

# MODUL LABORATORIUM ALGOTIRMA DAN STRUKTUR DATA



**Penyusun Modul**

**Hindarto, S.Kom, MT**

**Ade Eviyanti, S.Kom**

**Yunianita Rahmawati, S.Kom**



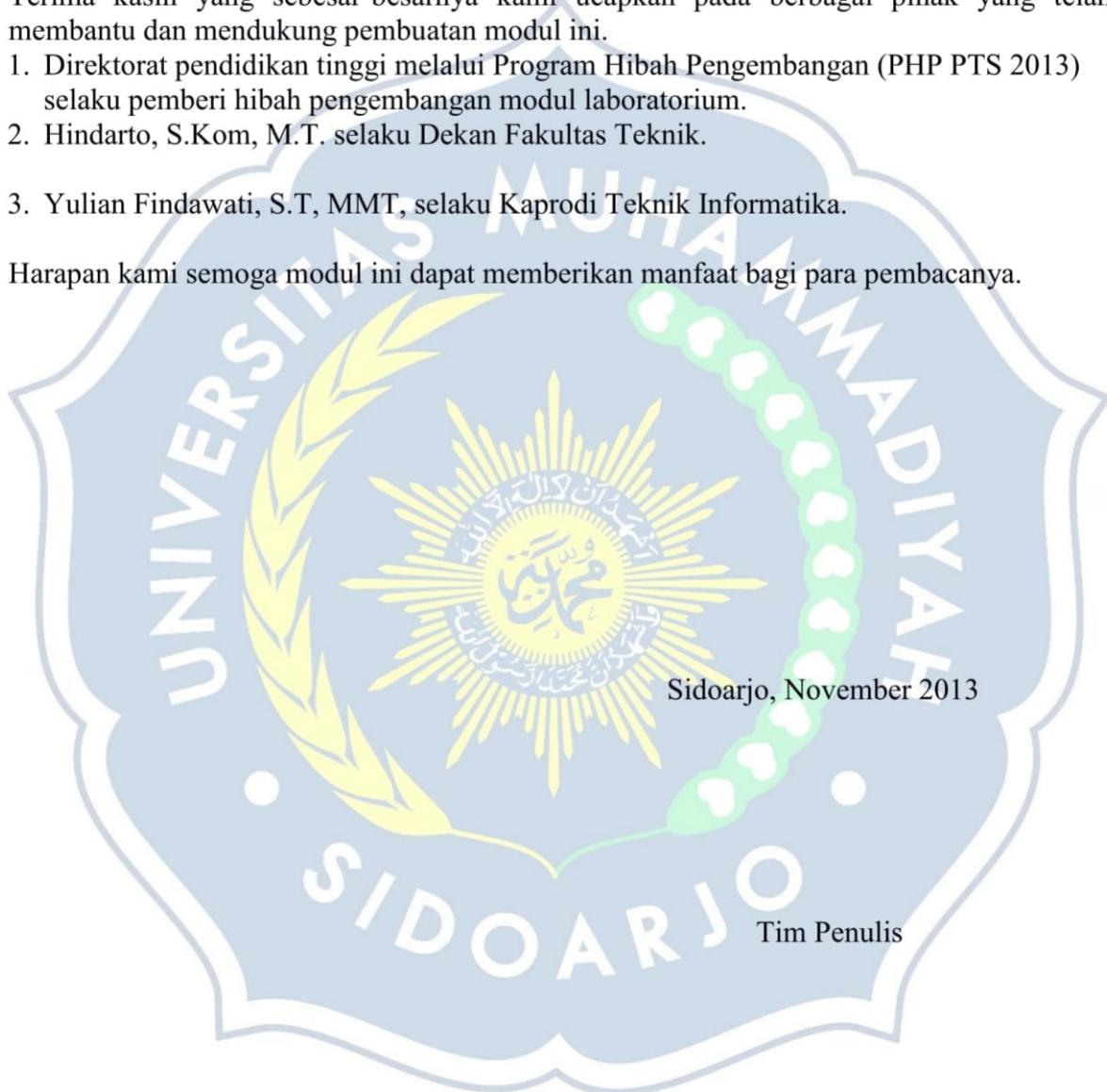
## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, sehingga modul Laboratorium algoritma dan struktur data ini dapat disusun dengan baik. Modul ini disusun sedemikian rupa agar dapat digunakan dengan mudah oleh mahasiswa teknik informatika sebagai panduan dalam penggunaan Laboratorium untuk kegiatan praktikum, untuk kegiatan penelitian, pengembangan dan inovasi dalam bidang Informatika dan Komputer.

Terima kasih yang sebesar-besarnya kami ucapkan pada berbagai pihak yang telah membantu dan mendukung pembuatan modul ini.

1. Direktorat pendidikan tinggi melalui Program Hibah Pengembangan (PHP PTS 2013) selaku pemberi hibah pengembangan modul laboratorium.
2. Hindarto, S.Kom, M.T. selaku Dekan Fakultas Teknik.
3. Yulian Findawati, S.T, MMT, selaku Kaprodi Teknik Informatika.

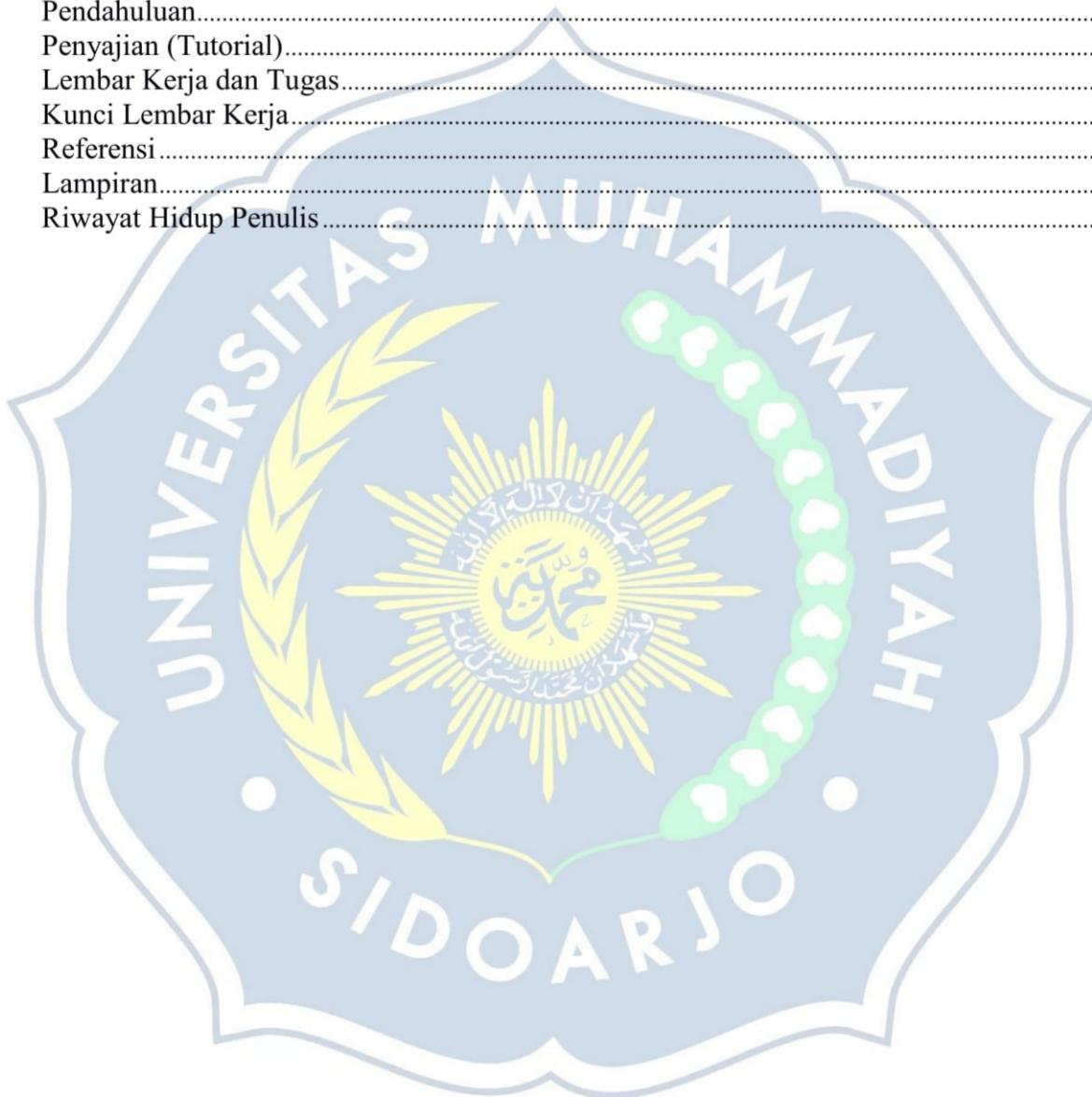
Harapan kami semoga modul ini dapat memberikan manfaat bagi para pembacanya.



## DAFTAR ISI

Cover .....	1
Kata Pengantar.....	2
Daftar Isi.....	3
Daftar Tabel.....	5
Daftar Gambar .....	6
Bab I Pendahuluan .....	7
A. Profil Laboratorium Algoritma dan Struktur Data .....	7
Visi Laboratorium .....	7
Misi Laboratorium.....	7
Sasaran Laboratorium.....	7
Tujuan Laboratorium.....	7
Manfaat Laboratorium .....	7
Foto Laboratorium.....	8
B. Manajemen Laboratorium Algoritma dan Struktur Data .....	9
Struktur Organisasi .....	9
SOP (Standard Operasional Prosedur) .....	10
Tata Tertib .....	13
Inventaris Laboratorium .....	13
C. Penggunaan Laboratorium .....	13
D. Peralatan Laboratorium .....	14
E. Peralatan Pendukung.....	14
Bab II Kurikulum.....	15
A. Analisis Materi / Instruksional .....	15
B. Silabus Praktik.....	15
C. Satuan Ajaran Praktikum (SAP) .....	15
Bab III Materi Modul .....	17
Pokok Bahasan 1 .....	17
Pendahuluan.....	17
Penyajian (Tutorial).....	17
Lembar Kerja dan Tugas.....	20
Kunci Lembar Kerja.....	23
Referensi.....	23
Pokok Bahasan 2 .....	24
Pendahuluan.....	24
Penyajian (Tutorial).....	24
Lembar Kerja dan Tugas.....	26
Kunci Lembar Kerja.....	31
Referensi.....	31
Pokok Bahasan 3 .....	32
Pendahuluan.....	32
Penyajian (Tutorial).....	32
Lembar Kerja dan Tugas.....	35
Kunci Lembar Kerja.....	39
Referensi.....	39
Pokok Bahasan 4 .....	40
Pendahuluan.....	40
Penyajian (Tutorial).....	40
Lembar Kerja dan Tugas.....	42

Kunci Lembar Kerja.....	47
Referensi.....	47
Pokok Bahasan 5 .....	48
Pendahuluan.....	48
Penyajian (Tutorial).....	48
Lembar Kerja dan Tugas.....	48
Kunci Lembar Kerja.....	51
Referensi.....	51
Pokok Bahasan 6 .....	52
Pendahuluan.....	52
Penyajian (Tutorial).....	52
Lembar Kerja dan Tugas.....	56
Kunci Lembar Kerja.....	60
Referensi.....	60
Lampiran.....	61
Riwayat Hidup Penulis.....	61



\

DAFTAR TABEL

Tabel 1 Daftar Range Nilai.....	11
Tabel 2 Satuan Acara Praktikum (SAP).....	16



**DAFTAR GAMBAR**

Gambar 1	Foto Laboratorium Algoritma Dan Struktur Data .....	8
Gambar 2	Struktur Organisasi.....	9
Gambar 3	Diagram alir Langkah-langkah yang dilakukan oleh Kepala Laboratorium, Mahasiswa, Asisten, Laboran dan Dosen Praktikum pada pelaksanaan praktikum 12	
Gambar 1.1	Struktur Array Satu Dimensi .....	18
Gambar 1.2	Output Soal 1.....	23
Gambar 1.3	Output Soal 2.....	23
Gambar 1.4	Output Soal 3.....	23
Gambar 2.1	List Tunggal.....	24
Gambar 2.2	List Tunggal dengan Kepala dan Ekor, List Tunggal Berputar.....	25
Gambar 2.3	List ganda dengan Kepala, List ganda dengan Kepala dan Ekor.....	25
Gambar 2.4	Output Soal 1.....	31
Gambar 2.5	Output Soal 2.....	31
Gambar 3.1	Ilustrasi Stack .....	32
Gambar 3.2	Representasi Stack Statis .....	34
Gambar 3.3	Representasi Stack Dinamis.....	34
Gambar 3.4	Output Soal 1.....	39
Gambar 3.5	Output Soal 2.....	39
Gambar 4.1	Ilustrasi Antrian dengan 8 Elemen.....	40
Gambar 4.2	Representasi Queue Statis .....	41
Gambar 4.3	Representasi Queue Dinamis.....	42
Gambar 4.4	Output Soal 1.....	47
Gambar 5.1	Output Soal 1.....	51
Gambar 5.2	Output Soal 2.....	51
Gambar 5.3	Output Soal 3.....	51
Gambar 5.4	Output Soal 4.....	51
Gambar 6.1	Langkah 1 Bubble Sort.....	53
Gambar 6.2	Langkah 2 Bubble Sort.....	53
Gambar 6.3	Langkah 3 Bubble Sort.....	53
Gambar 6.4	Langkah Selection Sort .....	54
Gambar 6.5	Contoh Merger Sort.....	55
Gambar 6.6	Output Soal 1.....	60
Gambar 6.7	Output Soal 2 .....	60

## BAB I PENDAHULUAN

### A. Profil Laboratorium Algoritma Dan Struktur Data

#### Visi Laboratorium

”Mewujudkan Laboratorium Algoritma dan Struktur Data yang bermutu Tingkat Nasional 2020 di Bidang Informatika dan Komputasi, serta Menjadi Unit Pendukung Kegiatan di Unit Kerja Lain di Lingkungan UMSIDA”.

