MODUL PRAKTIKUM
STRUKTUR DATA

Oleh : Rahmat Robi Waliyansyah, M.Kom.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK DAN INFORMATIKA
UNIVERSITAS PGRI SEMARANG
2019

DAFTAR ISI

Modul 1
Array/Larik

A. Tujuan :

- Mahasiswa dapat mendefinisikan array
- Mahasiswa dapat memahami konsep array 1 dimensi, 2 dimensi dan banyak dimensi
- Mahasiswa dapat menggunakan operasi pada array
- Mahasiswa dapat menerapkan konsep array pada studi kasus tertentu

B. Dasar Teori

Array merupakan struktur data statik yang menyimpan sekumpulan elemen(data) dengan tipe data yang sama. Setiap elemen array dapat diakses langsung melaui indek array. Indek tersebut memiliki tipe data yang menyatakan keterurutan misalnya integer atau karakter.

Operasi yang dimiliki oleh array adalah :

- Operasi untuk menyimpan (Store) nilai
- Operasi untuk mengambil (Retrieve) nilai

Definisi array

1. Sebagai Peubah

Contoh :

```
Arr : array[1..25] of integer
NamaMhs : array['a'..'j'] of string
```

2. Sebagai tipe baru

Contoh :

```
type Arrint : array[1..25] of integer
P : Arrint
```

3. Mendefinisikan ukuran maksimum elemen larik sebagai konstanta

Contoh :

```
Const Nmaks = 10
type Larikint : array[1..Nmaks] of integer
P : LarikInt
```

Mengisi elemen array (Operasi Store)

```
data[1]='A';
data[2]='B';
data[3]='C';
```

Menampilkan data pada array (Operasi Retrieve)

```
printf("%c \n",data[1]);
printf("%c \n",data[2]);
printf("%c \n",data[3]);
```

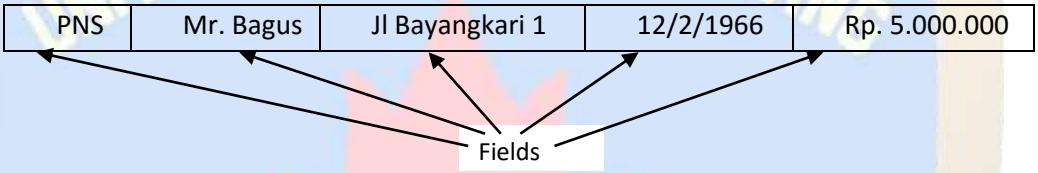
Modul 2
Record /Rekaman

A. Tujuan :

Mahasiswa dapat mendefinisikan record
Mahasiswa dapat mengakses elemen record
Mahasiswa dapat menerapkan record pada suatu studi kasus tertentu.

B. Dasar Teori

RECORDS digunakan untuk menyimpan element dengan tipe data yang berbeda-beda. Sebagai contoh terdapat record seperti berikut :



JIka kita perhatikan record diatas terdiri dari fields Status Kepegawaian, Nama, Alamat, Tanggal Lahir dan Gaji. Setiap fields tersebut mempunyai tipe data yang berbeda.Misalkan field Nama bertipe data char dan gaji bertipe data real.
Pada Bahasa C++ Record disebut Struct. Fields disebut member.

Deklarasi Struct

Deklarasi 1

```
struct nama_struct {
                tipe_data_1     nama_var_1;
                tipe_data_2     nama_var_2;
                tipe_data_3     nama_var_3;
                ……
    };
```

Deklarasi 2, Menggunakan kata tercadang *typedef*

```
typedef struct Mahasiswa {
      char NIM[8];
      char nama[50];
      float ipk;
};
```

Untuk menggunakan struct Mahasiswa dengan membuat variabel mhs1 dan mhs2

```
      Mahasiswa mh1s,mhs2;
```

Untuk menggunakan struct Mahasiswa dengan membuat variabel array mhs;

```
      Mahasiswa mhs[100];
```

Mengakses Komponen Record

```
Gunakan nama record
    nama member
    yang dipisah dot

Contoh :

Pegawai.nama;

Mhs[i].NIM;
```

