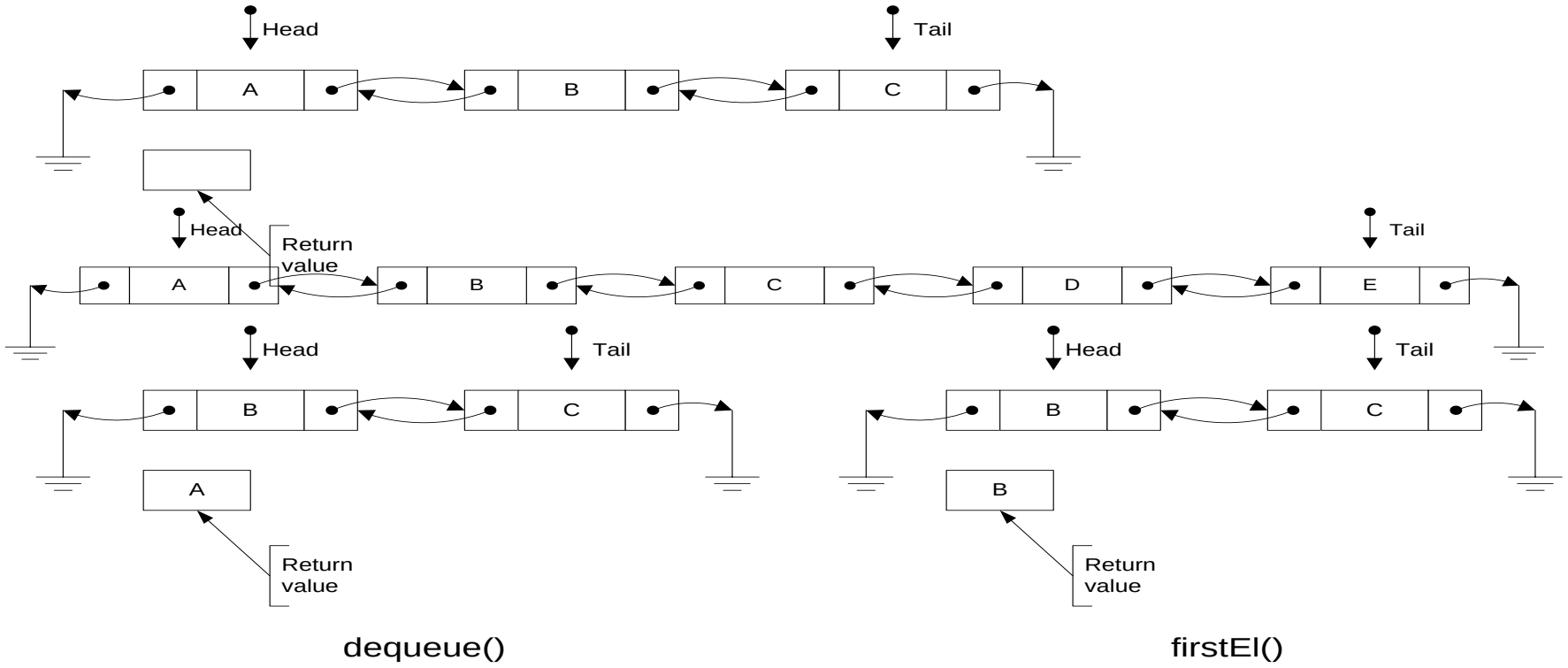
1.3.2 Struktur Data *Linked-List Based (Lanj.)*

Contoh Linked-List Base Stack:



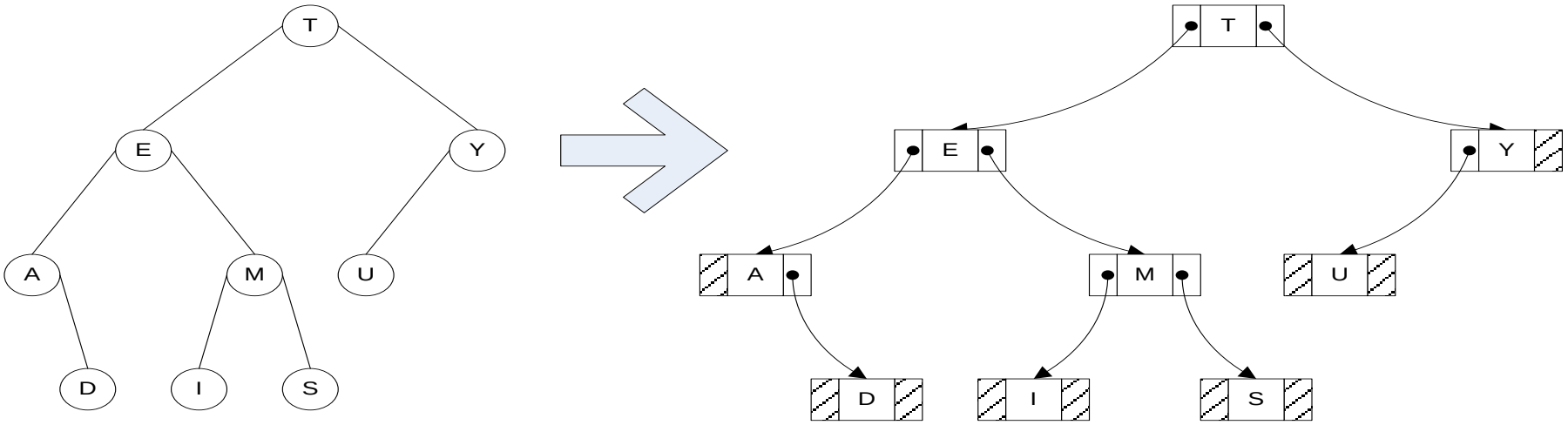
1.3.2 Struktur Data *Linked-List Based (Lanj.)*

Contoh *Linked-List Base Queue:*



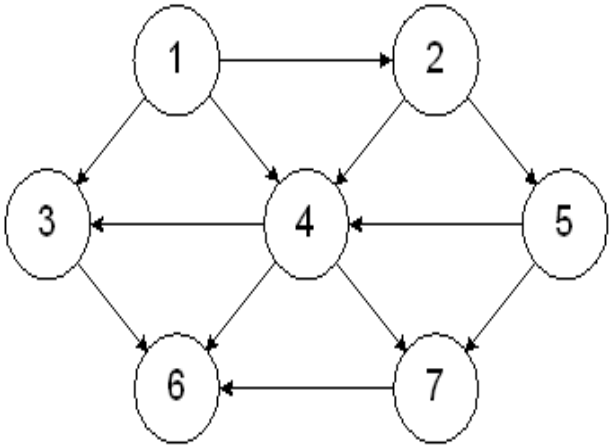
1.3.2 Struktur Data *Linked-List Based (Lanj.)*

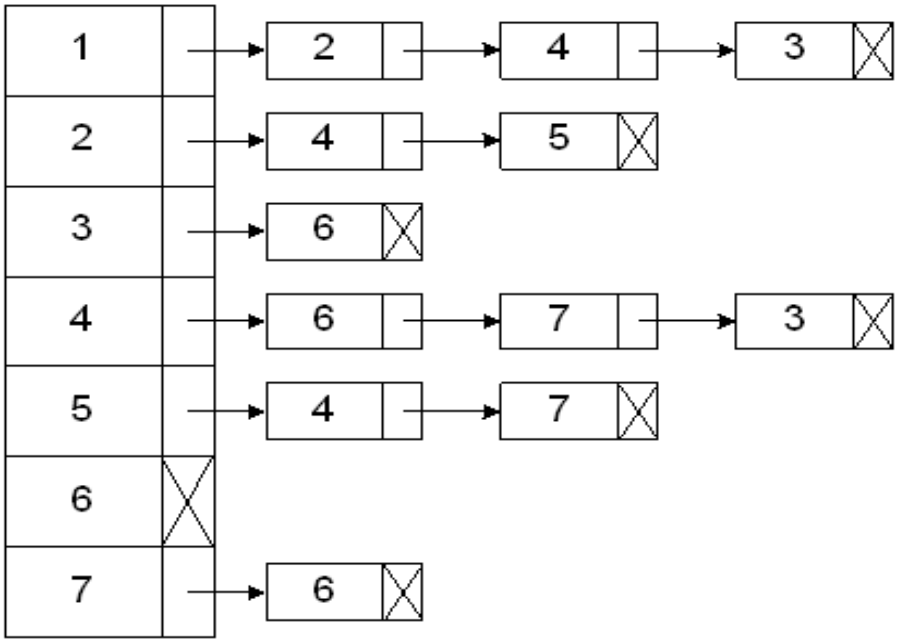
Contoh *Linked-List Base Tree:*



1.3.2 Struktur Data *Linked-List Based (Lanj.)*

Contoh *Linked List Base Graph:*





1.3.3 Kegunaan Struktur Data

- Kegunaan struktur data adalah membantu mempermudah proses,
- Tipe Data Abstrak dalam hal ini berperan, karena keteraturannya berupa penyimpanan data pada data set yang digunakan, dan dengan keteraturan operasi terhadap *dataset* tersebut, sehingga mempermudah proses yang seharusnya rumit menjadi mudah.

