BAB 2

TINJAUAN PUSTAKA

2.1 Sistem Informasi

Menurut Davis dan Kertahadi dalam buku yang ditulis oleh Al Fatta(2007), pengertian sistem informasi harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Data merupakan nilai, keadaan, atau sifat yang berdiri sendiri lepas dari konteks apapun. Sementara informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang.

2.2 Sistem Informasi Akademik

Sistem informasi akademik adalah perangkat lunak yang digunakan untuk menyajikan informasi dan menata administrasi yang berhubungan dengan kegiatan akademis, dan diharapkan kegiatan administrasi akademis dapat dikelola dengan baik dan informasi yang diperlukan dapat diperoleh dengan mudah dan cepat. Sistem informasi akademik adalah sistem informasi yang diciptakan dan digunakan untuk memenuhi kebutuhan manajemen kampus seperti, (Saputra, 2012):

1. KRS (Kartu Rencana Studi).
2. KHS (Kartu Hasil Studi).
3. Nilai mahasiswa.
4. Manajemen dosen.

5. Manajemen mata kuliah dan jadwal mata kuliah.

6. Manajemen program studi, jurusan dan fakultas.

7. Manajemen mahasiswa.

8. Manajemen pembayaran *(optional)*.

2.3 Kondisi Program Terdahulu

SIAKAD (Sistem Informasi Akademik) Universitas Lampung (Unila) memiliki perangkat lunak tambahan yang disebut Matahari yang berguna untuk membantu proses administrasi masing - masing jurusan di Unila. Matahari merupakan sistem yang dibangun dengan memodifikasi *source code moodle.org*. Pengembangan sistem Matahari dengan penambahan *utility* pencetakan pembimbing akademik yang dilakukan di Jurusan Ilmu Komputer memiliki tujuan yaitu untuk memudahkan pengecekan dan pencetakan daftar pembimbing akademik.

Utility Pembimbing Akademik memiliki tampilan sebagai berikut:

1. Halaman Utama

   Halaman utama dapat tampil, ketika *user* memasukkan alamat pada *browser* yaitu *adpc15.puskom.unila.ac.id* atau 192.168.20.74 yang hanya dapat diakses menggunakan SSH (Secure Shell). Unila membatasi wilayah akses hanya dengan intranet. Tampilan Halaman Utama dapat dilihat pada Gambar 2.1

Gambar 2.1: Halaman Utama

Pada Gambar 2.1, bagian paling kiri merupakan menu pilihan untuk *user* memilih kegiatan-kegiatan administrasi lainnya. Salah satunya adalah menu daftar pembimbing akademik yang dikembangkan oleh Choiranti Efrina pada tahun 2013.

2. Halaman Pilih Program Studi

   Pada proses pemilihan program studi dilakukan oleh *user* yang tidak harus *login*. Sistem ini, semua *utility*nya bersifat terbuka untuk data-data yang bersifat umum karena yang memiliki *login* hanya *admin*. Tampilan dapat dilihat pada Gambar 2.2.

Gambar 2.2: Halaman Pilih Program Studi

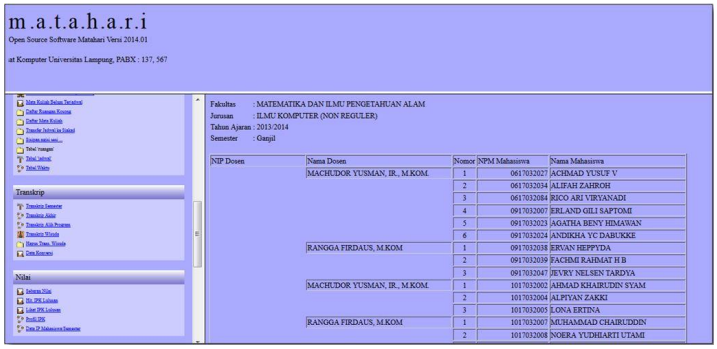
Gambar 2.2 merupakan gambar yang menunjukaan *user* untuk memilih fakultas, program studi, tahun ajaran dan semester.