Modul 3
Fungsi

A. Tujuan :

Mahasiswa dapat memahami funsi
Mahasiswa dapat menerapkan fungsi

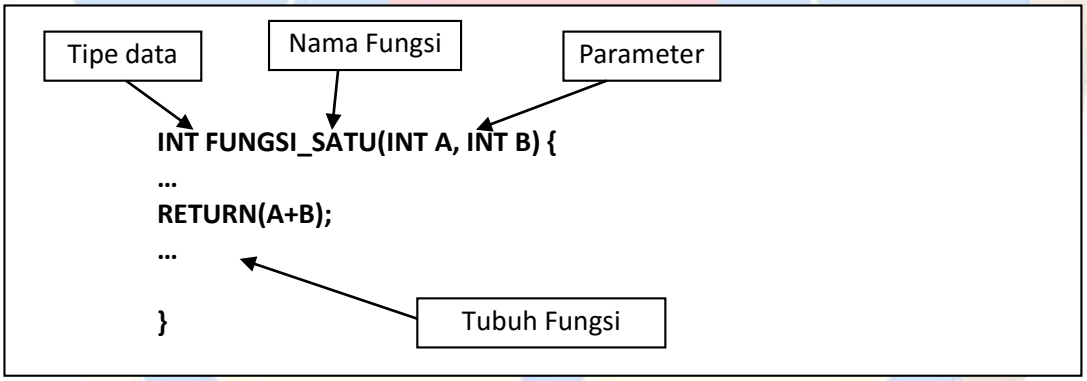
B. Dasar Teori

Procedure → tidak memberikan nilai balik
Function → memberikan nilai balik (return value)

Membuat Fungsi

Tentukan function yang akan dibuat :

- Return type
- Nama
- Tipe Parameter
- Tubuh fungsi (kode program)



Call-by-value vs Call-by-reference

- Call-by-value → tidak dapat merubah variabel yang memanggilnya secara langsung
  - Int penjumlahan (int a, int b)

- Call-by-reference → dapat merubah nilai variabel yang memanggilnya secara langsung. Nama altenative variabel/Shortcut harus diinisialisasi
  Deklarasi :
  - **int &bilangan;**
  - **double &total;**
  - **char &karakter;**

Modul 4
Rekursife

A. Tujuan :

Mahasiswa dapat memahami Rekursife
Mahasiswa dapat menerapkan Rekursife

B. Dasar Teori

Rekursife adalah function yang memanggil fungsi itu sendiri. Rekursife sangat memudahkan untuk memecahkan permasalahan yang kompleks

Sifat-sifat rekursif

- Dapat digunakan ketika inti dari masalah terjadi berulang kali
- Sedikit lebih efisien dari iterasi tapi lebih elegan
- Method-methodnya dimungkinkan untuk memanggil dirinya sendiri
- Data yang berada dalam method tersebut seperti argument disimpan sementara kedalam stack sampai method pemanggilnya diselesaikan

```
int factorial( int x ) {
if (x == 1)
return(1);
else
return(x * factorial(x-1));
}
```

Rekursive

Modul 5
Pengurutan/Sorting

A. Tujuan :

Mahasiswa dapat memahami Pengurutan
Mahasiswa dapat menerapkan pengurutan

B. Dasar Teori

Pengurutan data dapat dilakukan secara menaik (ascending) maupun menurun (descending). Ada beberapa keuntungan yang dapat kita peroleh pada data yang telah terurut, yaitu mudah dalam pencarian data, perbaikan kesalahan data, disisipi data baru atau menghapus data tertentu.

Terdapat beberapa metode pada pengurutan, diantaranya adalah :

- Bubble sort
- Selction sort
- Quick sort
- Merge Sort

Kerjakan dan jelaskan progam-program diatas

C. Tugas Pendahuluan

1. Jelaskan algortima metode pengurutan bubble sort, selction sort, quick sort dan merge Sort

D. Tugas Praktikum

1. Buatlah sebuah program yang dapat menerima input 10 buah data dengan menggunakan metode bubble sort, selction sort, quick sort dan merge Sort. Pengguna dapat memilih pada suatu menu metode pengurutan apa yang akan dipilih.

Modul 6
Pencarian/Searching

A. Tujuan :

Mahasiswa dapat memahami Pencarian
Mahasiswa dapat menerapkan Pencarian

B. Dasar Teori

Pencarian atau searching suatu data pada sekumpulan data merupakan proses yang sangat penting dalam kehidupan nyata. Seperti halnya pengurutan, pencarian juga dapat dilakukan dengan beberapa metode.

Terdapat beberapa metode pada pencarian, adalah :

- Pencarian beruntun (*sequential search*)
- Pencarian bagidua (*binary search*)

Modul 7
Pointer

A. Tujuan :

Mahasiswa dapat memahami Pointer
Mahasiswa dapat menerapkan Pointer

B. Dasar Teori

Pointer adalah variable yang berisi alamat memori sebagai nilainya dan berbeda dengan variabel biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variabel yang mempunyai nilai tertentu.