1.3.4 Penerapan Struktur Data

Dengan sifatnya ini, sebuah struktur data dapat diterapkan untuk:

- Pengolahan *database*
- Pengolah kata (*word processor*) yang kolomnya berubah secara dinamis.
- Pemampatan berkas atau citra
- Pemrosesan *Formula*
- Pengolahan *Graph*
- Membantu proses pencarian
- Membantu proses penyusunan dan pembacaan *formula*
- Membantu proses *backward chaining*

2. Konsep Tipe Data Abstrak

2.1. Tipe Data Abstrak

- Tipe Data Abstrak adalah suatu bentuk struktur data yang memiliki kegunaan atau perilaku yang serupa Model matematika termasuk operasinya
- TDA terdiri dari
  - *domain* (= *a set of values*)
  - *set of operations*

2.2 Tipe Data

- Tipe data :
  - Nilai yang mungkin terisi ke variabel
- Variabel agar dapat digunakan harus dideklarasikan sesuai dengan tipe data yang akan ditampungnya
- Suatu variabel tidak dapat menampung data yang tidak sesuai dengan tipe data peruntukannya
- Ada dua macam tipe data
  A. Tipe data sederhana/primitif
  B. Tipe data bentukan

2.3 Tipe Data Sederhana

Merupakan tipe data bawaan dari bahasa pemrograman.

Beberapa tipe data primitif (ada yang menyebutnya tipe data sederhana) yang umum terdapat pada berbagai bahasa pemrograman

- *Boolean* → Tipe data yang hanya memperbolehkan dua nilai 1/0 atau *TRUE/FALSE* saja
- *Character* → menampung 8 bit data yang diterjemahkan menjadi karakter, *Character* termasuk tipe data *integer*
- *Integer* → bilangan bulat, Terdapat beberapa jenis bilangan *integer* berdasarkan panjang bit nya: *Byte*, *short integer*, *integer*, *long integer*
- Pecahan → bilangan pecahan, umumnya direpresentasikan dalam bentuk *floating point*, berdasarkan panjang dan ketelitiannya, *floating point* dapat dibagi menjadi *single precision* (32 bit) dan *double precision* (64 bit)

2.4 Tiga Kategori Tipe Data Primitif

- *Integral* (bulat)

  Tipe data yang memperlakukan *integer* atau bilangan tanpa bagian, contoh *integer*, *char* dan boolean

- Pecahan

  Dinyatakan dalam bentuk *Floating – point*, contoh *single*, *double*, *real*

- *Enumeration* (enumerasi)

  user-defined data type. Contoh:

  enum bulan {JAN, PEB, MAR, APR, MEI, JUN, JUL, AGU, SEP, OKT, NOP, DES};

2.5 *Record / Structure*

- Rekaman atau *record* atau *structure* adalah sekumpulan data yang disusun dari tipe data yang sama atau tipe data yang berbeda.
- Sebuah record berisi beberapa variabel lain yang ‘dipaketkan’. Konsep struktur data seperti ini sedikit mirip dengan konsep class dan object dalam *object oriented programming*
- *Record*/*Structure* harus di definisikan terlebih dahulu,
- Hasil definisi *Record*/*Structure* diperlakukan seperti tipe data,
- Ketika akan digunakan, *Record*/*Structure* harus dideklarasikan dahulu pada sebuah variabel

2.5.1 Mengakses *Record*

- *Record* diakses pada *field-field*nya
- *Record* dapat diakses dengan menyebutkan terlebih dahulu nama *variable* diikuti nama *field* yang akan diakses setelah didahului tanda titik

2.5.2 *Record* dalam Pascal

- Definisi

```
Type
   RecordName = Record
     FieldName1 : vartype;
     FieldName2 : vartype;
     FieldName3 : vartype;
     …
     FieldNamen : vartype;
   End;
```