3. Halaman Hasil Proses

   Setelah memilih semua nama Fakultas , Program Studi, Tahun Ajaran dan Semester seperti contoh Gambar 2.3

Gambar 2.3: Memilih program studi

Setelah itu, diperoleh tampilan seperti Gambar 2.4 berikut:



Gambar 2.4: Hasil *Running* program.

Pencetakan fakultas, program studi, dan tahun ajaran yang telah dipilih pada tampilan Gambar 2.4 diperoleh dari *script* berikut ini:

```
//cari nama jurusan
$xkode_program_studi=substr($mkode_program_studi,2,3);
$mNAMA_JUR =cari_program_studi
($mkode_fak,$xkode_program_studi);

//cari nama fakultas
$NAMA_FAK=cari_fakultas($mkode_fak);

echo"<table border=0>";
echo"<tr>";
echo"<td>Fakultas";
echo"<td>:</td>";
echo"<td>$NAMA_FAK</td>";
echo"</tr>";

echo"<tr>";
echo"<td>Jurusan";
echo"<td>:</td>";
echo"<td>$mNAMA_JUR</td>";
echo"</tr>";

echo"<tr>";
echo"<td>Tahun Ajaran";
echo"<td>:</td>";
echo"<td>$mtahun_ajaran</td>";
echo"</tr>";

echo"<tr>";
echo"<td>Semester";
echo"<td>:</td>";

if ($msemester='1')
{
   echo"<td>Ganjil</td>";
}
else
{
   echo"<td>Genap</td>";
}

echo"</tr>";
echo"</table>";
echo"<br>";
```

`substr` merupakan fungsi untuk mengambil substring dari suatu string

dengan sintaks sebagai berikut:

substr (string string, int start [, int length]).

String merupakan *string* yang diambil *substringnya*. *Int start* adalah posisi awal *substring* yang diambil. *Int length* adalah banyaknya karakter yang diambil mulai dari posisi awal (bersifat opsional). Jika argumen ketiga tidak disertakan, maka *substring* yang diambil dimulai dari posisi awal yang ditentukan hingga akhir *string*.

Pencetakan nip dosen, nama dosen, nomor, npm mahasiswa dan nama mahasiswa pada tampilan Gambar 2.4 diperoleh dari *script* berikut ini:

```
//cari nama mhs dan PA
$urut = 0;
while (OCIFetch($cari_nama_mkx))
{
   $cari_nama_mk = OCIParse($db,"select
   nip_p_akademik,
   nama_lengkap from mahasiswa
   where npm='$cnpm'");
   OCIDefineByName($cari_nama_mk,"NAMA_LENGKAP",$cnama_mhs);
   OCIDefineByName($cari_nama_mk,"NIP_P_AKADEMIK",$cnip);
   OCIExecute($cari_nama_mk);

   while (OCIFetch($cari_nama_mk))
   {
      //cari nama PA
      $cari_nama_mkq = OCIParse($db,"select
      nama,nip_baru from dosen
      where nip='$cnip'");
      OCIDefineByName($cari_nama_mkq,"NAMA",$cnama_dosen);
      OCIDefineByName($cari_nama_mkq,"NIP_BARU",$cnip_baru);
      OCIExecute($cari_nama_mkq);

      while (OCIFetch($cari_nama_mkq))
      {
         $dosen[$urut] = $cnama_dosen;
         $nip[$urut] = $cnip_baru;
         $nama_mahasiswa[$urut]=$cnama_mhs;
         $npm[$urut] = $cnpm;
         $urut=$urut+1;
      }
```

```
    }
}
```

2.4 Oracle

Oracle adalah nama produk *software database* produksi perusahaan *software Oracle Corporation*. *Oracle* merupakan perusahaan pertama yang membuat server *database* relasional, meskipun idenya sebenarnya lahir dari seorang yang bekerja di IBM.Dengan kepopuleran *Oracle*, SQL juga ikut populer sehingga saat ini menjadi suatu bahasa standar dalam manajemen *database*.