#### Misi Laboratorium

1. Menyelenggarakan Laboratorium Algoritma dan Struktur Data berkualitas, dan pembimbingan berdasarkan kurikulum yang terintegratif dan berkelanjutan untuk seluruh Mahasiswa Informatika.
2. Menyediakan sarana dan prasarana untuk kegiatan penelitian, pengembangan dan inovasi dalam bidang Informatika dan Komputer.
3. Memberikan pelayanan laboratorium yang baik dan bermutu.
4. Meningkatkan Kerjasama dengan instansi pemerintah/swasta/masyarakat di bidang Informatika dan Komputer di Tingkat Nasional.

#### Sasaran Laboratorium

Tersedianya sarana dan prasarana untuk penunjang kegiatan Laboratorium. Tersedianya assisten laboratorium dan assisten praktikum yang professional dan sesuai dengan tugasnya. Adanya koordinasi atau kerja sama yang baik antara unit-unit yang terkait. Meningkatnya pengelolaan data akademik. Meningkatnya pelayanan administrasi Laboratorium.

#### Tujuan Laboratorium

1. Mengembangkan sumber daya laboratorium terpadu untuk peningkatan kualitas pelayanan praktikum, penelitian, pembelajaran dan pengembangan masyarakat.
2. Mengembangkan sumber-sumber pendanaan laoratorium yang berkelanjutan.
3. Pengembangan sistem manajemen Laboratorium yang sehat dan harmonis.
4. Mengembangkan kelembagaan laboratorium yang kuat dan dinamis serta meningkatkan kapabilitas dan kompetensi SDM.

#### Manfaat Laboratorium Algoritma Dan Struktur Data

1. Diharapkan dapat memfasilitasi pengembangan mata pelajaran TIK (Teknologi Informasi Komputer) sebagai bagian dasar pemanfaatan teknologi untuk mempersiapkan peserta didik yang memadai agar dimasa depan dapat berperan sebagai kontribusi dari penguasaan komputer.

2. Untuk menunjang proses pembelajaran yang bermutu, teratur dan berkelanjutan.
3. Meningkatkan pengalaman dan keterampilan dalam mengimplementasikan penguasaan komputer pada mata pelajaran lainnya.
4. Memberikan dampak kepada siswa untuk lebih terampil mengkomunikasikan teori dengan praktik dalam proses belajar-mengajar.
5. Memberikan pengalaman langsung kepada siswa melalui praktik-praktik lapangan.

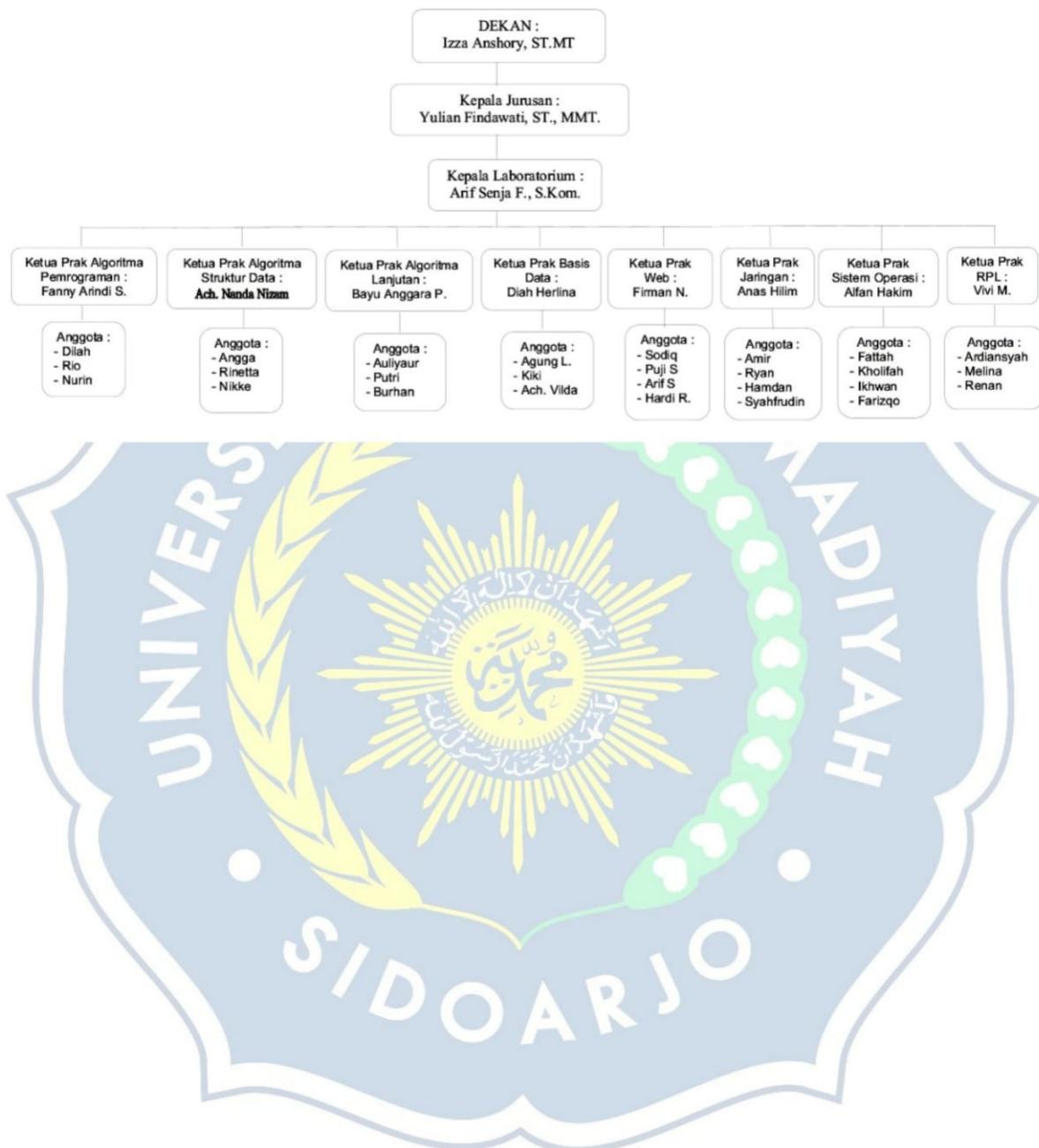
Foto Laboratorium



Gambar 1 Foto Laboratorium Algoritma Dan Struktur Data

## B. Manajemen Laboratorium

### Struktur Organisasi Laboratorium



## SOP (Standard Operasional Prosedur)

**SOP (Standard Operasional Prosedur)** adalah suatu pedoman tertulis yang dipergunakan untuk memperlancar kegiatan Laboratorium teknik Informatika.

### a. Tujuan SOP

1. Meningkatkan efisiensi pelaksanaan kegiatan laboratorium Algoritma dan struktur Data pada jurusan Teknik Informatika.
2. Memberikan sanksi bagi pengguna laboratorium yang tidak memenuhi aturan.

### b. Pihak Terkait

1. Mahasiswa;
2. Dosen;
3. Laboran;
4. Asisten; dan
5. Kalab.

### c. Waktu Dan Tempat Pelaksanaan

Waktu praktikum sesuai dengan jadwal yang telah ditentukan. Tempat pelaksanaan praktikum di Laboratorium Algoritma dan Struktur Data.

#### c.1 Jumlah tatap muka

Jumlah tatap muka praktikum Algoritma dan Struktur Data di Laboratorium Algoritma dan Struktur Data menetapkan 6 kali tatap muka.

#### c.2. Lama praktikum setiap tatap muka

Lama praktikum untuk setiap tatap muka adalah 2 jam dengan pertimbangan bahwa setengah jam pertama untuk persiapan peralatan sedangkan satu setengah jam berikutnya untuk materi praktikum . Jeda waktu antar praktikum 15 menit dengan pertimbangan bahwa diperlukan waktu kurang lebih 15 menit bagi asisten untuk mempersiapkan pelaksanaan praktikum berikutnya.

## Prosedur Penggunaan Laboratorium Algoritma Dan Struktur Data

1. Tidak menginstal software pada komputer yang digunakan
  - a. Tidak menginstal dan menghapus
  - b. Tidak membuat akun, didirektori
  - c. Tidak menambah atau mengurangi data yang ada, kecuali yang dibutuhkan
2. Pelanggaran atas aturan ini dikenakan sanksi tidak dapat mengikuti praktikum berikutnya.
3. Asisten harus melaporkan terjadinya pelanggaran ke laboran dan mencatat pelanggaran yang terjadi

4. Kerusakan karena kelalaian praktikan menjadi tanggung jawab praktikan yang bersangkutan.
5. Tidak membawa makanan dan minuman ke dalam laboratorium

#### Prosedur Pelaksanaan Praktikum

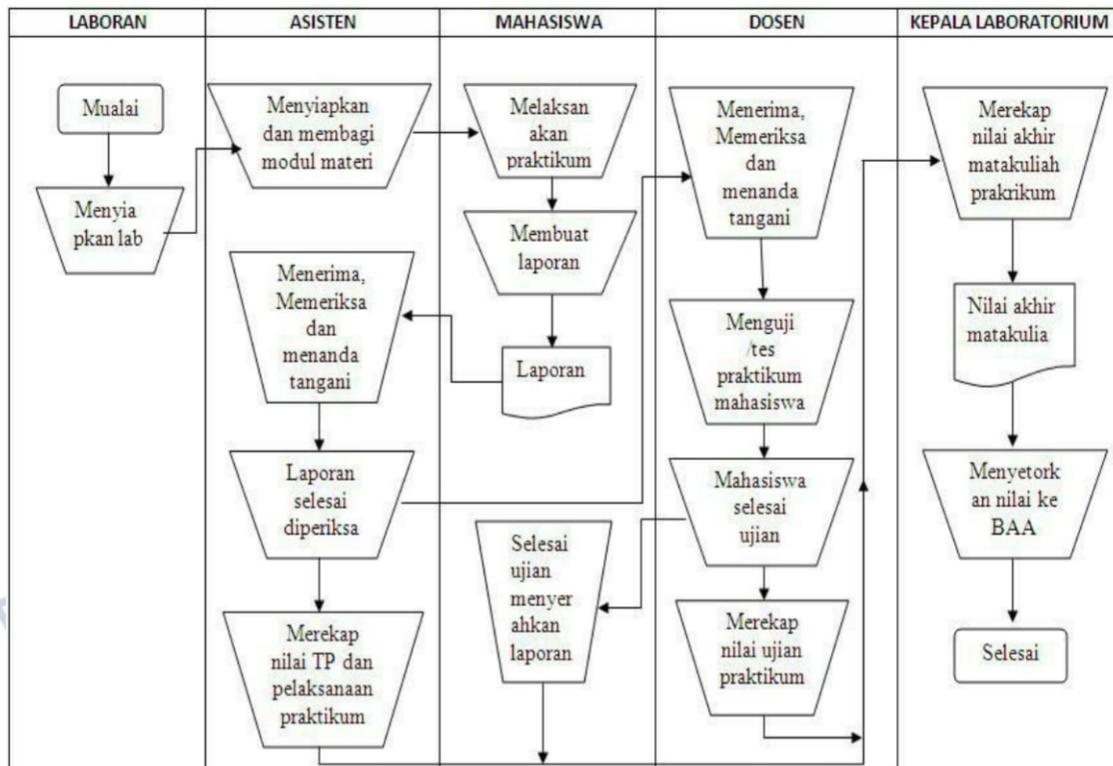
1. Laboran menyiapkan laboratorium dan perlengkapannya.
2. Asisten menyiapkan materi praktikum berdasarkan modul praktikum.
3. Mahasiswa melaksanakan praktikum didampingi asisten dan dosen pembimbing
4. Mahasiswa membuat laporan dan diserahkan kepada asisten pada pertemuan berikutnya.
5. Asisten memeriksa dan menandatangai asistensi laporan praktikum mahasiswa
6. Laporan yang telah diperiksa diserahkan kembali kepada mahasiswa
7. Pada akhir praktikum, Dosen memberikan tes/ujian yang harus diikuti oleh semua mahasiswa.
8. Dosen menyerahkan nilai hasil tes/ujian praktikum yang dilaksanakan mahasiswa ke Kepala Laboratorium.
9. Asisten merekap nilai praktikum (20% tugas + 40 % pelaksanaan praktikum )
10. Asisten menyerahkan nilai ke Kepala laboratorium.
11. Kepala laboratorium merekap nilai praktikum (20% tugas + 40 % pelaksanaan praktikum + 30 % ujian praktikum + 10 % laporan praktikum)
12. Kepala laboratorium menyerahkan nilai praktikum ke BAA
13. Kalab membuat kartu peserta (lampiran) dengan ketentuan nilai : Dinyatakan lulus praktikum jika nilai minimal praktikum C.