- Deklarasi

```
var
   varRecord = RecordName;
```

- Penugasan

```
varRecord.FieldNameN := data;
```

- Mengakses Record

```
varData:= varRecord.FieldNameN;
```

2.5.3 Contoh *Record* dalam Pascal

- Definisi

```
Type
   Bangun = Record
     x1 : integer;
     y1 : integer;
     x2 : integer;
     y2 : integer;
     x3 : integer;
     y3 : integer;
   End;
```

- Deklarasi

```
var
   Segitiga = Bangun;
```

- Penugasan

```
Segitiga.x1 := 10;
Segitiga.y1 := 16;
```

- Mengakses Record

```
temp:= Segitiga.x1;
```

2.5.4 *Record* dalam C++

- Definisi

```
struct RecordName
{
    vartype FieldName1;
    vartype FieldName2;
    vartype FieldName3;
    ...
    vartype FieldNameN;
};
```

- Deklarasi

```
RecordName varRecord;
```

- Penugasan

```
varRecord.FieldNameN = data;
```

- Mengakses Record

```
VarData = varRecord.FieldNameN;
```

2.5.5 *Record* dalam C++ (Lanj.)

- Definisi

```
struct Bangun
{
    int x1;
    int y1;
    int x2;
    int y2;
    int x3;
    int y3;
};
```

- Deklarasi

```
Bangun Segitiga;
```

- Penugasan

```
Segitiga.x1 = 10;
Segitiga.y1 = 16;
```

- Mengakses Record

```
Temp = Segitiga.x1;
```

2.5.6 Record pada Phyton

- Phyton tidak memiliki perintah untuk membuat structure record
- Untuk mensiasati hal ini, maka kita dapat mempergunakan Class untuk dibuat menjadi structure, kemudian dengan class tersebut ktia dapat menginstansiasi menjadi objek.

2.5.6 Record pada Phyton (lanjut)

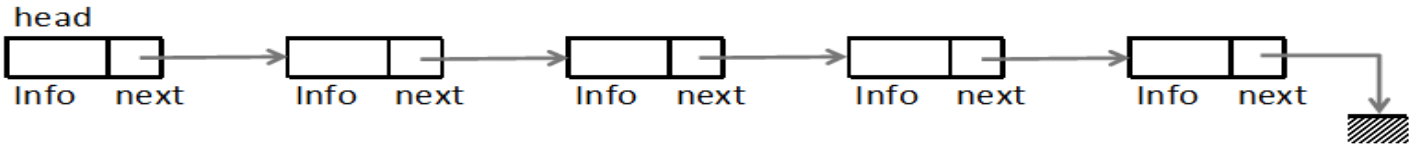
```
Contoh membuat structure Record
#Define Class
class beverage:
  def __init__(self, nama, harga):
    self.nama = nama
    self.harga = harga

 #membuat objek tunggal dari class beverage
p1 = beverage("Bajigur", 25000)  #objek
tunggal

#cara mengakses atribut/property objek dari
class beverage
print(p1.nama, p1.harga)
```

2.5.7 Array Record Phyton

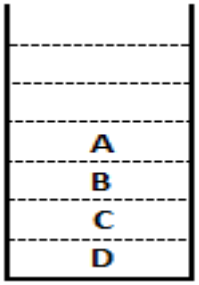
```
#membuat array objek dari class beverage
p = [] #list objek (array objek)
p.append(beverage("Bajigur", 25000))
p.append(beverage("Es Dogger", 15000))
p.append(beverage("Puding", 5000))
p.append(beverage("Cendol", 10000))

#cara mengakses atribut/property objek dari class beverage
print(p1.nama, p1.harga)
print(p[0].nama, p[0].harga)
print(p[1].nama, p[1].harga)
print(p[2].nama, p[2].harga)
print(p[3].nama, p[3].harga)
```