Adapun bentuk umum dari pernyataan variable pointer dalam C++ adalah :

```
Type *variable_name;
```

Dengan :

- type adalah tipe dasar pointer.
- variable_name adalah nama variable pointer.
- * adalah operator memori yang fungsinya untuk mengembalikan nilai variabel pada alamatnya yang ditentukan oleh operand.

Modul 8
Linked List

A. Tujuan :

Mahasiswa dapat memahami Linked List.
Mahasiswa dapat menerapkan Linked List

B. Dasar Teori

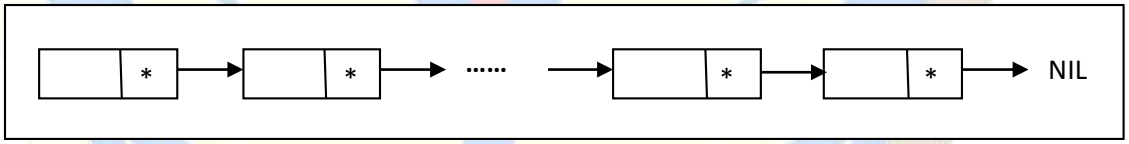
Linked list atau senarai adalah struktur data berisis kumpulan data / node yang tersusun secara sequential, saling sambung menyambung dan dinamis.
Linked list ini mirip array, namun linked list ini bersifat dinamis, penambahan data tidak terbatas, sequential acces, dan penghapusan data mudah.
Prinsip linked list dapat kita bandingkan seperti suatu rantai yang matanya dihubungkan satu sama lain. Mata rantai tersebut dapat kita asosiasikan dengan **record** atau **node**. Jadi, untuk selanjutnya dalam konteks linked list kita menggunakan terminology **NODE** untuk pengertian sebuah record.
Ciri khas suatu node dalam linked list adalah harus selalu terdapat field, paling sedikit dua bagian, yaitu :

1. Data
2. Pointer.

Secara umum linked list dibedakan atas 2 macam, yaitu :

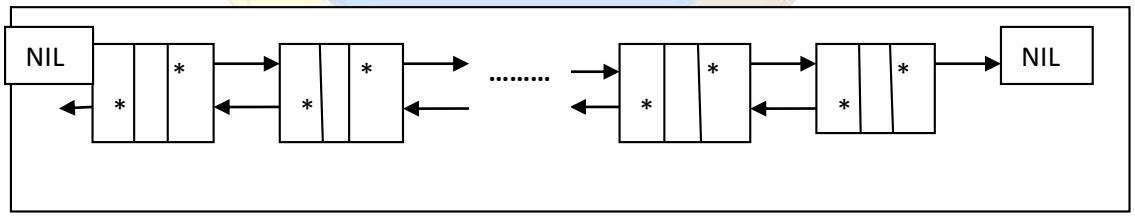
1. Single Linked List dan
2. Double Linked List

Single Linked List mempunyai satu pointer untuk setiap node yang menunjuk ke node berikutnya, artinya hanya punya satu arah.



Gambar 8.1 Node yang terakhir selalu menunjuk ke elemen kosong, dan diidentifikasi dengan nilai NIL

Pada Gambar 8.1 dapat kita lihat bahwa setiap record mempunyai satu pointer yang menunjuk ke record yang berikutnya, dengan pengecualian untuk record terakhir yang menunjuk ke record yang tidak ada. Record yang tidak ada tersebut kita definisi dengan nilai Nul (NIL) yang artinya juga sebagi akhir suatu list.
Double Linked List mempunyai dua pointer yang menunjuk ke node berikutnya dan sebelumnya, artinya punya dua arah.



Gambar 8.2 Previous pointer Node pertama dan Next pointer Node terakhir menunjuk ke elemen kosong, dan diidentifikasi dengan nilai NULL

Double Linked List dapat dilihat pada Gambar 8.2. Pointer Node pertama tidak mempunyai pendahulu, jadi pointer yang menunjuk ke elemen sebelumnya adalah elemen yang tidak ada (NIL) dan analog untuk node terakhir, dimana pointer yang menunjuk ke elemen berikutnya adalah elemen yang tidak ada (NIL).
Program dia atas masih terdapat kesalahan, coba betulkan sehingga dapat digunakan untuk menyisipkan, menghapus dan mencetak linked list.