Tabel 1 Daftar Range Nilai

Nilai Huruf	Range Nilai
A	>85,1
AB	76 – 85
B	66 – 75
BC	56 – 65
C	46 – 55
D	36– 45
E	< 35

### Flow Chart Pelaksanaan Praktikum

**Langkah-langkah yang dilakukan oleh Laboran, Asisten, Mahasiswa, Dosen dan Kalab disajikan dalam diagram alir pada Gambar 3.**



**Gambar 3 Diagram alir Langkah-langkah yang dilakukan oleh Kepala Laboratorium, Mahasiswa, Asisten, Laboran dan Dosen Praktikum pada pelaksanaan praktikum**

## Tata Tertib Praktikum Laboratorium Fakultas Teknik Umsida

- 1. Praktikum dilaksanakan tepat waktu sesuai dengan jadwal yang ditetapkan.**
- 2. Mahasiswa yang terlambat datang atau absen harus memberikan surat/bukti yang dapat dipercaya (surat dokter atau surat keterangan kerja dari perusahaan).**
- 3. Mahasiswa diperkenankan pindah kelompok/jam/hari praktikum dengan syarat mengkonfirmasi 1 minggu sebelum pelaksanakan praktikum melalui Koordinator Praktikum dan Kepala Lab.**
- 4. Mahasiswa yang tidak hadir pada saat jadwal yang telah ditentukan diperkenankan mengikuti praktikum dengan membayar denda Rp 25.000,- per modul praktikum selama proses praktikum masih berlangsung.**
- 5. Mahasiswa harus berbusana yang sopan dan rapi ( tidak diperkenankan memakai kaos oblong dan Sandal atau sepatu sandal ).**
- 6. Praktikum dianggap selesai jika mahasiswa telah menyerahkan laporan sementara dan alat yang dipinjam dalam keadaan baik, bersih, dan rapi.**
- 7. Kerusakan alat yang dipinjam oleh mahasiswa menjadi tanggung jawab penuh kelompok mahasiswa yang bersangkutan.**
- 8. Selama praktikum berlangsung, mahasiswa dilarang merokok, makan, bergurau, bermain alat, menghidupkan *hand phone*, atau pun keluar masuk ruangan tanpa seijin dosen pembimbing / asisten pendamping.**
- 9. Setelah melakukan praktikum, mahasiswa harus membuat laporan sementara hasil pengamatan praktikum rangkap dua dan menyerahkan kepada dosen pembimbing / asisten pada saat meninggalkan ruangan untuk ditanda tangani (yang nantinya dilampirkan dalam laporan akhir).**
- 10. Mahasiswa yang tidak melaksanakan praktikum 1 Modul dinyatakan tidak lulus.**
- 11. Laporan Akhir Praktikum, *cover*-nya menggunakan Standar Fakultas dan Laporan diserahkan 2 minggu setelah jadwal masing-masing kelompok.**
- 12. Apabila Laporan diserahkan lebih dari 2 minggu maka dinyatakan TIDAK LULUS dan laporan Praktikum diserahkan ke koordinator praktikum dan kepala lab.**
- 13. Mahasiswa yang dinyatakan tidak lulus Praktikum harus mengulang dijadwal praktikum berikutnya dengan membayar biaya praktikum yang telah ditentukan oleh Universitas melalui bank yang ditunjuk oleh UMSIDA.**

## Inventaris Laboratorium Algoritma dan Struktur Data

Monitor	:	12
CPU	:	12
Mouse	:	11
Keyboard	:	11
Meja	:	12
Kursi	:	25

## C. Penggunaan Laboratorium Algoritma dan Struktur Data

**Laboratorium untuk praktikum Algoritma dan Struktur Data.**

## D. Peralatan Laboratorium

Di laboratorium Algoritma dan Struktur Data Fakultas Teknik Universitas Muhammadiyah Sidoarjo memiliki peralatan Laboratorium yang ada untuk menunjang pelaksanaan praktikum yang berupa :

1. Modul Praktikum Algoritma dan Struktur Data
2. Komputer
3. LCD
4. Papan Tulis

#### E. Peralatan Pendukung

Software yang digunakan dalam praktikum Algoritma dan Struktur Data adalah Visual C++ 2010.



## BAB II

### KURIKULUM

#### A. Analisis Materi / Instruksional

**Mahasiswa diharapkan dapat :**

Memecahkan masalah menjadi sebuah algoritma (langkah-langkah) yang akan dijalankan oleh komputer, kemudian mengimplementasikannya menjadi sebuah program komputer Memecahkan masalah pemrograman yang harus diselesaikan dengan materi yang ada pada pemrograman lanjut seperti pointer, struct, operasi file dsb.

Merepresentasikan data yang digunakan dalam pemrograman (baik data input atau data output) dengan struktur data yang tepat.

Mengetahui & membandingkan macam-macam algoritma dalam proses pengurutan dan pencarian dan dapat menentukan algoritma yang digunakan dalam permasalahan pemrograman yang diselesaikannya.

#### B. Silabus Praktikum

**Silabus praktikum algoritma dan struktur data :**

1. Array, Pointer, dan Struktur.
2. Linked List.
3. Stack.
4. Queue.
5. Rekursif.
6. Sorting.

#### C. Satuan Acara Praktikum (SAP)

Pertemuan Ke-	Tujuan Instruksional Khusus (TIK)	Topik	Sub Topik
1.	<b>Menjelaskan dan mendeklarasikan konsep dasar struktur data, array, pointer, dan struktur.</b>	Array, Pointer dan Struktur	Menjelaskan dan mendeklarasikan konsep dasar struktur data, array, pointer, dan struktur.  Latihan
2.	<b>Memahami Linked List. Memahami jenis-Jenis Linked List. Memahami operasi insert simpul pada Linked</b>	Linked List	Deklarasi Linked List. Jenis-Jenis Linked List. Operasi insert simpul pada  Linked. Latihan
3.	<b>Memahami Konsep</b>	Stack	<b>Konsep dasar</b>

<p><b>dasar</b> <b>tumpukan (LIFO)</b> Memahami operasi-operasi pada tumpukan (PUSH dan POP)</p>	<p>(Tumpukan)</p>	<p><b>tumpukan</b> <b>(LIFO)</b> Menjelaskan operasi-operasi pada tumpukan (PUSH dan POP)</p>
--	-------------------	---



### BAB III

#### MATERI MODUL

##### POKOK BAHASAN 1

##### **STRUKTUR DATA, ARRAY, POINTER, DAN STRUKTUR**

#### **PENDAHULUAN**

Pada pokok bahasan ini berisi penjelasan disertai contoh mengenai konsep struktur data, array, pointer, dan struktur yang menjadi pemahaman dasar bagi mahasiswa sebelum mempelajari struktur data, dimana konsep array, pointer, dan struktur digunakan untuk merepresentasikan sebuah struktur data, diharapkan mahasiswa dapat :

- a. Mengetahui konsep dasar struktur data.
- b. Memahami konsep array, pointer, dan struktur.

#### **PENYAJIAN (TUTORIAL)**

##### **A. Konsep Dasar Struktur Data**

Struktur data adalah sebuah bagian dari ilmu pemrograman dasar yang mempunyai karakteristik yang terkait dengan sifat dan cara penyimpanan sekaligus penggunaan atau pengaksesan data.

Struktur data bertujuan agar cara merepresentasikan data dalam membuat program dapat dilakukan secara efisien dalam pengolahan di memori dan pengolahan penyimpanan dari program ke storage juga lebih mudah dilakukan.

##### **B. Konsep Dasar Array**

Array adalah kumpulan elemen-elemen data. Kumpulan elemen tersebut mempunyai susunan tertentu yang teratur. Jumlah elemen terbatas, dan semua elemen mempunyai tipe data yang sama. Jenis-jenis array :

###### **Array Satu Dimensi**

Struktur array satu dimensi dapat dideklarasikan dengan bentuk umum berupa : **tipe\_var nama\_var [ukuran];**

Dengan :

- **Tipe\_var** : untuk menyatakan jenis elemen array (misalnya int, char, unsigned).
- **Nama\_var** : untuk menyatakan nama variabel yang dipakai.
- **Ukuran** : untuk menyatakan jumlah maksimal elemen array.

**Contoh :** float nilai\_ujian [5];

###### **Array Dua Dimensi**

Tipe data array dua dimensi biasa digunakan untuk menyimpan, mengolah maupun menampilkan suatu data dalam bentuk tabel atau matriks. Untuk mendeklarasikan array agar dapat menyimpan data adalah :

**tipe\_var nama\_var [ukuran1][ukuran2];**

**Dimana :**

- **Ukuran1** menunjukkan jumlah/nomor baris.
- **Ukuran2** menunjukkan jumlah/nomor kolom.

Jumlah elemen yang dimiliki array dua dimensi dapat ditentukan dari hasil perkalian :

**ukuran1 x ukuran 2.**

Seperti halnya pada array satu dimensi, data array dua dimensi akan ditempatkan pada memori secara berurutan.

**Array Multidimensi / Dimensi Banyak**

Array berdimensi banyak atau multidimensi terdiri dari array yang tidak terbatas hanya dua dimensi saja. Bentuk umum pendeklarasian array multidimensi adalah : **tipe\_var nama\_var [ukuran1][ukuran2]...[ukuran n];**

**Contoh :** int data\_angka [3][6][6];

Yang merupakan array tiga dimensi

#### **Mengakses Elemen Array :**

Dalam bahasa C++, data array akan disimpan dalam memori pada alokasi yang berurutan.

Elemen pertama biasanya mempunyai indeks bernilai 0. Contoh :

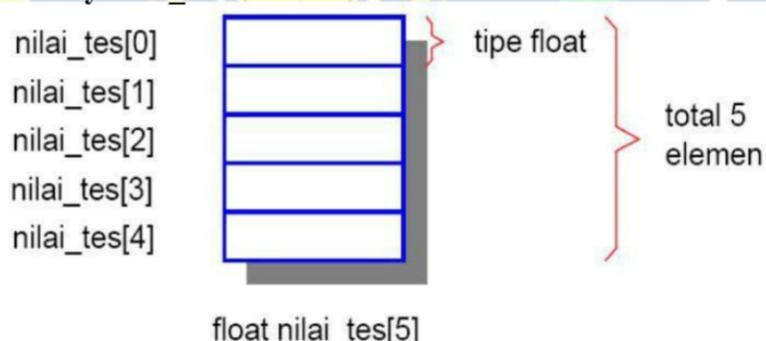
**Float nilai\_tes[5];**

Jika pada contoh di atas, variabel nilai\_tes mempunyai 5 elemen, maka elemen pertama mempunyai indeks sama dengan 0, elemen kedua mempunyai indeks 1, dan seterusnya. Bentuk umum pengaksesan suatu elemen variable array adalah :

**Nama\_var[indeks];**

Gambar berikut memperlihatkan urutan komponen array dalam memori.

Untuk variable array nilai\_tes :



**Gambar 1.1 Struktur Array Satu Dimensi**

#### **Inisialisasi Array :**

Array dapat diinisialisasikan secara langsung saat pertama kali dideklarasikan (efisien)

untuk array berdimensi sedikit).

**Contoh :** int x[2]={1,2};

Array dapat dideklarasikan terlebih dahulu, baru kemudian diisi elemennya.

**Contoh :**

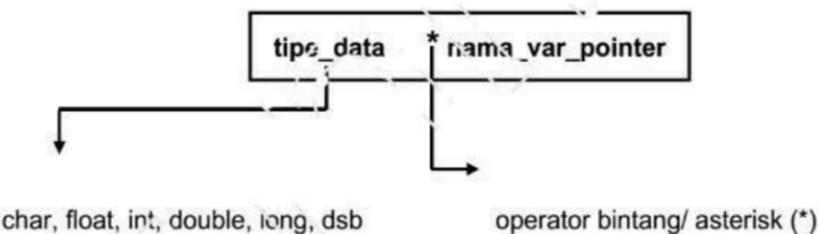
**int x[2];**

**x[0]=1;**

**x[1]=2;**

### **C. Konsep Dasar Pointer**

Pointer adalah sebuah variabel yang berisi alamat variable yang lain. Suatu pointer dimaksudkan untuk menunjuk ke suatu alamat memori sehingga alamat dari suatu variabel dapat diketahui dengan mudah. Deklarasi pointer :



**Operator pointer :**

Operator '&' : untuk mendapatkan alamat memori operand / variabel pointer.

Operator '\*' : untuk mengakses nilai data operand / variabel pointer.

#### D. Konsep Dasar Struktur

Struktur adalah koleksi dari variabel yang dinyatakan dengan sebuah nama, dengan sifat setiap variabel dapat memiliki tipe yang berlainan. Struktur biasa dipakai untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah satu kesatuan. Contoh sebuah struktur adalah informasi data tanggal, yang berisi tanggal, bulan, dan tahun.

**Mendeklarasikan Struktur :**

Contoh pendefinisian tipe data struktur adalah : struct  
data\_tanggal  
{int tanggal;

Masing-masing tipe dari elemen struktur dapat berlainan. Adapun variabel\_structur1 sampai dengan variabel\_structur M menyatakan bahwa variabel struktur yang dideklarasikan bisa lebih dari satu jika ada lebih dari satu variabel, antara variabel struktur dipisahkan dengan tanda koma.