Untuk mendukung akses *database Oracle* dari lingkungan PHP, PHP menyediakan dukungan terhadap beberapa ekstensi, namun yang dipakai pada sistem ini adalah OCI8 *(Oracle 8 Call-Interface)* yaitu ekstensi yang dapat mendukung paling banyak fungsi yang terdapat di dalam *Oracle Call Interface* (OCI) (Raharjo, 2011):

2.5 Oracle 8 Functions

Beberapa fungsi yang disediakan oleh *database Oracle* 8 dan *Oracle* 7 digunakan oleh OCI8 *(Oracle 8 Call-Interface)*. Dalam menggunakan ekstensi ini dibutuhkan *Oracle8 client libraries*. Ekstensi ini lebih fleksibel dibanding dengan standar ekstensi *Oracle*, mendukung secara global dan lingkungan variable- variable PHP (Bakken dan Egon, 2001).

Beberapa fungsi *Oracle* yang digunakan pada pengembangan *utility* daftar pembimbing akademik :

(a) *OCIDefineByName*

Mengambil kolom SQL dan menjadikan variable PHP. Harus berhati-hati karena dalam *Oracle* semua nama kolom menggunakan huruf besar

(ALL UP-PERCASE), dan pada pemilihan nama kolom dapat menggunakan huruf kecil *(lower case)*.

(b) *OCIExecute*

Melaksanakan sebuah pernyataan sebelumnya

(c) *OCIFetch*

Mengambil baris selanjutnya (untuk pernyataan *SELECT*) ke dalam *result-buffer* bagian dalam.

(d) *OCIParse*

Menyiapkan pernyataan *Oracle* untuk eksekusi.

Berikut contoh penggunaan fungsi- fungsi tersebut :

```
while (OCIFetch($cari_nama_mkx))
{
   $cari_nama_mk = OCIParse($db,"select nip_p_akademik,
                   nama_lengkap from mahasiswa
                   where npm='$cnpm'");
   OCIDefineByName($cari_nama_mk,"NAMA_LENGKAP",$cnama_mhs);
   OCIDefineByName($cari_nama_mk,"NIP_P_AKADEMIK",$cnip);
   OCIExecute($cari_nama_mk);
```

2.6 PHP

PHP adalah akronim dari *Hypertext Prepocessor*, yaitu suatu bahasa pemrograman berbasiskan kode-kode *(script)* yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke *web browser* menjadi kode HTML. Kode PHP mempunyai ciri-ciri khusus, yaitu:

(a) Hanya dapat dijalankan menggunakan *web server*, misal: *Apache*.

(b) Kode PHP diletakkan dan dijalankan di *web server*.

(c) Kode PHP dapat digunakan untuk mengakses *database*, seperti: *MySQL*, *PostgreSQL*, *Oracle* dan lain-lain.

(d) Merupakan *software* yang bersifat *open source*.

(e) Gratis untuk di-*download* dan digunakan.

(f) Memiliki sifat *multiplatform*, artinya dapat dijalankan menggunakan sistem operasi apapun, seperti: *Linux*, *Unix*, *Windows*, dan lain-lain.

PHP merupakan sebuah bahasa, sehingga memerlukan tatacara/aturan dalam membuat kode-kodenya. Kode PHP diawali dengan tanda <**?php** dan diakhiri dengan tanda **?**>. Blok kode PHP selalu diapit oleh kedua tanda tersebut. Untuk setiap perintah diakhiri dengan tanda ; (titik koma). Berikut contoh kode PHP:

```
<?php
     echo "Hello World";
?>
```

Kode PHP juga dapat dikombinasikan dengan kode HTML, seperti contoh berikut (Oktavian, 2010):

```
<html>
   <body>
           <?php
              echo "Hello World";
           ?>
   </body>
</html>
```

Beberapa sintaks PHP yang digunakan pada skrip (Gilmore, 2010):

- <?php ... ?>

Disebut sebagai *delimiter syntax* yaitu suatu dokumen PHP diawali dan diakhiri oleh sintaks ini

- `include (/path/to/file X) atau include "/path/to/file X"`

Mengevaluasi kemudian ”melampirkan” file X bersama *script*. Sama artinya menyalin semua isi *file* X ke lokasi di dalam *script* dimana fungsi `include()` dipanggil.

