Tipe Data Abstrak

(TIB11 – Struktur Data)

Pertemuan 3, 4

Sub-CPMK

- Mahasiswa mampu menjelaskan konsep tipe data abstrak (C2, A2)

Materi:

1. *Record*
2. *Data set*
3. Konsep Tipe Data Abstrak

1. *Record*

1.1. 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

1.2. 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)

1.3. 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};

1.4. *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

1.5. 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

1.6. *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;
```

1.7. 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;
```

1.8. *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;
```

1.8. *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. *Data Set*

2.1. 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*.

2.2. Bentuk *Data Set*

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

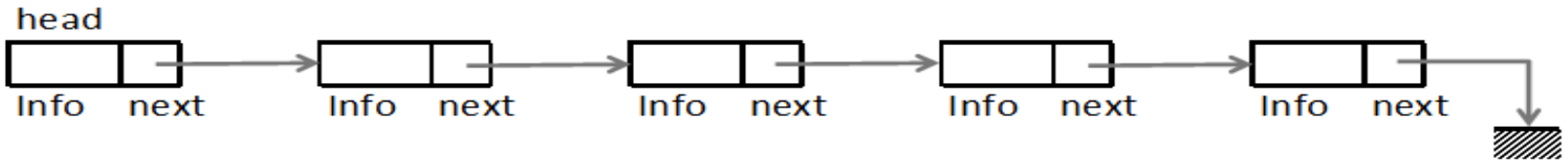
3. Konsep Tipe Data Abstrak

3.1. Tipe Data Abstrak

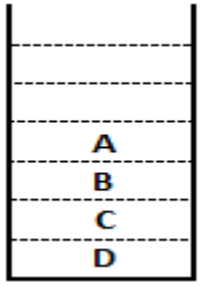
- Tipe Data Abstrak adalah suatu bentuk struktur data yang memiliki kegunaan atau perilaku yang serupa Model matematika termasuk operasinya
- TDA terdiri dari
  - *domain* (= *a set of values*)
  - *set of operations*

3.2. ADT *Basic Form*

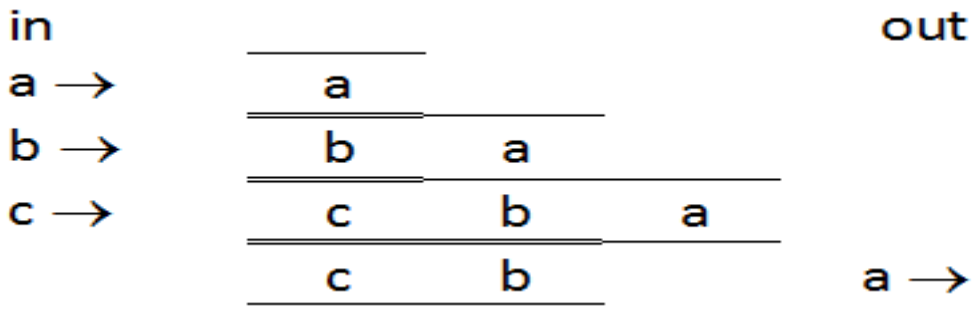
- *Linked list*



- *Stack*



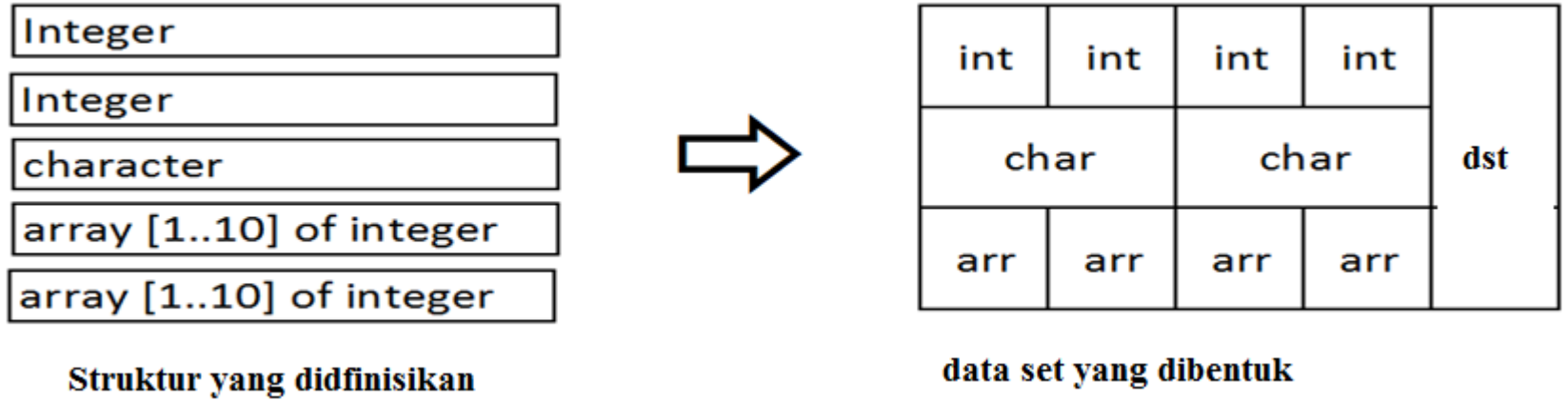
- *Queue*



3.3. 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

3.4. Contoh Generalisasi dalam Bentuk *Record* / *Struct* menjadi Sel-sel Array untuk *Stack*