**Mengakses Elemen Struktur :**

Elemen dari struktur dapat diakses dengan menggunakan bentuk :

variabel\_structur.nama\_field

Antara variabel\_structur dan nama\_field dipisahkan dengan operator titik (disebut operator anggota struktur). Contoh berikut merupakan instruksi untuk mengisikan data pada field tanggal :

```
tgl_lahir.tang
gal=30 int
bulan;
int tahun;
};
```

Yang mendefinisikan tipe struktur bernama data\_tanggal, yang terdiri dari tiga buah elemen berupa tanggal, bulan, dan tahun. Bentuk umum dalam mendefinisikan dan mendeklarasikan struktur adalah :

Struct nama\_tipe\_struktur

{

```

Tipe
field1
; Tipe
field2
; Tipe
field3
;
}variabel_struktur1.... variabel_strukturM;

```

## LEMBAR KERJA DAN TUGAS

### 1. Program pangkat dengan array dimensi satu.

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std ;

int main()
{
    int square[100]; // --> Array 1 dimensi dengan tempat yang dipesan sebanyak 100
    int i;
    int k;

    //Perhitungan
    for (i = 0; i < 10; i++) // angka yang ditampilkan 1-10
    {
        k = i + 1;
        square[i] = k * k;
        printf("\n Pangkat dari %d adalah %d", k, square[i]);
    }
    getch();
}

```

### 2. Program array dimensi dua.

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std ;

void printArray(int [][][3]); //-->Penulisan array 2 dimensi

int main()
{
    //Pengisian elemen array
    int
matrik1[2][3]={{1,2,3},{4,5,6}},matrik2[2][3]={1,2,3,4,5},matrik3[2][3]
={{1,
2},{4}};
    printArray(matrik1);
    printArray(matrik2);
    printArray(matrik3);
    getch();
}

//Prosedur tampilan array

```

```

void printArray(int a[][3])
{
    int i, j;
    for (i = 0; i<=1; i++)
    {
        for (j = 0; j<=2; j++)
            printf("%d ", a[i][j]);
        printf("\n");
    }
}

```

### 3. Program pointer.

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std ;
//cetak p dan *p
void main(void)
{
    int v=7, *p;//untuk mengakses nilai
    data p = &v; //untuk mendapatkan
    alamat memori
    cout<<"Nilai v = " <<v;
    cout<<endl;
    cout<<endl;
    cout<<"Nilai *p = " <<*p;
    cout<<endl;
    cout<<endl;
    cout<<"Alamatnya = " <<p;
    _getch();
}

```

### 4. Mendeklarasikan struktur, memasukkan, dan menampilkan data struktur.

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include <iostream>
#include <conio.h>

using namespace std ;
#define MAX 10 //-->mx untuk nilai

struct dtnilai //-->prosedur struktur
{
    char nim[12];
    char nama[20];
    double nilai[MAX];
};

void main()
{
    struct dtnilai data;
    int i, jml;
    char strnilai[5], strjum[5];
    printf("NIM : ");
    gets(data.nim);
    cout<<endl;
    printf("Nama : ");

```

```

gets(data.nama);
cout<<endl;
printf("Jumlah Test : ");
gets(strjum);
jml = atoi(strjum);
cout<<endl;
for(i=0;i<jml;i++)
{
    printf("Nilai Test %d : ", i+1);
    gets(strnilai);
    data.nilai[i]=atof(strnilai);
}
cout<<endl;
printf("=====\\n");
cout<<endl;
printf("DATA MAHASISWA YANG TELAH DIINPUTKAN :
\\n"); cout<<endl;
printf("NIM :      %s\\n",data.nim);
cout<<endl;
printf("Nama :     %s\\n",data.nama);
cout<<endl;

for(i=0;i<jml;i++)
{
    printf("Nilai Test %d : %lf\\n", i+1, data.nilai[i]);
}
_getch();
}

```

5. Memesan tempat memori, mendeklarasikan, member nilai, mengcopy alamat pointer.

```

#include<stdio.h>
#include<stdlib.h>
#include <iostream>
#include <conio.h>

using namespace std ;

void p(void);
int *a,*b;

void main()
{
    p();
}
void p(void)
{
    a=(int *)malloc(sizeof(int));
    b=(int *)malloc(sizeof(int));
    *a=19;
    *b=5;
    a=b;
    *b=8;
    printf("Alamat a = %x\\t Isi a =
%d\\n",a,*a); printf("Alamat b = %x\\t
Isi b = %d\\n",b,*b); _getch();
}

```

Tugas!

- 1. Buatlah program menggunakan POINTER untuk merubah karakter yang dimasukkan dari huruf kecil menjadi huruf besar.**
- 2. Buatlah program untuk perhitungan karakter dengan ARRAY.**
- 3. Buatlah program array dalam struktur, yang menampilkan judul film dan tahun terbit dari data yang telah diinputkan. Data yang dimasukkan sebanyak**

**REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.
- Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012.



## POKOK BAHASAN 2

---

### LINKED LIST (SENARAI)

---

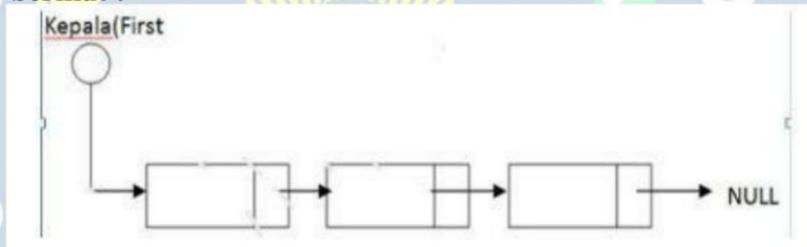
#### PENDAHULUAN

Pada pokok bahasan ini akan dibahas mengenai struktur data senarai (list) yang pembahasannya meliputi definisi dan representasi list, jenis-jenis list serta operasi-operasi dasar pada list. Sehingga setelah mempelajari bab ini diharapkan mahasiswa mampu :

- a. Menjelaskan definisi dan representasi list.
- b. Mengetahui jenis-jenis list.
- c. Memahami operasi-operasi pada list.

#### PENYAJIAN (TUTORIAL)

*Linked List* adalah sejumlah objek atau elemen yang dihubungkan satu dengan lainnya sehingga membentuk suatu list. Sedangkan objek atau elemen itu sendiri adalah merupakan gabungan beberapa data (variabel) yang dijadikan satu kelompok atau *structure* atau *record* yang dibentuk dengan perintah *struct*. Untuk menggabungkan objek satu dengan lainnya, diperlukan paling tidak sebuah variabel yang bertipe pointer. Syarat *linked list* adalah harus dapat diketahui alamat simpul pertama atau biasa dipakai variabel *First/Start/Header*. Struktur dasar sebuah list seperti gambar berikut :



**Gambar 2.1 List Tunggal**

Istilah-istilah dalam *linked list* :

- Simpul

Simpul terdiri dari dua bagian yaitu :

- a. Bagian data
- b. Bagian pointer yang menunjuk ke simpul berikutnya

- First/Header

Variabel First/Header berisi alamat (*pointer*)/acuan (*reference*) yang menunjuk lokasi simpul pertama *linked list*, digunakan sebagai awal penelusuran *linked list*.

- Nil/Null

Tidak bernilai, digunakan untuk menyatakan tidak mengacu ke manapun.

- Simpul Terakhir (Last)

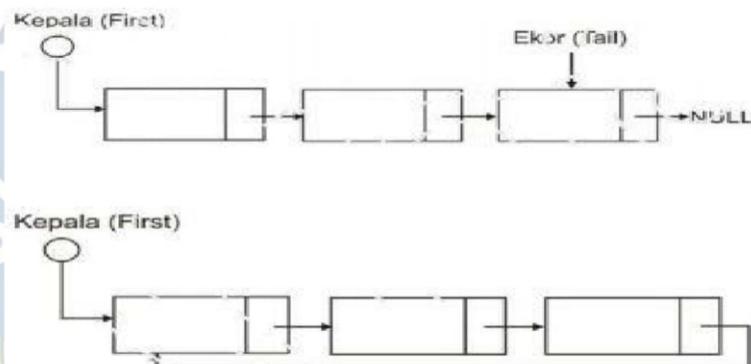
Simpul terakhir linked list berarti tidak menunjuk simpul berikutnya. Tidak terdapat alamat disimpan di *field pointer* (bagian kedua dari simpul). Nilai null atau nil disimpan di *field pointer* di simpul terakhir.

**Jenis-jenis linked list :****List Kosong**

List kosong hanya terdiri dari sebuah penunjuk elemen yang berisi NULL (kosong), tidak memiliki satu buah elemen pun sehingga hanya berupa penunjuk awal elemen berisi NULL.

**List Tunggal**

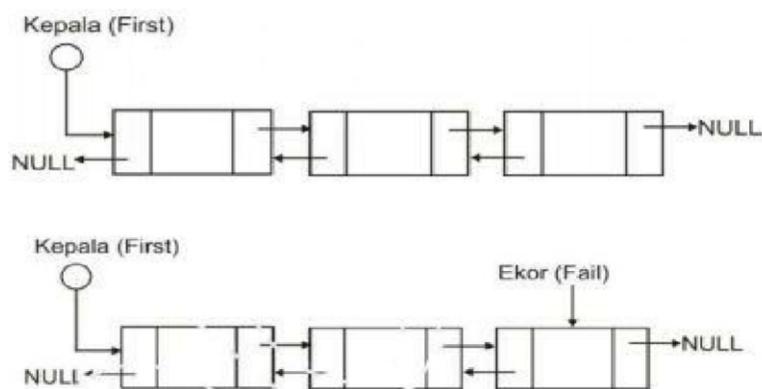
List tunggal adalah list yang elemennya hanya menyimpan informasi elemen setelahnya (*next*), sehingga jalannya pengaksesan list hanya dapat dilakukan secara maju. List tunggal terbagi menjadi tiga jenis yaitu list tunggal dengan kepala (*first*), list tunggal dengan kepala (*first*) dan ekor (*tail*), serta list tunggal yang berputar.



Gambar 2.2 List Tunggal dengan Kepala dan Ekor, List Tunggal Berputar

**List Ganda**

List ganda adalah sebuah list yang elemennya menyimpan informasi elemen sebelumnya dan informasi elemen setelahnya, sehingga proses penelusuran list dapat dilakukan secara maju dan mundur. List ganda terbagi menjadi tiga jenis yaitu list ganda dengan kepala (*first*), list ganda dengan kepala (*first*) dan ekor (*tail*), serta list ganda yang berputar.



Gambar 2.3 List ganda dengan Kepala, List ganda dengan Kepala dan Ekor

**Operasi Dasar pada Linked List :**

**IsEmpty :** Fungsi ini menentukan apakah linked list kosong atau tidak.

**Size :** operasi untuk mengirim jumlah elemen di *linked list*.

- Create* : operasi untuk penciptaan list baru yang kosong.
- Insertfirst* : operasi untuk penyisipan simpul sebagai simpul pertama.
- Insertafter* : operasi untuk penyisipan simpul setelah simpul tertentu.
- Insertlast* : operasi untuk penyisipan simpul sebagai simpul terakhir.
- Insertbefore* : operasi untuk penyisipan simpul sebelum simpul tertentu.
- Deletefirst* : operasi penghapusan simpul pertama.
- Deleteafter* : operasi untuk penghapusan setelah simpul tertentu.
- Deletelast* operasi penghapusan simpul terakhir.

## LEMBAR KERJA DAN TUGAS

- Contoh program sisip senarai (*linked list*).

```
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

using namespace std;

typedef struct nod
{
    int data;
    struct nod *next;
}NOD, *NODPTR;

void CiptaSenarai(NODPTR *s)
{
    *s = NULL;
}

NODPTR NodBaru(int m)
{
    NODPTR n;
    n = (NODPTR) malloc (sizeof(NOD));
    if (n != NULL)
    {
        n->data=m;
        n->next =
        NULL;
    }
    return n;
}

void SisipSenarai(NODPTR *s, NODPTR t, NODPTR p)
{
    if(p==NULL)
    {
        t->next=*s;
        *s=t;
    }
    else
    {
        t->next=p->next;
        p->next=t;
    }
}

void CetakSenarai (NODPTR s)
```

```

{
    NODPTR ps;
    for (ps=s; ps!=NULL; ps=ps->next)
        printf("%d --> ", ps->data);
    printf("NULL\n");
}

int main()
{
    NODPTR pel;
    NODPTR n;

    CiptaSenarai(&pel);
    n=NodBaru(55);
    SisipSenarai(&pel, n, NULL);

    n=NodBaru(75);
    SisipSenarai(&pel, n, NULL);
    CetakSenarai(pel);
    _getch();
}

```

2. Mendeklarasikan, memasukkan data, dan menampilkan data pada Single Linked List

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
#include <conio.h>

using namespace std ;

struct dtnilai *tampung;
struct dtnilai *ujung;
struct dtnilai *awal;
int j;
char strnilai[5], jawab[6];

struct dtnilai
{
    char nim[10];
    char nama[20];
    double nilai;
    struct dtnilai *next;
};

void main()
{
    printf("DATA MAHASISWA : \n");
    printf("===== \n");
    while(1)
    {
        if(j==0)
        {
            awal=(struct dtnilai*) malloc(sizeof(struct dtnilai));
            printf("NIM : ");
            gets(awal->nim);
            printf("\n");
            printf("Nama : ");
            gets(awal->nama);
            printf("\n");
            printf("Nilai Test : ");

```

```

        gets(strnilai);
        printf("\n");
        awal->nim = atof(strnilai);
        tampung=awal;
        tampung->next = NULL;
    }
else
{
    ujung=(struct dtnilai*) malloc(sizeof(struct dtnilai));
    tampung->next=ujung;
    printf("NIM : ");
    gets(ujung->nim);
    printf("\n");
    printf("Nama : ");
    gets(ujung->nama);
    printf("\n");
    printf("Nilai Test : ");
    gets(strnilai);
    printf("\n");
    ujung->nim=atof(strnilai);
    ujung->next=NULL;
    tampung=ujung;
}
printf("Masukkan Data lagi? (y/t) : ");
gets(jawab);
printf("\n");
if((strcmp(jawab, "Y")==0||strcmp(jawab, "y")==0))
{
    j++;
    continue;
}
else if((strcmp(jawab, "Y")==0||strcmp(jawab, "t")==0))
    break;
printf("Data mahasiswa yang telah diinputkan : \n");
printf("=====\\n");
printf("NIM\\tNama\\tNilai\\n");
printf("\n");
ujung=awal;
while(ujung!=NULL)
{
    printf("%s\\t%s\\t%6.2lf\\n",ujung->nim,ujung->nama,ujung-
>nim);
    ujung=ujung->next;
}
_getch();
}

```

### 3. Menghapus data pada simpul tertentu.

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
#include <conio.h>

using namespace std ;

struct dtnilai
{
    char nim[15];
    char nama[20];
    double nilai ;
    struct dtnilai *next;
}