- `extract($_POST)`

`POST` merupakan salah satu metode pengiriman data dari satu skrip ke skrip lainnya. Setiap data yang dilewatkan biasanya berbentuk variabel. Perintah `extract`, singkatnya, mengimpor variabel-variabel ini ke *script* yang memanggilnya. Adapun `$_POST` adalah istilah yang digunakan untuk merujuk kepada data tersebut.

- `void echo(string argument1 [, ...string argumentn])`

Mencetak satu atau lebih string

- `while (expression) { statements }`

Menangani proses perulangan *(looping)*. `statements` akan terus dieksekusi selama `expression` bernilai `TRUE`. Jika sebaliknya, perulangan dihentikan.

- `Operator += (assignment)`

`a+=b` berarti `a=a+b`. Operator ini mengisi variabel di ruas kiri dengan nilai di ruas kanan.

- `Operator ++ (increment)`

`a++` berarti `a=a+1`. Operator ini menaikkan nilai variabel di ruas kiri sebanyak satu satuan.

2.6.1 Variabel

Variabel merupakan sebuah istilah yang menyatakan sebuah tempat yang digunakan untuk menampung nilai-nilai dimana nilai di dalamnya bisa diubah-ubah. Variabel memungkinkan untuk menciptakan rumus maupun nilai *operand*-nya bisa dialokasikan secara dinamis.

Variabel dikenali dengan adanya tanda ($) yang kemudian bisa diikuti dengan angka, huruf dan *underscore*. Namun variabel tidak bisa mengandung *white space*/spasi.

Untuk mendifinisikan variabel, *programmer* hanya perlu menuliskannya, maka otomatis variabel dikenali oleh PHP parser. Berikut ini contoh pendifinisian variabel:

```
$nama
$0_alamat
$no_telp
$_pekerjaan
```

Data yang dimuat oleh variabel bisa berupa *null* (belum ada jenisnya), angka, *string*, *array* (kumpulan variabel yang bertipe data sama), dan isinya bisa diubah-ubah. Berikut ini cara mendeklarasikan dan mengalokasikan nilai variabel secara bersamaan (Zaki, 2008):

```
$nama        = "Choiranti Efrina" ;
$0_alamat    = "Rajabasa" ;
$no_telp     = 123456789101 ;
$_pekerjaan  = "Mahasiswi" ;
```

2.6.2 Kondisional

Pernyataan kondisional yang terdapat pada suatu program berguna untuk menentukan statemen yang dieksekusi, yang didasarkan pada kondisi-kondisi tertentu yang didefinisikan. Jika kondisi terpenuhi (bernilai *true*) maka *statemen* yang terdapat di dalam blok bersangkutan akan dieksekusi. Sebaliknya jika tidak terpenuhi (bernilai *false*) maka *statemen* tersebut diabaikan oleh program. Dalam PHP, pemilihan *statemen* dapat dilakukan dengan dua cara, yaitu (Raharjo, 2012):

1. Menggunakan *statemen if*
2. Menggunakan *statemen switch*

Berikut contoh penggunaan *statemen if* :

```
for ($j=1; $j<$urut; $j++)
{
  $status=TRUE;
  for ($k=0 ; $k<$urut-1;$k++)
  {


    if (strcmp($nip[$k],$nip[$k+1])>0)


    {
       $status=FALSE;
       //tukar nip dosen
       $temp=$nip[$k];
       $nip[$k]=$nip[$k+1];
       $nip[$k+1]=$temp;
       //tukar nama dosen
       $temp=$dosen[$k];
       $dosen[$k]=$dosen[$k+1];
       $dosen[$k+1]=$temp;
       //tukar npm mahasiswa
       $temp=$npm[$k];
       $npm[$k]=$npm[$k+1];
       $npm[$k+1]=$temp;
       //tukar nama mhs
```

```
        $temp=$nama_mahasiswa[$k];
        $nama_mahasiswa[$k]=$nama_mahasiswa[$k+1];
        $nama_mahasiswa[$k+1]=$temp;
      }
  }
```

```
if($status==TRUE)break;
}
```