2.6 ADT *Basic Form*

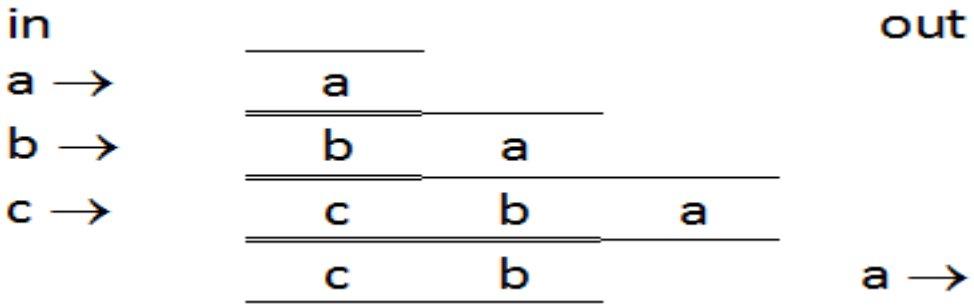
- *Linked list*



- *Stack*



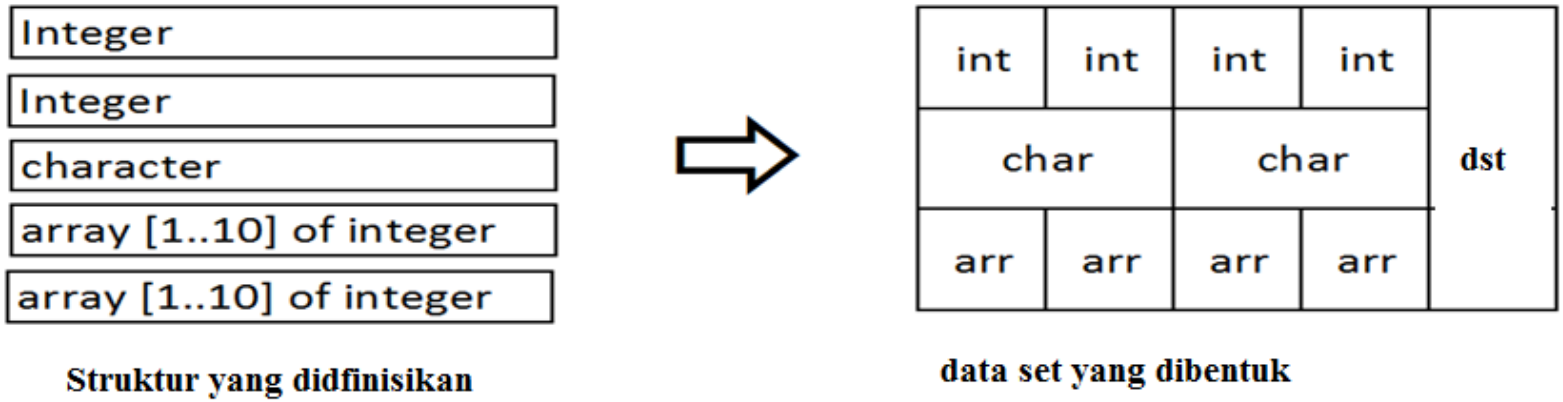
- *Queue*



2.7 Generalisasi

- Pembentukan semua Sel/elemen (jika pada Array atau Stack) ataupun simpul (jika pada linked-list, binary tree) pada data set dilakukan berdasarkan field-field struct / record yang didefinisikan
- semua sel/elemen/simpul/verteks tersebut memiliki struktur yang sama dengan struct yang digunakan untuk mendeklarasikannya

Contoh



Struktur yang didfinisikan **data set yang dibentuk**

2.7.1 Contoh Generalisasi dalam Bentuk *Record / Struct* menjadi Sel-sel Array (*phyton)*

```
#Define Class
class Mahasiswa:
  def __init__(self, nama, nim, nilai):
    self.nama = nama
    self.nim = nim
    self.nilai = nilai

#membuat array objek dari class beverage
p = [] #list objek (array objek)
p.append(mahasiswa("Nobita", 3210001, 0))
p.append(Mahasiswa("Shizuka", 3210001, 100))
```

Maka semua element pada sel-sel array Mahasiswa yang dibentuk dari record tersebut akan memiliki struktur yang samaco

2.7.2 Contoh Generalisasi dalam Bentuk *Record / Struct* menjadi Sel-sel Array untuk *Stack (C++)_*

```
Struct DataMhs {
   char nama[10];
   int nim[10];
   int nilai;
}
DataMhs Mahasiswa[100];
int UkuranStack;
```

- Maka semua element stacks pada sel-sel array Mahasiswa yang dibentuk dari record tersebut akan memiliki struktur yang sama

2.7.3 Contoh Generalisasi dalam Bentuk *Record / Struct* menjadi simpul linked–list untuk *Stack (phyton)*

```
class Simpul:
  def __init__(self,nama,
nim, nilai):
    self.nama = nama
    self.nama = nim
    self.nama = nilai
    self.next = None


class LinkedList:
  def __init__(self,
head=None):
    self.head = head


  def append(self,
simpulBaru):
    ptr = self.head
    if ptr:
      while ptr.next:
        ptr = ptr.next
      ptr.next = simpulBaru
    else:
      self.head = simpulBaru


l1 = LinkedList()
for i in range(0,10):
  temp = Simpul(a[i])
  l1.append(temp)
```