```
Struct DataMhs {
   char NIM[10];
   int Nilai;
   int tugas[10];
}
DataMhs Mahasiswa[100];
int UkuranStack;
```

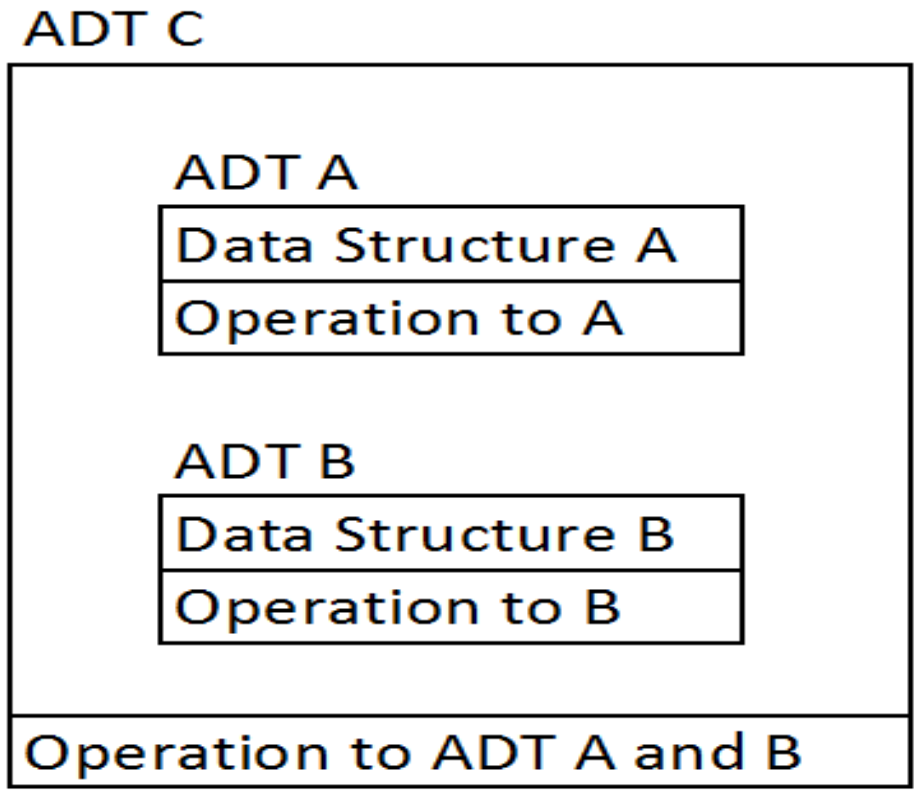
- Maka semua element stacks pada sel-sel array Mahasiswa yang dibentuk dari record tersebut akan memiliki struktur yang sama

3.4. Contoh Generalisasi dalam Bentuk *Record* / *Struct* menjadi simpul linked–list untuk *Stack*

```
Struct DataMhs {
   char NIM[10];
   int Nilai;
   int tugas[10];
   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

3.5. 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

Ringkasan

- Sejumlah bentuk struktur data yang memiliki kegunaan atau perilaku yang serupa, berupa model matematika yang memiliki *domain* dan set operasi
- Struktur Data sebuah *set variable* yang berisi beberapa tipe data yang berbeda serta memiliki relasi-relasi satu sama lain untuk setiap variabel
- Rekaman atau *record* atau *structure* adalah sekumpulan data yang disusun dari tipe data yang sama atau tipe data yang berbeda.
- *Record*/*Structure* harus di definisikan terlebih dahulu
- Hasil definisi *Record*/*Structure* diperlakukan seperti tipe data, sehingga ketika akan digunakan, *Record*/*Structure* harus dideklarasikan dahulu pada sebuah variabel

Terimakasih

TUHAN Memberkati Anda

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