`strcmp()` merupakan fungsi untuk membandingkan dua *string* dengan sintaks:

`strcmp(string str1, string str2)`

str1 dan *str2* adalah string yang dibandingkan. Hasil pembandingan dengan fungsi ini (Peranginangin, 2006):

- <0 jika str1 lebih kecil dari str2
- =0 jika str1 sama dengan str2
- >0 jika str1 lebih besar dari str2

Perlu diperhatikan bahwa dalam PHP, kondisi harus berada di dalam tanda kurung. Jika tidak, maka program tidak bisa dijalankan. Bentuk *if* tersebut hanya memiliki satu kondisi. Ini berarti bahwa jika kondisi tersebut tidak terpenuhi (bernilai *false*), maka di setiap blok pada *statement* tersebut tidak akan dilaksanakan dan eksekusi program akan dilanjutkan ke *statemen* yang terdapat di bawah blok tersebut.

Terdapat fungsi **`if...else`** yang digunakan dalam script :

```
$kode_jdwl_mk = substr($mtahun_ajaran,2,2).$mkode_fak.$msemester;
$kode_npm     = '1'.$mkode_fak.substr($mkode_program_studi,2,3);
if (($mkode_program_studi == '74032') or
   ($mkode_program_studi  == '74051'))
   {
   $cari_nama_mkx = OCIParse($db,"select  distinct(npm) from
                                  krs_mahasiswa
                                  where ( (npm like '%17051%') or
                                  npm like '%17032%') )
```

```
                                    and jadwal_mk_id=
                                   '$kode_jdwl_mk'");
    }

else {
$cari_nama_mkx = OCIParse($db,"select  distinct(npm) from
                 krs_mahasiswa
                 where  npm like '%$kode_npm%'
                 and jadwal_mk_id='$kode_jdwl_mk'");
 }
OCIDefineByName($cari_nama_mkx,"NPM",$cnpm);
OCIExecute($cari_nama_mkx);
```

Perintah bentuk **`if... else...`** pada prinsipnya mirip dengan perintah `if`, karena jika suatu *kondisi* terpenuhi (benar), maka lakukan *blok pernyataan1*. Jika tidak terpenuhi (salah) lakukan *blok pernyataan2*.

Fungsi `distinct` digunakan untuk menghilangkan nilai ganda. Jadi nilai yang sama dan diulang- ulan dieksekusi satu nilai saja (Heryanto, 2006)

2.6.3 Perulangan

Dalam mengembangankan aplikasi *web* kerap dijumpai kasus-kasus yang menuntut untuk melakukan pengulangan terhadap *statemen-statemen* tertentu. PHP menyediakan beberapa bentuk blok pengulangan, yaitu *while, do-while, for* dan *foreach*.

Bentuk umum statemen *for* adalah sebagai berikut (Raharjo, 2012):

```
for ($j=1; $j<$urut; $j++)
{
   $status=TRUE;

   for ($k=0 ; $k<$urut-1;$k++)
```

```
   {
      if (strcmp($nip[$k],$nip[$k+1])>0)
      {
        ....
        ....
      }
   }
if($status==TRUE)break;
}
```

Statement `for` merupakan *statement* yang digunakan untuk mengulang blok pernyataan dalam jumlah yang ditentukan berdasarkan inisialisasi awal atau akhir kondisi, serta nilai penambahan atau pengurangan yang ditentukan dengan sintaks (Peranginangin, 2006):

```
for (inisialisasi; kondisi; increment)
{
blok pernyataan;

}
```