```

```

};

void main ()
{
    printf("DATA MAHASISWA : \n");
    printf("=====\\n");
    struct dtnilai *tampung;
    struct dtnilai *ujung;
    struct dtnilai *awal;
    struct dtnilai *tanda;
    struct dtnilai *tanda2;

    char strnilai[10];
    char jawab[5];
    char hapus[5];
    int j=0;

    while (1)
    {
        if(j==0)
        {
            awal=(struct dtnilai*)malloc(sizeof(struct dtnilai));
            printf("NIM : ");
            gets(awal->nim);
            printf("\\n");
            printf("Nama : ");
            gets(awal->nama);
            printf("\\n");
            printf("Nilai Test : ");
            gets(strnilai);
            printf("\\n");
            awal->nilai = atof(strnilai);
            tampung=awal;
            tampung->next=NULL;
        }
        else
        {
            ujung=(struct dtnilai*)malloc(sizeof(struct dtnilai));
            tampung->next=ujung;
            printf("NIM : ");
            gets(ujung->nim);
            printf("\\n");
            printf("Nama : ");
            gets(ujung->nama);
            printf("\\n");
            printf("Nilai Test : ");
            gets(strnilai);
            printf("\\n");
            ujung->nilai=atof(strnilai);
            ujung->next=NULL;
            tampung=ujung;
        }
        printf("Masukkan Data Lagi? (y/t) : ");
        gets (jawab);
        printf("\\n");
        if(strcmp(jawab, "Y")==0||strcmp(jawab, "y")==0)
        {
            j++;
            continue;
        }
        else if((strcmp(jawab, "T")==0||strcmp(jawab, "t")==0))
            break;
    }
}

```

```

printf("\nData Mahasiswa yang Diinputkan : \n");
printf("=====\
\n");
printf("NIM \tNama \tNilai\n");
ujung=awal;

while (ujung!=NULL)
{
    printf("%s\t%s\t%6.2f\n",ujung->nim,ujung->nama,ujung-
>nilai); ujung=ujung->next;
}
while(1)
{
    printf("\n");
    printf ("NIM yang akan dihapus (NIM/(t/T)) :
\n"); gets (hapus) ;
    if ((strcmp (hapus, "t")==0 ||strcmp(hapus,"T")==0))
    {
        printf ("Masukkan NIM\n");
        continue;
    }
    else
        break;
}
if (strcmp(awal -> nim,hapus)==0)
{
    tanda=awal->next;

    free (awal);
    awal=tanda;
}
else
{
    tanda2=awal;
    tanda=tanda2->next;
    while (tanda!=NULL)
    {
        if (strcmp(tanda->nim,hapus)==0)
        {
            if (tanda->next!=NULL)
                tanda2->next=tanda->next;
            else
                tanda2->next=NULL;
            free (tanda);
            break;
        }
        tanda2=tanda;
        tanda=tanda->next;
    }
}
printf("\n");
printf ("Data Mahasiswa yang telah Dihapus (FIFO) :\n");
printf("=====\
\n");
printf ("NIM \tNama \tNilai\n");
ujung=awal;
while (ujung!=NULL)
{
    printf ("%s\t%s\t%6.2f\n", ujung ->nim, ujung -> nama,ujung -
>nilai);
    ujung=ujung -> next;
}
getch();
}

```

Tugas !

1. Buatlah program untuk menampilkan 10 bilangan secara menurun yaitu 10, 9, 8, sampai 1 dengan menggunakan LINKED LIST.
2. Buatlah sebuah program yang mengimplementasikan Linked List, dimana data yang dipakai adalah data buku yang ada dalam sebuah perpustakaan (ID Buku, Judul, dan Jumlah Buku). Program juga mengimplementasikan penambahan dan pengurangan simpul pada Linked List berdasarkan ID Buku.

**TULISKAN DAN JELASKAN SCRIPTNYA SERTA  
PRINTSCREEN HASILNYA !!!!!**

**REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.
- Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012.

## POKOK BAHASAN 3

### STACK (TUMPUKAN)

#### PENDAHULUAN

Pada pokok bahasan ini akan dibahas mengenai struktur data tumpukan atau stack, dimana stack merupakan suatu kumpulan data yang seolah-olah ada data yang diletakkan di atas data yang lain. Setelah mempelajari materi ini diharapkan mahasiswa mampu untuk :

- a. Mengetahui dan memahami definisi stack.
- b. Memahami operasi-operasi dasar stack.
- c. Memahami representasi statis dan dinamis stack.

#### PENYAJIAN (TUTORIAL)

Stack adalah kumpulan elemen-elemen yang tersimpan dalam suatu tumpukan. Aturan penyisipan dan penghapusan elemennya tertentu:

- Penyisipan selalu dilakukan "di atas" TOP
- Penghapusan selalu dilakukan pada TOP

Karena aturan penyisipan dan penghapusan semacam itu, TOP adalah satu-satunya alamat tempat terjadi operasi, elemen yang ditambahkan paling akhir akan menjadi elemen yang akan dihapus. Dikatakan bahwa elemen Stack tersusun secara LIFO (*Last In First Out*).

Seperti halnya jika kita mempunyai sebuah tumpukan buku, agar tumpukan buku itu tidak ambruk ketika kita mengambil sebuah buku di dalam tumpukan itu maka harus diambil satu per satu dari tumpukan yang paling atas dari tumpukan.



Gambar 3.1 Ilustrasi Stack

Perhatikan bahwa dengan definisi semacam ini, representasi tabel sangat tepat untuk mewakili stack, karena operasi penambahan dan pengurangan hanya dilakukan disalah satu ujung tabel.

Beberapa contoh penggunaan stack adalah pemanggilan prosedur, perhitungan ekspresi aritmatika, rekursifitas, backtracking, penanganan interupsi, dan lain-lain. Karakteristik penting stack sebagai berikut :

1. Elemen *stack* yaitu *item-item* data di elemen *stack*
2. TOP (elemen puncak dari *stack*)
3. Jumlah elemen pada *stack*
4. Status/kondisi *stack*, yaitu :
  - Penuh

Bila elemen di tumpukan mencapai kapasitas maksimum tumpukan. Pada kondisi ini, tidak mungkin dilakukan penambahan ke tumpukan. Penambahan di elemen menyebabkan kondisi kesalahan *Overflow*.

- **Kosong**

Bila tidak ada elemen tumpukan. Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen tumpukan. Pengambilan elemen menyebabkan kondisi kesalahan *Underflow*.

Stack memiliki operasi-operasi pokok sebagai berikut :

**Push** : Untuk menambahkan item pada tumpukan paling atas.

```
void Push (ItemType x, Stack *S)
{
    if (Full (S))
        Printf("Stack FULL");
    else
    {
        S->Item[S->Count]=x;
        ++(S->count);
    }
}
```

**Pop** : Untuk mengambil item teratas

```
int Pop (Stack S, ItemType x)
{
    if (Empty (S))
        Printf("Stack Kosong");
    else
    {
        --(S->Count);
        x=s->Item(s->Count);
    }
}
```

**Clear** : Untuk mengosongkan stack

```
void InitializeStack (Stack S)
{
    S->Count=0;
}
```

**IsEmpty** : Untuk memeriksa apakah stack kosong

```
int Empty (Stack *S)
{
    return (S->Count==0);
}
```

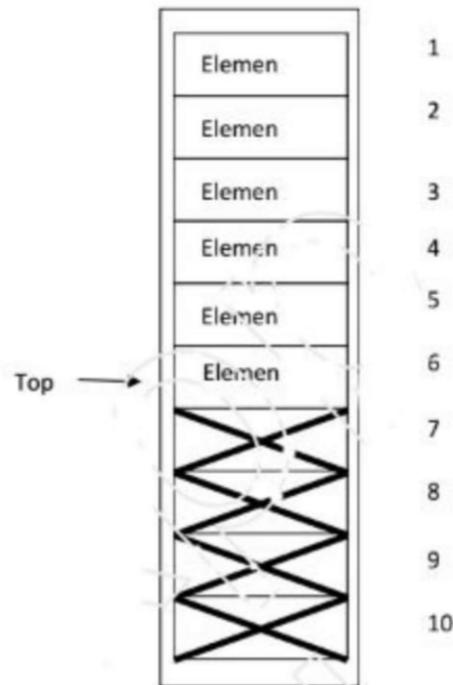
**IsFull** : Untuk memeriksa apakah stack sudah penuh

```
int Full (Stack S)
{
    return (S->Count==MAXSTACK);
}
```

**Representasi stack :**

- **Representasi statis**

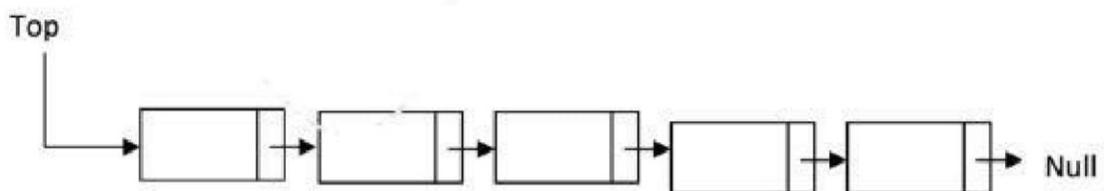
Stack dengan representasi statis biasanya diimplementasikan dengan menggunakan array. Sebuah array memiliki tempat yang dialokasikan diawal sehingga sebuah elemen yang dimasukkan dalam sebuah array terbatas pada tempat yang ada pada array. Karena menggunakan array maka stack dengan representasi statis dalam mengalami kondisi elemen penuh. Ilustrasi stack dengan representasi statis dapat dilihat pada gambar 3.2 :



Gambar 3.2 Representasi Stack Statis

- Representasi dinamis

Stack dengan representasi dinamis biasanya diimplementasikan dengan menggunakan pointer yang menunjuk pada elemen-elemen yang dialokasikan pada memori. Ilustrasi stack dengan representasi dinamis dapat dilihat pada gambar 3.3 :



Gambar 3.3 Representasi Stack Dinamis

Karena semua operasi pada sebuah stack diawali dengan elemen yang paling atas maka jika menggunakan representasi dinamis saat elemen ditambahkan akan menggunakan penambahan elemen pada awal stack (*addfirst*) dan saat pengambilan atau penghapusan elemen menggunakan penghapusan di awal stack (*delfirst*).

#### LEMBAR KERJA DAN TUGAS

##### 1) Program Stack :

```
#include <stdio.h>
#include <conio.h>
#include <iostream>

#define MAXSTACK 3

typedef int itemType;

typedef struct
{
```

```

        int item[MAXSTACK];
        int jml;
    } Stack;

void init(Stack *s)
{
    s->jml=0;
}

int kosong(Stack *s)
{
    return (s->jml==0);
}

int penuh(Stack *s)
{
    return (s->jml==MAXSTACK);
}

void isi(itemType x, Stack *s)
{
    if(penuh(s))
        printf("\nMAAF!!! Data PENUH\n");
    else{
        s->item[s->jml]=x;
        ++(s->jml);
    }
}

void ambil(Stack *s, itemType *x){
    if(kosong(s))
        printf("\nMaaf Data Kosong\n");
    else
    {
        --(s->jml);
        *x=s->item[s->jml];
        s->item[s->jml]=0;
        printf("\nData %i Berhasil Diambil\n",*x);
    }
}

void tampil(Stack *s){
    if(kosong(s))
        printf("\nMaaf Data Masih Kosong\n");
    else
        printf("\n");
        for(int i=s->jml-1;i>=0;i--){
            printf("Data: %d\n",s->item[i]);
        }
}

void hapus(Stack *s){

    s->jml=0;
    printf("\nSemua Data Berhasil Dihapus\n");
}

void main(){
    int pil;
    Stack tumpukan;
    itemType data;
    init(&tumpukan);

    do{

```

```

        printf("\nMENU: \n 1. Isi (Data Angka)\n 2. Ambil\n 3. Lihat\n 4.
Hapus (Hapus Semua Data)\n 5. Keluar\n");
        printf("\n");
        printf("Masukkan Pilihan : "); scanf("%i",&pil);

        switch(pil){
            case 1:
                printf("\nMasukkan Data Angka : ");
                scanf("%i",&data); isi(data,&tumpukan);
                break;
            case 2:
                ambil(&tumpukan,&data);
                break;
            case 3:
                tampil(&tumpukan);
                break;
            case 4:
                hapus(&tumpukan);
                break;
        }
    }while(pil!=5);

    getch();
}
}

```

- 2) Mendeklarasi, memasukkan, dan menampilkan elemen data pada sebuah stack dengan menggunakan array :

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <iostream>
#include <conio.h>

using namespace std ;

#define MAX 10

struct ttumpukan
{
    int elm[MAX];
    int puncak;
};

struct ttumpukan t;
int top;

void inisialisasi()
{
    t.puncak=0;
    top=0;
}

void push(int x)
{
    if(t.puncak==MAX)
        printf("Stack Sudah Penuh\n");
    else
    {
        t.puncak=top+1;
        t.elm[t.puncak]=x;
        top=t.puncak;
        printf("PUSH %d : %d\n",t.puncak,t.elm[t.puncak]);
    }
}

```

```

        }
    }
void pop()
{
    if(top==0)
        printf("Stack Kosong\n");
    else
    {
        printf("\nPOP %d = %d\n",top,t.elm[top]);
        t.puncak=top-1;
        top=t.puncak;
    }
}
void main()
{
    int i,jum;char strnilai[5],strjum[5];
    inisialisasi ();
    printf("Masukkan Jumlah Data : ");gets(strjum);
    printf("\n");
    jum=atoi(strjum);
    for(i=0;i<jum;i++)
    {
        printf("Nilai Integer %d : ",i+1);
        gets(strnilai);push(atoi(strnilai));
        printf("\n");
    }
    for(i=jum;i>0;i--)
        printf("Elemen Data :
        %d\n",t.elm[i]); for(i=0;i<jum;i++)
        pop();
    _getch();
}

```

3) Merepresentasikan stack menggunakan Linked List :

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include <iostream>
#include <conio.h>

using namespace std ;

struct sStack
{
    int isi;
    struct sStack*next;
};

int x,i=1,j=1;
struct sStack*atas, *bantu, *baru;
void push(int nilai)
{
    if(j==1)
    {
        atas=(struct sStack*)malloc(sizeof(struct sStack));
        atas->isi=nilai;
        atas->next=NULL;
    }
    else
    {
        baru=(struct sStack*)malloc(sizeof(struct sStack));

```

```

        baru->isi=nilai;
        baru->next=atas;
        atas=baru;
    }
    j++;
}
void pop()
{
    bantu=atas;
    printf("Data yang di POP %d : %d\n",i,bantu->isi);
    atas=atas->next;
    i++;
    free(bantu);
}
void main()
{
    char jawab[2],strNilai[5];
    while(1)
    {
        printf("Data yang di PUSH ke %d : ",j);
        gets(strNilai);x=atoi(strNilai);
        push(x);
        printf("Data lagi? <y/t> : ");
        gets(jawab);
        if((strcmp(jawab,"Y")==0)|| (strcmp(jawab,"y")==0))
            continue;
        else
            break;
    }
    j--;
    printf("Data nilai yang telah diinputkan
    :\n"); while(atas!=NULL)
        pop();
    getch();
}

```

Tugas!