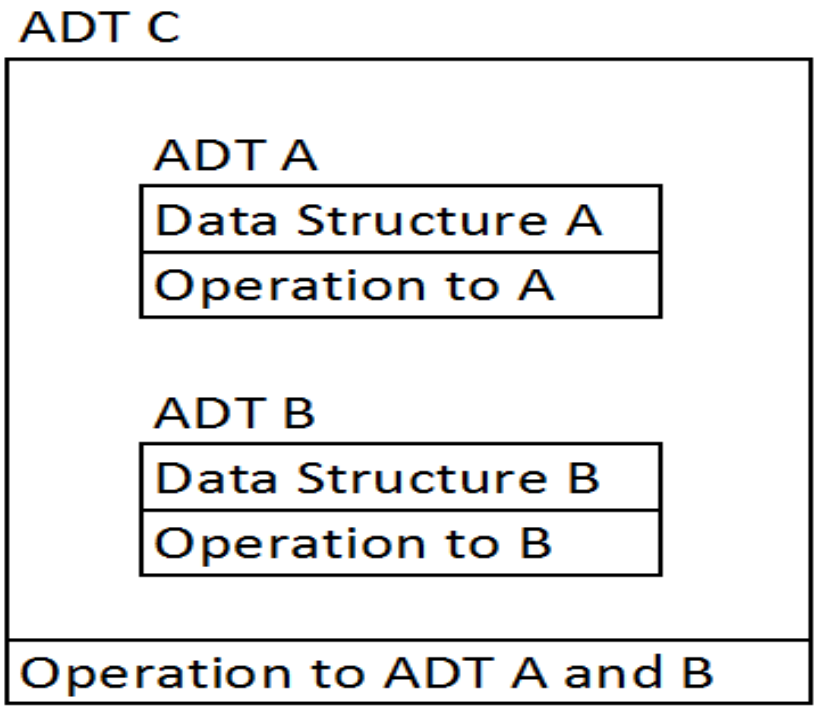
- Maka semua element pada simpul-simpul linked-list yang dibentuk dari record tersebut akan memiliki struktur yang sama

2.7.4 Contoh Generalisasi dalam Bentuk *Record / Struct* menjadi simpul linked–list

```
Struct DataMhs {
    char nama[10];
    char nim[10];
    int nilai;
    Struct DataMhs *Next; //untuk keperluan Linked
    List
}
Struct DataMhs *Ptr, *Head, *Temp;
...
Ptr = (Struct DataMhs *) malloc(sizeof(Struct DataMhs));
```

- Maka semua element stacks pada simpul-simpul linked-list yang dibentuk dari record tersebut akan memiliki struktur yang sama

2.8 Enkapsulasi



- Enkapsulasi terjadi pada ADT,
- Tipe data abstrak beserta operasinya dapat menjadi penyusun tipe data abstrak lainnya
- Setiap dataset akan tetap dapat dioperasikan sesuai dengan operasi ADT asalnya
- Contoh: sebuah *stack* dengan operasinya beserta *stack* lain dengan operasinya beserta operasi pemindahan isi *stack* yang satu ke *stack* yang lain

2.9 Pengertian *Data Set*

- *Data Set* (kumpulan data) adalah sejumlah data dengan susunan homogen
- *Data set* terdiri *record-record* sejenis yang tersusun secara sequensial
- Tiap *record*nya dapat memiliki *field-field* yang serupa ataupun berbeda
- *Field-field* dari tiap recordnya berisi nilai-nilai yang dapat diakses
- Kumpulan data juga bisa terdiri dari kumpulan dokumen atau *file*.

2.10 Bentuk *Data Set*

- Dapat diimplementasikan dalam bentuk
  - *Array* → disebut *Array Based*
  - *Linked List* → disebut *Linked-list Base*

Ringkasan

- Struktur data adalah cara penyimpanan, penyusunan dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien.
- Manfaat dari struktur data adalah Mengoptimalkan pengorganisasian data dalam *memory* sehingga dapat membantu dalam proses yang rumit
- Struktur Data disusun dalam dua basis yaitu *Array Base* dan *Linked List Base*
- Struktur data diterapkan dalam berbagai metode penyelesaian masalah

PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)