• *inisialisasi* sebagai nilai awal

• *kondisi* diuji; jika bernilai *true*(benar), maka perulangan dilanjutkan dengan mengerjakan *blok pernyataan*, sedangkan jika bernilai *false*(salah), maka perulangan berhenti dan blok pernyataan dilompati

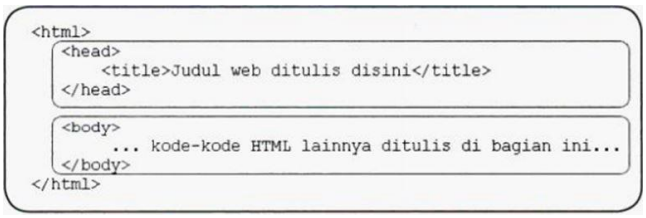
• Jika *blok pernyataan* hanya terdiri dari satu baris, maka tanda kurung kurawal({}) dapat ditiadakan.

• *increment* merupakan nilai penambahan atau pengurangan untuk mengulangi pengerjaan *blok pernyataan* setelah penambahan atau pengurangan yang nilai kebenarannya diuji apakah kondisi masih terpenuhi.

2.7 HTML *(HyperText Markup Language)*

HTML adalah suatu bahasa yang dikenal oleh *web browser* untuk menampilkan informasi dengan lebih menarik dibandingkan dengan tulisan teks biasa (*plain text*). Sedangkan *web browser* adalah program komputer yang digunakan untuk membaca HTML, kemudian menerjemahkan dan menampilkan hasilnya secara visual ke layar komputer.

Karena sebuah bahasa, maka HTML mempunyai aturan dan struktur tertentu untuk menuliskan perintah-perintahnya yang biasa dinamakan dengan TAG HTML. Aturan tersebut diawali dengan lambang <tag> dan biasanya diakhiri dengan lambang </tag>. Berikut gambaran struktur HTML.



Gambar 2.5: Struktur HTML

Dari struktur di atas, terlihat bahwa penulisan kode-kode HTML lainnya untuk keperluan isi situs *web* diletakkan di bagian tag <body>. Dengan demikian dapat dikatakan bahwa informasi yang berupa kode-kode diapit oleh tag awal dan tag akhir, dan sebuah apitan tag bisa juga diapit oleh tag lainnya.

Dalam HTML ada beberapa tag yang tidak perlu diakhiri/ditutup, misal tag
. Selain itu ada beberapa tag yang memiliki atribut-atribut untuk pengaturan teks maupun halaman, misalnya *tag anchor* yang sering dituliskan dengan lambang <a> dan memiliki atribut **href**, **rel name** dan sebagainya. Contoh penulisan tag yang memiliki atribut seperti berikut:

  (Oktavian, 2010).

Tag- tag yang terdapat dalam skrip (HTML Reference;HTML5 Compliant, 2014):

• <html></html>

Suatu dokumen HTML diawali dan diakhiri oleh tag ini, yang disebut *root tag*.

• <body></body>

Mendefinisikan tubuh dokumen.

• <table></table>

Mendefinisikan tabel.

• <tr></tr>

Mendefinisikan baris tabel.

• <td></td>

Mendefinisikan sel tabel.

•

Menyisipkan karakter *line-break* (1x tombol *Enter*)

• Mendefinisikan paragraf di dalam dokumen.

2.8 MySQL

MySQL adalah sebuah software *database*. MySQL merupakan tipe dara relasional yang artinya MySQL menyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan. Keuntungan menyimpan data di database adalah kemudahannya dalam penyimpanan dan menampilkan data karena dalam bentuk tabel.

Ada banyak *database* untuk PHP, namun MySQL merupakan *software database* yang paling disarankan. Berikut ini adalah keuntungan MySQL:

1. Gratis dan *opern source*.
2. Ada versi komersialnya juga, digunakan jika ingin memberikan dukungan teknis.

3. Biaya yang harus dikeluarkan jauh lebih murah
4. Tersedia di banyak platform.
5. Menggunakan standar penulisan SQL ANSI.