Modul 9
STACK

A. Tujuan :

Mahasiswa terminologi yang terkait dengan struktur data stack.
Mahasiswa dapat memahami operasi-operasi yang ada dalam stack.
Mahasiswa dapat mengidentifikasi permasalahan-permasalahan pemrograman yang harus diselesaikan dengan menggunakan stack, sekaligus menyelesaikannya.

B. Dasar Teori

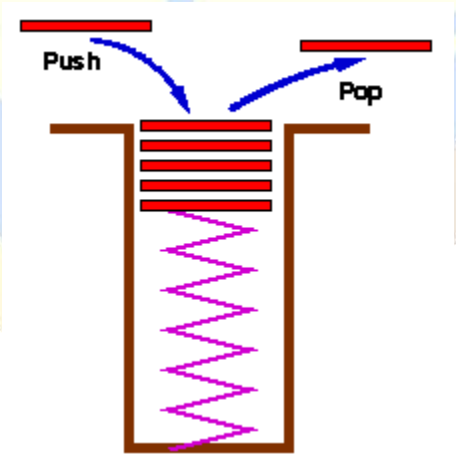
Deskripsi Stack

Salah satu konsep yang efektif untuk menyimpan dan mengambil data adalah "terakhir masuk sebagai yang pertama keluar" (Last In First Out / LIFO). Dengan konsep ini, pengambilan data akan berkebalikan urutannya dengan penyimpanan data.

Stack adalah sebuah kumpulan data dimana data yang diletakkan diatas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO. Dengan demikian, elemen terakhir yang disimpan dalam stack menjadi elemen pertama yang diambil. Dalam proses komputasi, untuk meletakkan sebuah elemen pada bagian atas dari stack, maka kita melakukan push. Dan untuk memindahkan dari tempat yang atas tersebut, kita melakukan pop.

Untuk menjelaskan pengertian diatas kita mengambil contoh sebagai berikut. Misalnya kita mempunyai dua buah kotak yang kita tumpuk, sehingga kotak kita letakkan diatas kotak yang lain. Jika kemudian stack dua buah kotak tersebut kita tambah dengan kotak ketiga dan seterusnya, maka akan kita peroleh sebuah stack kotak, yang terdiri dari N kotak.

Secara sederhana, sebuah stack bisa kita ilustrasikan seperti pada Gambar 4.1. Dari gambar tersebut, kita bisa mengatakan bahwa kotak B ada di atas kotak A dan ada di bawah kotak C. Dari gambar tersebut kita dapat melihat bahwa kita hanya bisa menambah atau mengambil kotak lewat satu ujung, yaitu ujung bagian atas. Nampak pula bahwa stack merupakan kumpulan data yang sifatnya dinamis, artinya kita bisa menambah atau mengambil data darinya.



Gambar 3.1 Ilustrasi Sebuah Stack

Penyajian Stack

Ada beberapa cara untuk menyajikan sebuah stack tergantung pada permasalahan yang akan kita selesaikan. Kita dapat menggunakan array untuk menyajikan sebuah stack, dengan anggapan bahwa banyaknya elemen maksimum dari stack tersebut tidak akan melebihi batas maksimum banyaknya elemen dalam array.

Pada saat ukuran stack, kalau kita teruskan menambahkan data lagi, akan terjadi overflow. Dengan demikian perlu data tambahan untuk mencatat posisi ujung stack. Dengan kebutuhan seperti itu, kita dapat menyajikan stack dengan menggunakan tipe data struktur (struct) yang terdiri dari dua field. Field pertama bertipe array untuk menyimpan elemen stack, medan kedua bertipe integer untuk mencatat posisi ujung stack.

Operasi Pada Stack

Operasi-operasi dasar pada stack adalah sebagai berikut :

1. Push : digunakan untuk menambah item pada stack pada tumpukan paling atas
2. Pop : digunakan untuk mengambil item pada stack pada tumpukan paling atas
3. Clear : digunakan untuk mengosongkan stack
4. IsEmpty : fungsi yang digunakan untuk mengecek apakah stack sudah kosong
5. IsFull : fungsi yang digunakan untuk mengecek apakah stack sudah penuh

Notasi POLISH

Salah satu pemanfaatan stack adalah untuk menulis ungkapan menggunakan notasi tertentu. Seperti kita ketahui, dalam penulisan ungkapan, khususnya ungkapan numeris, kita selalu menggunakan tanda kurung untuk mengelompokkan bagian mana yang akan dikerjakaan lebih dahulu. Sebagai contoh, dalam ungkapan :

(A + B) * (C – D)

Suku (A + B) akan dikerjakan lebih dahulu, kemudian suku (C – D), dan terakhir mengalikan hasil yang diperoleh dari dua suku tersebut. Cara penulisan ungkapan tersebut sering disebut dengan notasi infix, yang artinya adalah bahwa operator ditulis diantara dua operand.