1. Buatlah program stack statis yang menampilkan data yang di-*push* dan di-*pop*.
2. Diketahui data-data berikut : T, I, D, dan E.  
Buatlah program untuk memasukkan data-data di atas sehingga akan muncul tampilan berikut : T, E, D, I. Manfaatkan operasi PUSH dan POP.

**TULISKAN DAN JELASKAN SCRIPTNYA SERTA  
PRINTSCREEN HASILNYA !!!!!**

#### **REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.
- Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012.

## POKOK BAHASAN 4

---

### QUEUE (ANTRIAN)

---

#### PENDAHULUAN

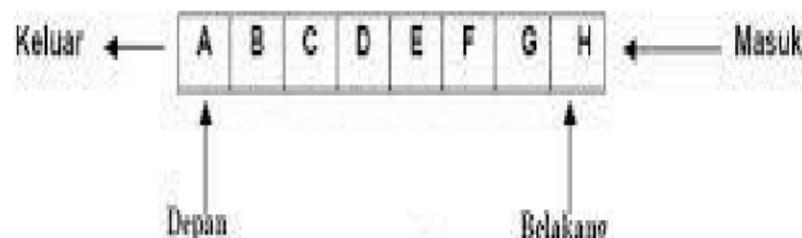
Pada pokok bahasan ini akan dibahas mengenai antrian atau queue, dimana struktur data ini hamper sama dengan tumpukan atau stack yang merupakan struktur data yang linier. Perbedaannya adalah pada operasi penambahan dan pengurangan pada ujung yang berbeda. Setelah mempelajari materi ini diharapkan mahasiswa mampu :

- Mengetahui dan memahami definisi antrian.
- Memahami operasi-operasi dasar pada antrian.
- Memahami representasi statis dan dinamis pada antrian.

#### PENYAJIAN (TUTORIAL)

Antrian adalah suatu kumpulan data yang penambahan elemennya hanya bisa dilakukan pada suatu ujung (disebut sisi belakang atau REAR), dan penghapusan atau pengambilan elemen dilakukan lewat ujung yang lain (disebut sisi depan atau FRONT). Prinsip yang digunakan dalam antrian ini adalah FIFO (*First In First Out*) yaitu elemen yang pertama kali masuk akan keluar pertama kalinya.

Penggunaan antrian antara lain simulasi antrian di dunia nyata (antrian pembelian tiket), sistem jaringan komputer (pemrosesan banyak paket yang datang dari banyak koneksi pada suatu *host, bridge, gateway*), dan lain-lain.



Gambar 4.1 Ilustrasi Antrian dengan

8 Elemen Karakteristik penting antrian sebagai berikut :

- Elemen antrian yaitu item-item data yang terdapat dalam antrian.
- Head/front* (elemen terdepan antrian).
- Tail/rear* (elemen terakhir antrian).
- Jumlah antrian pada antrian (*count*).
- Status/kondisi antrian, ada dua yaitu :
  - **Penuh**  
Bila elemen di antrian mencapai kapasitas maksimum antrian. Pada kondisi ini, tidak mungkin dilakukan penambahan ke antrian. Penambahan di elemen menyebabkan kondisi kesalahan *Overflow*.
  - **Kosong**  
Bila tidak ada elemen antrian. Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen antrian. Pengambilan elemen menyebabkan kondisi kesalahan *Underflow*.

**Operasi-operasi pokok pada antrian diantaranya adalah :**

1. Create Membuat antrian baru.

NOEL(CREATE(Q)) = 0

FRONT(CREATE(Q)) = tidak terdefinisi

REAR(CREATE(Q)) = tidak terdefinisi

2. IsEmpty Untuk memeriksa apakah Antrian sudah penuh atau belum.

ISEMPTY(Q) = True, jika Q adalah queue kosong.

3. IsFull mengecek apakah Antrian sudah penuh atau belum.

ISFULL(Q) = True, jika Q adalah queue penuh.

4. Enqueue/Insert menambahkan elemen ke dalam Antrian, penambahan elemen selalu

ditambahkan di elemen paling belakang.

REAR (INSERT(A,Q)) = A

ISEMPTY (INSERT(A,Q)) = FALSE

Algoritma QINSERT :

a. IF FRONT = 1 AND REAR = N, OR IF FRONT = REAR + 1, THEN OVERFLOW, RETURN

b. IF FRONT := NULL, THEN

SET FRONT := 1 AND REAR := 1

ELSE IF REAR = N,

THEN SET REAR

:= 1

ELSE

SET REAR := REAR+1

c. SET QUEUE[REAR] := ITEM

d. RETURN

5. Dequeue/Remove untuk menghapus elemen terdepan/pertama dari Antrian.

Algoritma QDELETE :

a. IF FRONT := NULL, THEN UNDERFLOW, RETURN

b. SET ITEM := QUEUE[FRONT]

c. [FIND NEW VALUE OF

FRONT] IF FRONT =

REAR, THEN

SET FRONT := NULL AND REAR

:= NULL ELSE IF FRONT = N, THEN

SET FRONT :=1

ELSE

SET FRONT := FRONT+1

d. RETURN

**Representasi**

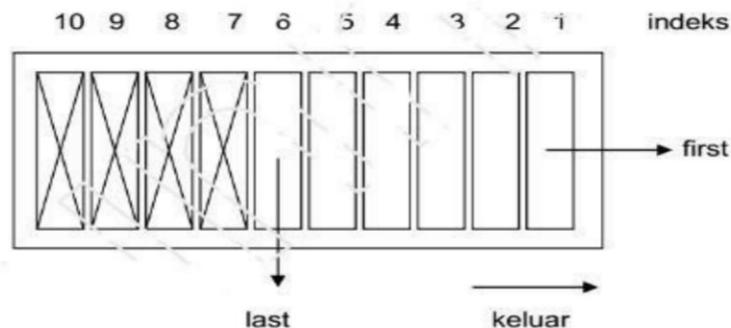
**queue :**

**Representasi**

**statis**

Queue dengan representasi statis biasanya diimplementasikan dengan menggunakan array. Sebuah array memiliki tempat yang dialokasikan diawal sehingga sebuah elemen yang dimasukkan dalam sebuah array terbatas pada tempat yang ada pada array. Karena menggunakan array maka queue dengan representasi statis dalam

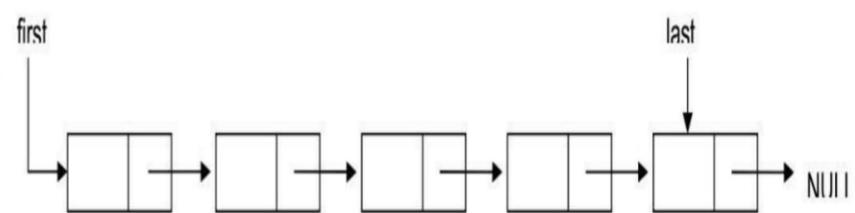
mengalami kondisi elemen penuh. Ilustrasi queue dengan representasi statis dapat dilihat pada gambar :



Gambar 4.2 Representasi Queue Statis

#### Representasi dinamis

Queue dengan representasi dinamis biasanya diimplementasikan dengan menggunakan pointer yang menunjuk pada elemen-elemen yang dialokasikan pada memori. Ilustrasi queue dengan representasi dinamis dapat dilihat pada gambar :



Gambar 4.3 Representasi Queue Dinamis

### LEMBAR KERJA DAN TUGAS

#### 1. Program Queue Statis :

```
#include <queue>
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    queue <int> que;
    que.push(10);
    que.push(2);
    que.push(3);

    cout<<"Paling depan : "<<que.front()<<endl;
    cout<<"Paling belakang : "<<que.back()<<endl;

    que.pop();
    cout<<"10 sudah dikeluarkan"<<endl;
    cout<<"Paling depan : "<<que.front()<<endl;
    cout<<"Paling belakang : "<<que.back()<<endl;
}
```

```

    que.push(6);
    cout<<"Angka 6 dimasukkan"<<endl;
    cout<<"Paling depan : "<<que.front()<<endl;
    cout<<"Paling belakang : "<<que.back()<<endl;

    _getch();
}

```

## 2. Program Queue :

```

#include <iostream>

using namespace std;

#define MAX 5           // Maximum Isi

class queue
{
private:
    int t[MAX];
    int al;           // tambah
    int dl;           // hapus

public:
    queue()
    {
        dl=-1;
        al=-1;
    }

    void del()
    {
        int tmp;
        if(dl== -1)
        {
            cout<<"Queue kosong";
        }
        else
        {
            for(int j=0;j<=al;j++)
            {
                if((j+1)<=al)
                {
                    tmp=t[j+1];
                    t[j]=tmp;
                }
                else
                {
                    al--;
                }
            }
            if(al== -1)
                dl=-1;
            else
                dl=0;
        }
    }

    void add(int item)
    {
        if(dl== -1 && al== -1)

```

```

    {
        dl++;
        al++;
    }
    else
    {
        al++;
        if(al==MAX)
        {
            cout<<"Queue penuh\n";
            al--;
            return;
        }
    }
    t[al]=item;
}

void display()
{
    if(dl!=-1)
    {
        for(int iter=0 ; iter<=al ; iter++)
            cout<<t[iter]<<" ";
    }
    else
        cout<<"Kosong";
}

int main()
{
    queue a;
    int data[5]={32,23,45,99,24};

    cout<<"Queue sebelum penambahan elemen : ";
    a.display();
    cout<<endl<<endl;

    for(int iter = 0 ; iter < 5 ; iter++)
    {
        a.add(data[iter]);
        cout<<"Penambahan Angka : "<<(iter+1)<<" : ";
        a.display();
        cout<<endl;
    }
    cout<<endl;
    cout<<"Queue setelah penambahan elemen : ";
    a.display();
    cout<<endl<<endl;

    for(int iter=0 ; iter < 5 ; iter++)
    {
        a.del();
        cout<<"Penghapusan Angka : "<<(iter+1)<<" : ";
        a.display();
        cout<<endl;
    }
    system("pause");
    return 0;
}

```

3. Buatlah program queue untuk memasukkan dan mengeluarkan data berupa huruf dalam antrian.

```
#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std;

typedef char Titem;

#define MAXN 3
typedef enum {NOT_OK, OK} Tboolean;

typedef struct
{
    Titem array[MAXN];
    int first;
    int last;
    int number_of_items;
}Tqueue;

void initialize_queue(Tqueue *Pqueue);
Tboolean enqueue(Tqueue *p, Titem item);
Tboolean dequeue(Tqueue *p, Titem *Pitem);
void print_queue(const Tqueue *Pqueue);

#include <ctype.h>
void main()
{
    Tqueue queue;
    Tboolean succeed;
    char chr;

    initialize_queue(&queue);
    printf("\n# Masukkan Sebuah Huruf Untuk Masuk Dalam Antrian\n");
    printf("\n# Atau Tekan Angka 1 Untuk Mengeluarkan Sebuah Huruf Dalam Antrian\n");
    printf("\n# Atau Tekan x Untuk Keluar :\n");
    printf("\n");
    chr = getche();
    while(chr != '0' && chr != '1')
    {
        if (isalpha(chr))
        {
            succeed=enqueue(&queue, chr);
            print_queue(&queue);
            if (!succeed)
                printf("\n Operasi Pemasukan Data Gagal \n");
        }
        if(chr == '1')
        {
            succeed = dequeue(&queue, &chr);
            if (succeed)
            {
                printf("\n");
                printf("\n Huruf Yang Keluar Dari Antrian %c adalah : ", chr);
                print_queue(&queue);
            }
        }
    }
}
```

```

        }
        else printf("\nOperasi Pengambilan Data Gagal\n");
    }
    chr = getche();
}
 getch();
}

void initialize_queue(Tqueue *Pqueue)
{
    Pqueue->first=0;
    Pqueue->last=-1;
    Pqueue->number_of_items=0;
}

Tboolean enqueue(Tqueue *Pqueue, Titem item)
{
    if (Pqueue->number_of_items >= MAXN)
        return(NOT_OK);
    else
    {
        Pqueue->last++;
        if (Pqueue->last > MAXN -1)
            Pqueue->last = 0;
        Pqueue->array[Pqueue->last] = item;
        Pqueue->number_of_items++;
        return(OK);
    }
}

Tboolean dequeue(Tqueue *Pqueue, Titem *Pitem)
{
    if(Pqueue->number_of_items == 0)
        return(NOT_OK);
    else
    {
        *Pitem = Pqueue->array[Pqueue->first++];
        if (Pqueue->first > MAXN - 1)
            Pqueue->first = 0;
        Pqueue->number_of_items--;
        return(OK);
    }
}

void print_queue(const Tqueue *Pqueue)
{
    int i, n;
    printf("\n");
    printf("\n Data Antrian Sekarang : \n\n");
    for (n = 1, i = Pqueue->first; n <= Pqueue->number_of_items; n++)
    {
        printf("%c", Pqueue->array[i]);
        i++;
        if(i > MAXN - 1)
            i = 0;
    }
    printf("\n\n");
}

```

#### 4. Program untuk pengurutan data secara ascending menggunakan QUEUE.

```

#include<stdio.h>
#include <iostream>
#include <conio.h>
#include <queue>

```

```

using namespace std ;
void main()
{
    queue <int> q;
    for(int i = 0; i < 10; i++)
        q.push(i);
    cout<<"Isi : ";

    do
    {
        cout<<' '<<q.front();
        q.pop();
    } while(!q.empty());

    cout<<'\n';
    _getch();
}

```

**Tugas!**

1. Buatlah program queue untuk memasukkan dan mengeluarkan data berupa angka dalam antrian (Modifikasi program latihan 3).
2. Buatlah program untuk pengurutan data secara *Descending* menggunakan *QUEUE* (Modifikasi program latihan 4).