SQL singkatan dari *Structured Query Language*. PHP menggunakan SQL untuk berkomunikasi dengan *database* dan melakukan pengolahan data. Agar dapat mengolah *database* perlu menggunakan sql *statement*. Ada tiga perintah sql *statement*, yaitu CREATE untuk membuat, INSERT untuk memasukkan data, dan DELETE untuk menghapus data.

CREATE adalah perintah untuk membuat *database* atau tabel. Misalnya untuk membuat tabel digunakan perintah:

```
CREATE TABLE nama_tabel ( nama_field_1 jenis_field),
nama_field_2 VARCHAR(30), dst);
```

Sementara untuk membuat *database* digunakan perintah berikut:

```
CREATE DATABASE nama_database;
```

Untuk menambahkan data ke tabel, bisa menggunakan perintah INSERT:

```
INSERT INTO nama_tabel (field_1, field_2, dst) VALUES
('konten_1', 'konten_2', dst);
```

Untuk menghapus data di tabel, bisa menggunakan perintah DELETE:

```
DELETE FROM nama_tabel WHERE isi_field = 'nilai tertentu';
```

Untuk menampilkan isi tabel di *database*, bisa menggunakan perintah (Zaki, 2008):

```
SELECT nama_field FROM nama_tabel WHERE kondisi;
```

2.9 Bubble Sort

Algoritma *Bubble sort* membuat sejumlah penukaran pada elemen-elemen yang hendak diurutkan. Dalam setiap penukaran itu, sebelumnya dilakukan perbandingan antara elemen-elemen yang bersebelahan. Jika urutan elemen-elemen ini tidak benar maka mereka ditukar . Setiap penukaran dimulai dari elemen yang paling awal, jadi pada pertukaran pertama, elemen 1 dan 2 yang dibandingkan, kemudian elemen 2 dan 3, kemudian elemen 3 dan 4 dan seterusnya hingga seluruh elemen terurut dengan benar. (McConnel, 2001)

Bubble sort yang digunakan pada program sebelumnya :

```
for ($j=1; $j<$urut; $j++)
{
  $status=TRUE;
  for ($k=0 ; $k<$urut-1;$k++)
  {
     if (strcmp($nip[$k],$nip[$k+1])>0)
     {
       $status=FALSE;
       $temp=$nip[$k];
       $nip[$k]=$nip[$k+1];
       $nip[$k+1]=$temp;
       $temp=$dosen[$k];
       $dosen[$k]=$dosen[$k+1];
       $dosen[$k+1]=$temp;
       $temp=$npm[$k];
       $npm[$k]=$npm[$k+1];
       $npm[$k+1]=$temp;
       $temp=$nama_mahasiswa[$k];
       $nama_mahasiswa[$k]=$nama_mahasiswa[$k+1];
       $nama_mahasiswa[$k+1]=$temp;
     }
   }
   if($status==TRUE)break;
}
```

2.10 Selection Sort

Selection Sort merupakan konsep *sorting* dengan mencari (memilih) nilai terkecil dan menukarnya dengan elemen paling awal (paling kiri) pada setiap tahap, sehingga hasil proses tiap tahap (jika n=7, maka dilakukan n-1 =6 tahap) dapat digambarkan sebagai berikut (Sjukani,2010):

Data asli	15	10	7	22	17	5	12	Perpindahan Perubahan
Tahap I	5	10	7	22	17	15	12	kolom 0 - 6 (6 kali perbandingan)
Tahap II	5	7	10	22	17	15	12	kolom 1 - 6 (5 kali perbandingan)
Tahap III	5	7	10	22	17	15	12	kolom 2 - 6 (4 kali perbandingan)
Tahap IV	5	7	10	12	17	15	22	kolom 3 - 6 (3 kali perbandingan)
Tahap V	5	7	10	12	15	17	22	kolom 4 - 6 (2 kali perbandingan)
Tahap VI	5	7	10	12	15	17	22	kolom 5 - 6 (1 kali perbandingan)