Seorang ahli matematika mengembangkan satu cara penulisan ungkapan numeris yang selanjutnya disebut notasi polish atau notasi prefix, yang artinya adalah operator ditulis sebelum kedua operand yang akan disajikan. Sebagai contoh :

Infix	Prefix
A – B / (C * D ^ E)	-A/B*C^DE

Notasi lain, yang merupakan kebalikan notasi prefix, adalah notasi postfix atau notasi suffix, atau lebih dikenal dengan notasi Polish Terbalik (*Reverse Polish Notation* atau RPN). Dalam hal ini operator ditulis sesudah operand. Sebagai contoh :

Infix	Postfix
A – B / (C * D ^ E)	ABCDE^*/-

Modul 10
QUEUE/ANTRIAN

A. Tujuan :

Mahasiswa paham terminologi yang terkait dengan struktur data queue.
Mahasiswa dapat memahami operasi-operasi yang ada dalam queue

B. Dasar Teori

Deskripsi Queue

Konsep antrian kita kenal dengan istilah FIFO yaitu First in First out (elemen yang pertama kali masuk akan keluar pertama kalinya).

Struktur data in banyak digunakan di Teknik informatika untuk merepresentasikan
Antrian job pada system operasi
Antrian dalam dunia nyata.

Operasi Pada Queue

Adapun operasi pada antrian diantaranya adalah :

1. Create → membuat antrian baru
2. IsEmpty → Untuk memeriksa apakah Antrian sudah penuh atau belum
3. IsFull → mengecek apakah Antrian sudah penuh atau belum
4. Enqueue → menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang
5. Dequeue → untuk menghapus elemen terdepan/pertama dari Antrian
6. Clear → menghapus elemen-elemen Antrian
7. Tampil → Untuk menampilkan nilai-nilai elemen Antrian

Modul 11
TREE/POHON

A. Tujuan :

Mahasiswa dapat memahami terminologi yang terkait dengan struktur data tree.
Mahasiswa dapat memahami operasi-operasi yang ada dalam tree.

B. Dasar Teori

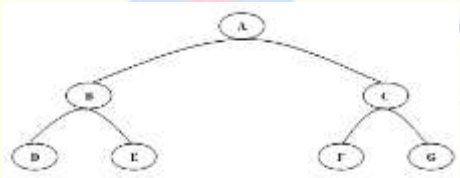
Deskripsi Tree

- Kumpulan node yang saling terhubung satu sama lain dalam suatu kesatuan yang membentuk layakya struktur sebuah pohon.
- Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (one-to-many) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node-node dari atas ke bawah.
- Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (one-to-many) dan tidak linier antara elemen-elemennya.

Contoh penggunaan struktur pohon :

- Silsilah keluarga
- Parse Tree (pada compiler)
- Struktur File
- Pertandingan

Struktur Pohon



Ancestor (F)	= C,A
Descendant (C)	= F,G
Parent (D)	= B
Child (A)	= B,C
Sibling (F)	= G
Size	= 7
Height	= 3
Root	= A
Leaf	= D,E,F,G
Degree (C)	= 2

Operasi-operasi Pada Tree

- **Insert**: menambah node ke dalam Tree secara rekursif. Jika data yang akan dimasukkan lebih besar daripada elemen root, maka akan diletakkan di node sebelah kanan, sebaliknya jika lebih kecil maka akan diletakkan di node sebelah kiri. Untuk data pertama akan menjadi elemen root.

- **Find**: mencari node di dalam Tree secara rekursif sampai node tersebut ditemukan dengan menggunakan variable bantuan ketemu. Syaratnya adalah tree tidak boleh kosong.
- **Traverse**: yaitu operasi kunjungan terhadap node-node dalam pohon dimana masing-masing node akan dikunjungi sekali.
- **Count**: menghitung jumlah node dalam Tree.
- **Height** : mengetahui kedalaman sebuah Tree.
- **Find Min dan Find Max** : mencari nilai terkecil dan terbesar pada Tree.
- **Child** : mengetahui anak dari sebuah node (jika punya).

Jenis Traverse

- **PreOrder**: cetak node yang dikunjungi, kunjungi left, kunjungi right.
- **InOrder**: kunjungi left, cetak node yang dikunjungi, kunjungi right.
- **PostOrder**: kunjungi left, kunjungi right, cetak node yang dikunjungi.