**TULISKAN DAN JELASKAN SCRIPTNYA SERTA  
PRINTSCREEN HASILNYA !!!!!**

**REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.
- Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012.

## POKOK BAHASAN 5

---

### REKURSIF

---

#### PENDAHULUAN

Pada pokok bahasan ini akan dibahas mengenai rekursif. Setelah mempelajari bab ini diharapkan mahasiswa mampu :

- b. Mengetahui dan memahami definisi rekursif.
- c. Memahami sifat-sifat rekursif.
- d. Mengaplikasikan rekursif.

#### PENYAJIAN (TUTORIAL)

Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri, artinya fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri. Contoh menghitung nilai factorial. Rekursif 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 ke dalam stack sampai method pemanggilnya diselesaikan.

#### LEMBAR KERJA DAN TUGAS

##### 1. Program Bilangan Genap dan Bilangan Ganjil.

```
#include<iostream>
#include<conio.h>
using namespace std ;

void odd (int a);
void even(int a);

void main(void)
{
    int i;
    do
    {
        cout<<"Masukkan Bilangan 1 - 9 (0 untuk Keluar) : \n";
        cin>>i;
        odd(i);
        cout<<endl;
    } while
    (i!=0);
    _getch();
}

void odd(int a)
{
    if ((a%2) !=0) cout << "Bilangan GANJIL
\n"; else
        even (a);
}

void even(int a)
{
```

```

    if ((a%2) ==0) cout << "Bilangan
GENAP \n"; else
    odd (a);
}

```

## 2. Program Fibonacci.

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std ;

long fibonacci (long n)
{
if(n==1 || n==2)
    return(1);
else
    return fibonacci(n-1) + fibonacci(n-2);
}

void main()
{
    int x;
    printf("Mencari Nilai Fibonacci \n");
    printf("Masukkan Nilai X : ");
    scanf("%d", &x);
    printf("Nilai Fibonacci dari %d = %d \n", x, fibonacci(x));
    system("pause");
    _getch();
}

```

## 3. Program faktorial.

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std ;

int faktorial (int n)
{
    if(n==1)
        return(1);
    else
        return (n*faktorial(n-1));
}

void main()
{
    int x;
    printf("Mencari Nilai Faktorial \n");
    printf("Masukkan Nilai X : ");
    scanf("%d", &x);
    printf("Nilai Faktorial dari %d= %d \n", x, faktorial(x));
    system("pause");
    _getch();
}

```

## 4. Program penjumlahan kuadrat sejumlah bilangan menggunakan iterasi.

$$\text{JumKuadrat } (m,n) = m^2 + (m+1)^2 + (m+2)^2 + \dots + n^2 \text{ dengan } m \leq n.$$

$$\text{JumKuadrat } (5,9) = 5^2 + 6^2 + 7^2 + 8^2 + 9^2 = 255$$

```

#include<stdio.h>
#include<stdlib.h>

```



**Tugas !**

- 1. Buatlah program untuk menentukan bilangan yang terbesar dan terkecil dari dua buah bilangan yang diinputkan.**
- 2. Buatlah program dengan cara rekursi untuk menampilkan perkalian 3 buah bilangan, dimana ketiga bilangan tersebut nilainya diinputkan.**
- 3. Buatlah program untuk mengurutkan bilangan secara *Descending* dengan menggunakan *Rekursif*.**
- 4. Buatlah program perhitungan gaji karyawan. Dengan ketentuan :**
  - a. Jika gaji kotor  $\leq 1.500.000$  maka mendapat potongan 10%.**
  - b. Jika gaji kotor  $> 1.500.000$  maka mendapat potongan 15%.**

**TULISKAN DAN JELASKAN SCRIPTNYA SERTA PRINTSCREEN  
HASILNYA !!!!!**

**REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.
- Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012.

## POKOK BAHASAN 6

---

### SORTING (PENGURUTAN)

---

#### PENDAHULUAN

Setelah mempelajari bab ini diharapkan mahasiswa mampu :

- a. Menunjukkan beberapa algoritma dalam pengurutan.
- b. Menunjukkan bahwa pengurutan merupakan suatu persoalan yang bisa diselesaikan dengan sejumlah algoritma yang berbeda satu sama lain.
- c. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

#### PENYAJIAN (TUTORIAL)

Pengurutan data (*sorting*) didefinisikan sebagai suatu proses untuk menyusun kembali himpunan obyek menggunakan aturan tertentu. Ada dua macam urutan yang biasa digunakan dalam proses pengurutan yaitu :

- Urutan naik (*ascending*) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar.
- Urutan turun (*descending*) yaitu dari data yang mempunyai nilai paling besar sampai paling kecil.

Contoh : data bilangan 5, 2, 6, dan 4 dapat diurutkan naik menjadi 2, 4, 5, 6 atau diurutkan turun menjadi 6, 5, 4, 2. Pada data yang bertipe char, nilai data dikatakan lebih kecil atau lebih besar dari yang lain didasarkan pada urutan relatif (*collating sequence*) seperti dinyatakan dalam tabel ASCII. Keuntungan dari data yang sudah dalam keadaan terurut yaitu :

Data mudah dicari, mudah untuk dibetulkan, dihapus, disisipi atau digabungkan. Dalam keadaan terurutkan, kita mudah melakukan pengecekan apakah ada data yang hilang. Misalnya kamus bahasa, buku telepon.

Mempercepat proses pencarian data yang harus dilakukan berulang kali.

Beberapa faktor yang berpengaruh pada efektifitas suatu algoritma pengurutan antara lain :

Banyak data yang diurutkan.

Kapasitas pengingat apakah mampu menyimpan semua data yg kita miliki.

Tempat penyimpanan data, misalnya piringan, pita atau kartu, dll.

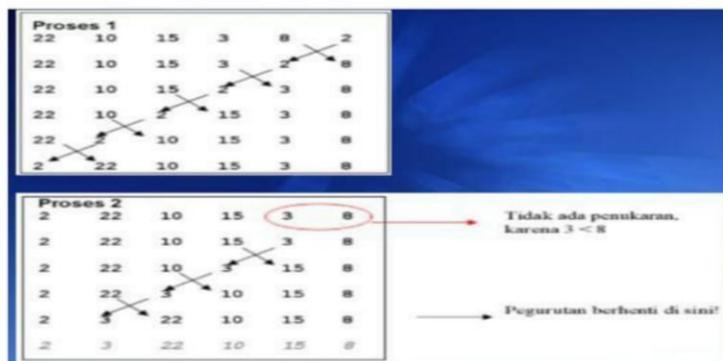
Beberapa algoritma metode pengurutan dan prosedurnya sebagai berikut :

##### 1. *Bubble Sort*

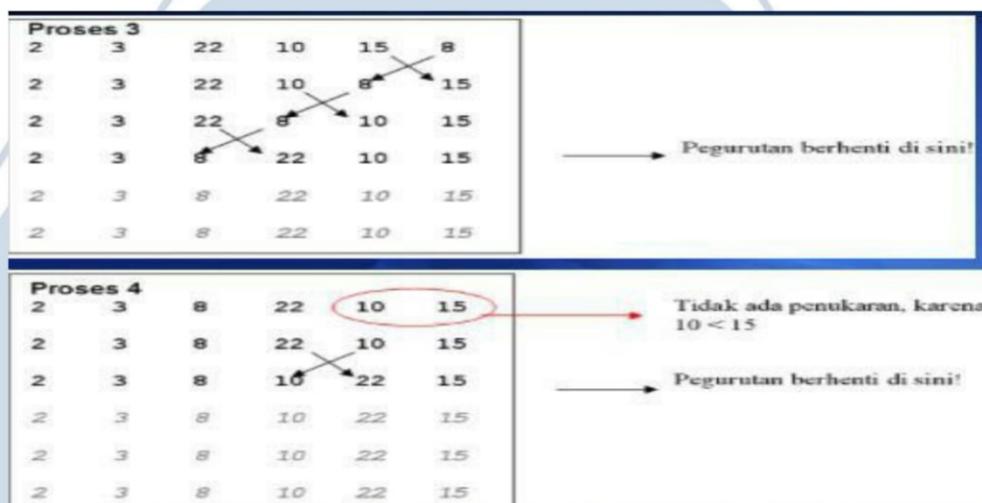
*Bubble Sort* adalah suatu metode pengurutan yang membandingkan elemen yang sekarang dengan elemen berikutnya. Apabila elemen sekarang  $>$  elemen berikutnya, maka posisinya ditukar. Kalau tidak, tidak perlu ditukar. Diberi nama “Bubble” karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.

Proses Bubble Sort :

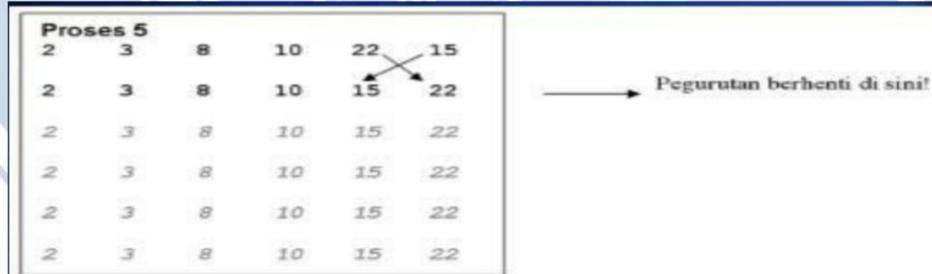
Data paling akhir dibandingkan dengan data di depannya, jika ternyata lebih kecil atau besar maka tukar sesuai dengan ketentuan (*descending* atau *ascending*). Dan pengecekan yang sama dilakukan terhadap data yang selanjutnya sampai dengan data yang paling awal



Gambar 6.1 Langkah 1 Bubble Sort



Gambar 6.2 Langkah 2 Bubble Sort



Gambar 6.3 Langkah 3 Bubble Sort

Algoritma *Bubble Sort* :

1.  $i = 0$
2. selama ( $i < N-1$ ) kerjakan baris 3 sampai 7
3.  $j = N - 1$
4. Selama ( $j \geq i$ ) kerjakan baris 5 sampai 7
5. Jika ( $Data[j-1] > Data[j]$ ) maka tukar  $Data[j-1]$  dengan  $Data[j]$
6.  $j = j - 1$
7.  $i = i + 1$

Prosedur yang menggunakan metode gelembung :

```
void BubbleSort()
{
    int i, j;
    for(i=1; i<Max-1; i++)
        for(j=Max-1; j>=i; j--)
            if(Data[j-1] > Data[j])
                Tukar(&Data[j-1], &Data[j]);
}
```

## 2. Selection Sort

Metode seleksi melakukan pengurutan dengan cara mencari data yang terkecil kemudian menukarkannya dengan data yang digunakan sebagai acuan atau sering dinamakan pivot. Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses. Proses pengurutan dengan metode seleksi dapat dijelaskan sebagai berikut :

Langkah pertama dicari data terkecil dari data pertama sampai data terakhir. Kemudian data terkecil ditukar dengan data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding data yang lain.

Langkah kedua, data terkecil kita cari mulai dari data kedua sampai terakhir. Data terkecil yang kita peroleh ditukar dengan data kedua dan demikian seterusnya sampai semua elemen dalam keadaan terurutkan.

<b>Proses 1</b> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>32</td><td>75</td><td>69</td><td>58</td><td>21</td><td>40</td></tr> </table> <p>Pembanding      Posisi            32 &lt; 75      0            32 &lt; 69      0            32 &lt; 58      0            32 &gt; 21 (tukar idx) 4            21 &lt; 40      4</p> <p>Tukar data ke-0 (32) dengan data ke-4 (21)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>75</td><td>69</td><td>58</td><td>32</td><td>40</td></tr> </table>	0	1	2	3	4	5	32	75	69	58	21	40	0	1	2	3	4	5	21	75	69	58	32	40	<b>Proses 3</b> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td><b>69</b></td><td>58</td><td>75</td><td>40</td></tr> </table> <p>Pembanding      Posisi            69 &gt; 58 (tukar idx) 3            58 &lt; 75      3            58 &gt; 40      5</p> <p>Tukar data ke-2 (69) dengan data ke-5 (40)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>40</td><td>58</td><td>75</td><td>69</td></tr> </table>	0	1	2	3	4	5	21	32	<b>69</b>	58	75	40	0	1	2	3	4	5	21	32	40	58	75	69	<b>Proses 5</b> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>40</td><td>58</td><td><b>75</b></td><td>69</td></tr> </table> <p>Pembanding      Posisi            75 &gt; 69      5</p> <p>Tukar data ke-4 (75) dengan data ke-5 (69)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>40</td><td>58</td><td>69</td><td>75</td></tr> </table>	0	1	2	3	4	5	21	32	40	58	<b>75</b>	69	0	1	2	3	4	5	21	32	40	58	69	75
0	1	2	3	4	5																																																																					
32	75	69	58	21	40																																																																					
0	1	2	3	4	5																																																																					
21	75	69	58	32	40																																																																					
0	1	2	3	4	5																																																																					
21	32	<b>69</b>	58	75	40																																																																					
0	1	2	3	4	5																																																																					
21	32	40	58	75	69																																																																					
0	1	2	3	4	5																																																																					
21	32	40	58	<b>75</b>	69																																																																					
0	1	2	3	4	5																																																																					
21	32	40	58	69	75																																																																					
<b>Proses 2</b> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td><b>75</b></td><td>69</td><td>58</td><td>32</td><td>40</td></tr> </table> <p>Pembanding      Posisi            75 &gt; 69 (tukar idx) 2            69 &gt; 58 (tukar idx) 3            58 &gt; 32 (tukar idx) 4            32 &lt; 40      4</p> <p>Tukar data ke-1 (75) dengan data ke-4 (32)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>69</td><td>58</td><td>75</td><td>40</td></tr> </table>	0	1	2	3	4	5	21	<b>75</b>	69	58	32	40	0	1	2	3	4	5	21	32	69	58	75	40	<b>Proses 4</b> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>40</td><td><b>58</b></td><td>75</td><td>69</td></tr> </table> <p>Pembanding      Posisi            58 &lt; 75      3            58 &lt; 69      3</p> <p>Tukar data ke-3 (58) dengan data ke-3 (58)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>21</td><td>32</td><td>40</td><td>58</td><td>75</td><td>69</td></tr> </table>	0	1	2	3	4	5	21	32	40	<b>58</b>	75	69	0	1	2	3	4	5	21	32	40	58	75	69																									
0	1	2	3	4	5																																																																					
21	<b>75</b>	69	58	32	40																																																																					
0	1	2	3	4	5																																																																					
21	32	69	58	75	40																																																																					
0	1	2	3	4	5																																																																					
21	32	40	<b>58</b>	75	69																																																																					
0	1	2	3	4	5																																																																					
21	32	40	58	75	69																																																																					

Gambar 6.4 Langkah Selection Sort

Algoritma seleksi dapat dituliskan sebagai berikut :

1.  $i = 0$
2. selama ( $i < N-1$ ) kerjakan baris 3 sampai dengan 9
3.  $k = i$
4.  $j = i + 1$
5. Selama ( $j < N$ ) kerjakan baris 6 dan 7
6. Jika ( $Data[k] > Data[j]$ ) maka  $k = j$

7.  $j = j + 1$
8. Tukar Data[i] dengan Data[k]
9.  $i = i + 1$

Di bawah ini merupakan prosedur yang menggunakan metode seleksi :

```
void SelectionSort()
{
    int i, j, k;
    for(i=0; i<Max-1;i++)
    {
        k = i;
        for (j=i+1; j<Max; j++)
        if(Data[k] > Data[j])
            k = j;
        Tukar(&Data[i], &Data[k]);
    }
}
```

### 3. Merger Sort

Algoritma Merge Sort ialah algoritma pengurutan yang berdasarkan pada strategi *divide and conquer*. Algoritma ini terdiri dari dua bagian utama, pembagian list yang diberikan untuk di-sort ke dalam beberapa sublist yang lebih kecil, dan *sort* (mengurutkan) dan *merge* (menggabungkan) *sublist-sublist* yang lebih kecil ke dalam list hasil yang sudah diurutkan. Pembagian bisa dikatakan cukup mudah karena sublist-sublist tersebut dibagi ke dalam dua sublist yang ukurannya adalah setengah dari ukuran semula. Hal ini terus diulang sampai sublist itu cukup kecil untuk di-sort secara efisien (umumnya telah terdiri dari satu atau dua elemen). Dalam langkah merge dua *sublist* disatukan kembali dan diurutkan pada saat yang sama. Algoritma untuk *merge sort* ialah sebagai berikut :

- A. Untuk kasus  $n=1$ , maka table a sudah terurut sendirinya (langkah solve)
- B. Untuk kasus  $n \geq 1$ , maka :
  - a. **DIVIDE:** bagi table a menjadi dua bagian, bagian kiri dan bagian kanan, masing-masing bagian berukuran  $n/2$  elemen.
  - b. **CONQUER:** secara rekursif, terapkan algoritma D-and-C pada masing-masing bagian.
  - c. **MERGE:** gabung hasil pengurutan kedua bagian sehingga diperoleh table a yang terurut.



## LEMBAR KERJA DAN TUGAS

### 1. Program ascending dengan menggunakan *bubble sort*

```
#include <iostream>
#include <conio.h>
#include <iomanip>

using namespace std ;

void main (void)
{
    int dataku[] = {5, 34, 32, 25, 75, 42, 2};
    int adaPertukaran;
    int n;

    cout << "Data BELUM diurutkan : \n";

    for (int ctr = 0; ctr<7; ctr++)
    {
        cout << setw( 3 ) << dataku[ctr];
    }

    cout << endl << endl;

//PENGURUTAN
do {
    adaPertukaran = 0;

    for (int i = 0; i < 7-1; i++){
        if (dataku[i+1] < dataku[i]){
            n = dataku[i];
            dataku[i] = dataku[i+1];
            dataku[i+1] = n;
            adaPertukaran = 1;
        }
    }
}
```

```

} while (adaPertukaran == 1);

//MENAMPILKAN HASIL PENGURUTAN
cout << "Data SETELAH diurutkan : \n";
for (int i = 0; i < 7; i++){
    cout << dataku[i];
    cout << " ";
}

_getch();
}

```

## 2. Program aplikasi array untuk mengurutkan bilangan dengan metode bubble sort

```

#include <stdio.h>
#include <iostream>
#include <conio.h>

#define MAX 20
void input(int jum);
void buble(int jum);
void output(int jum);
int n, A[MAX];

using namespace std ;

void main()
{
    printf("Masukkan Jumlah Bilangan : ");
    scanf("%d" ,&n);
    cout<<endl;
    input(n);
    buble(n);
    output(n);
    _getch();
}

void input (int jum)
{
    int i;
    for(i=0;i<jum;i++)
    {
        printf("Bilangan ke %d : ",i+1);
        scanf("%d", &A[i]);
        cout<<endl;
    }
    cout<<endl;
    cout<<"Hasil Pengurutan Secara Ascending :
\n"; cout<<endl;
}
void buble(int jum)
{
    int i,j,temp;
    for(i=1;i<=jum-1;i++)
    {
        for (j=i;j<n;j++)
        {
            if (A[i-1]>A[j])
            {
                temp=A[i-1];
                A[i-1]=A[j];
                A[j]=temp;
            }
        }
    }
}

```

```

        }
    }
void output (int jum)
{
    int i;
    for(i=0;i<jum;i++)
    {
        printf("Bilangan ke %d =
        %d\n",i+1,A[i]); cout<<endl;
    }
}

```

### 3. Program ascending dengan menggunakan *selection sort*

```

#include <iostream>
#include <conio.h>

using namespace std ;

int data[10],data2[10];
int n;

void Select(int a, int b)
{
    int t;
    t = data[b];
    data[b] = data[a];
    data[a] = t;
}
void selection_sort()
{
    int pos,i,j;
    for(i=1;i<=n-1;i++)
    {
        pos = i;
        for(j = i+1;j<=n;j++)
        {
            if(data[j] < data[pos]) pos = j;
        }
        if(pos != i) Select(pos,i);
    }
}

void main()
{
    cout<<"====PROGRAM SELECTION
SORT===="<<endl; cout<<endl;
    //Input Data
    cout<<"Masukkan Jumlah Data : ";
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cout<<"Masukkan Data Ke - "<<i<<" : ";
        cin>>data[i];
        data2[i]=data[i];
    }

    selection_sort();

    cout<<"\n\n";
}

```

```

//Menampilkan Data
cout<<"Data Setelah di Sort : ";
for(int i=1; i<=n; i++)
{
    cout<<" " <<data[i];
}
cout<<"\n\n Proses Selesai \n";
_getch();
}

```

#### 4. Program ascending dengan menggunakan merger sort

```

#include <stdio.h>
#define MAX 5
int Data[MAX];
int temp[MAX];

// Procedure Merger Sort
void merge(int Data[], int temp[], int kiri, int tengah, int kanan)
{
    int i, left_end, num_elements, tmp_pos;
    left_end = tengah - 1;
    tmp_pos = kiri;
    num_elements = kanan - kiri + 1;

    while ((kiri <= left_end) && (tengah <= kanan))
    {
        if (Data[kiri] <= Data[tengah])
        {
            temp[tmp_pos] = Data[kiri];
            tmp_pos = tmp_pos + 1;
            kiri = kiri + 1;
        }
        else
        {
            temp[tmp_pos] = Data[tengah];
            tmp_pos = tmp_pos + 1;
            tengah = tengah + 1;
        }
    }
    while (kiri <= left_end)
    {
        temp[tmp_pos] = Data[kiri];
        kiri = kiri + 1;
        tmp_pos = tmp_pos + 1;
    }
    while (tengah <= kanan)
    {
        temp[tmp_pos] = Data[tengah];
        tengah = tengah + 1;
        tmp_pos = tmp_pos + 1;
    }

    for (i=0; i <= num_elements; i++)
    {
        Data[kanan] = temp[kanan];
        kanan = kanan - 1;
    }
}

// Prosedur Kumpulan Data
void m_sort(int Data[], int temp[], int kiri, int kanan)
{
    int tengah;
    if (kanan > kiri)

```

```

    {
        tengah = (kanan + kiri) / 2;
        m_sort(Data, temp, kiri, tengah);
        m_sort(Data, temp, tengah+1, kanan);
        merge(Data, temp, kiri, tengah+1,
              kanan);
    }
}

void mergeSort(int Data[], int temp[], int array_size)
{
    m_sort(Data, temp, 0, array_size - 1);
}

int main()
{
    int i;
    printf("Masukkan DATA SEBELUM TERURUT :
\n"); for (i = 0; i < MAX; i++) {
        printf ("Data ke %i : ", i+1);
        scanf ("%d", &Data[i]);
    }

    mergeSort(Data, temp, MAX);
    printf("\n DATA SETELAH TERURUT : ");
    for (i = 0; i < MAX; i++)
    printf("%d ", Data[i]);
    printf("\n");
    scanf("%d");
    return(0);
}

```

**TUGAS !**

- Buatlah program untuk mengurutkan data-data berikut ini secara *Descending* dengan metode *Bubble Sort*.**  
4, 8, 5, 9, 6, 2, 7, 5, 9, 5, 23, 30
- Buatlah program untuk mengurutkan data-data berikut ini secara *Descending* dengan metode *Selection Sort*.**
- Buatlah program untuk mengurutkan data-data berikut ini secara *Descending* dengan metode *Merge Sort*.**

**REFERENSI**

- Fachrurrozi M., *Modul Praktikum Struktur Data*, Laboratorium Universitas Brawijaya Malang, Malang, 2006.
- Haryanto Bambang, *Struktur Data*, Informatika, Bandung, 2008.
- Kristanto Andi, *Struktur Data dengan C++*, Graha Ilmu, Yogyakarta, 2009.  
Warni Elly, *Buku Bahan Ajar Struktur Data*, Universitas Hasanuddin, 2012

## LAMPIRAN

### Riwayat Hidup Penulis

Hindarto, lahir di Surabaya tanggal 30 Juli 1973. Setelah menyelesaikan Sekolah Dasar, SLTP, dan SLTA di Surabaya, Melanjutkan Pendidikan di Politeknik Elektronika ITS Surabaya kemudian transer S1 di Jurusan Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Sidoarjo lulus tahun 2003. Meraih gelar Magister Teknik (MT) di Teknik Elektro Fakultas Teknologi Industri ITS surabaya tahun 2007. Saat ini penulis masih dalam proses studi lanjut Doctor (S3) di Teknik Elektro Fakultas Teknologi Industri ITS surabaya dan menjabat Dekan di Fakultas Teknik Universitas Muhammadiyah Sidoarjo. Mata Kuliah yang di ampuh penulis yaitu Sistem Digital, Algoritma & Struktur Data, Kecerdasan Buatan, Mikroprosesor, dan Jaringan Syaraf Tiruan.

Ade Eviyanti, lahir di Jakarta tanggal 24 Juni 1978. Setelah menyelesaikan Sekolah Dasar, SLTP, dan SLTA di Makassar, melanjutkan Pendidikan ke Universitas Muhammadiyah Sidoarjo. Meraih gelar sarjana (S.Kom) Fakultas Teknik Prodi Informatika tahun 2003. Saat ini penulis masih dalam proses study lanjut (S2) di Sekolah Tinggi Teknik surabaya (STTS) Prodi Teknologi Informasi. Kini menjabat Ketua Program Studi Diploma III Teknik Informatika UMSIDA. Aktifitas keseharian berhidmat di Pimpinan Daerah Aisyiyah Sidoarjo. Mata Kuliah yang di ampuh penulis yaitu Algoritma dan Pemrograman, Algoritma dan Struktur Data, Basis Data, Kecerdasan Buatan, Pengantar Teknologi Informasi, dan Jaringan Syaraf Tiruan.

Yunianita Rahmawati, lahir di Magetan tanggal 1 Juni 1985. Setelah menyelesaikan Sekolah Dasar, SLTP, dan SLTA di Jawa Timur, melanjutkan Pendidikan ke STIKOM (Sekolah Tinggi Manajemen Informatika & Teknik Komputer) Surabaya. Meraih gelar sarjana (S.Kom) Jurusan Sistem Informasi tahun 2008. Saat ini penulis masih dalam proses study lanjut (S2) di Sekolah Tinggi Teknik surabaya (STTS) Prodi Teknologi Informasi. Mata Kuliah yang di ampuh penulis yaitu Matematika Diskrit, Aljabar Linier, Grafika Komputer, Pengembangan Aplikasi Berbasis Web, Algoritma & Struktur Data, dan Jaringan Syaraf Tiruan.