Berikut contoh program yang menggunakan *selection sort*:

```
#include <stdio.h>
#define n 7
void main()
{
int A [n] = {15, 10, 7, 22, 17, 5, 12};
int X, I, K;
printf ('Sebelum disort\n");
    for (I=0; I<= n-1; I++)
       printf ("%d", A[I]);
       printf ("\n");

       K = 0;
       while (K <= n-2)
       {
          J = K; I=K+1;
          while (I<= n-1)
          {
             if (A[I] < A[J]){ J=I ;} I++;
          }

          X= A[J];
          A[J]=A[K];
          A[K]=X;
          K++;
```

```
        }

printf ("Sesudah Sort \n");
for (I=0 ; I<=n-1; I++){
printf ("%d", A[I]);}
}
```

2.11 Quick Sort

Quick sort diketahui sebagai algoritma *sorting* yang paling cepat. *Running time* rata-ratanya *O(n log n).* kecepatannya ini dikarenakan optimisasi *looping* yang sangat cepat dan sangat tinngi. *Worts case running time*nya $O(n^2)$.

Untuk mengurutkan *array* S, algoritma ini terdiri dari 4 langkah (Weiss, 2007):

1. Jika nomor elemen dalam S adalah 0 atau 1 maka kembali
2. Ambil, elemen *v* apa saja dalam *S*, elemen ini disebut pivot
3. Partisi S-{v} (elemen yang tersisa dari S) ke dalam 2 kelompok yang saling lepas : $S_1 = \{x \varepsilon S\text{-} \{v\}— x \leq v\}$, dan $S_2 = \{x \varepsilon S\text{-} \{v\}/x \geq v\}$
4. Kembalikan *quick sort* (S_1) diikuti dengan v diikuti juga dengan $O(n^2)$ *quick-sort* (S_2)

2.12 Running Time

Running time merupakan waktu yang dibutuhkan untuk mengeksekusi setiap instruksi di dalam program sampai selesai. Di dalam program terdapat beberapa operasi yaitu penambahan (+), pengurangan (-), perkalian (*), pembagian (/), *return* (pengembalian nilai dari fungsi), inisialisasi, dan perbandingan dimana setiap operasi ini *running time*nya masing-masing 1 unit.

Berikut contoh program untuk menghitung *running time* (Weiss, 2007):

```
unsigned int
sum (int)
{
unsigned int i, partial_sum;
/*1*/ partial_sum = 0;
/*2*/ for ( i=1); i<=n; i++)
/*3*/ partial_sum += i*i*i;
/*4*/ return partial_sum;
```

Berikut contoh program dengan menggunakan *array* untuk menghitung *running time*:

```
unsigned int
sum (int)
{
unsigned int i;
/*1*/ unsigned int partial_sum = new array(0,0,0,0,0,...);
/*2*/ for ( i=0; i<=n; i++)
/*3*/ partial_sum[i] += i*i*i;
/*4*/ return partial_sum[n];
```

Perhitungan pada program tidak menggunakan *array*: Karena dalam sebuah program diketahui memiliki operasi penambahan (*addition*), pengurangan (*substraction*), perkalian (*multiplication*), pembagian (*division*), pemberian nilai (*assignment/initialization*), kondisional (*conditional*), perulangan (*looping*), dan pengembalian nilai (*sintaks return*) bernilai 1 unit waktu.

Running time dari baris 1 yang berisi perintah inisialisasi dan baris 4 yang berisi pengembalian nilai dari fungsi masing-masing bernilai 1 unit. Pada baris 2, meskipun hanya terdapat satu kalimat perulangan, namun melibatkan tiga operasi yang tiap-tiapnya menghabiskan *running time* berbeda.

Ketiga buah operasi ini adalah inisialisasi `(i=1)` yang bernilai 1 unit saja karena hanya dieksekusi di awal perulangan, kondisional `(i<=n)` yang dieksekusi

sebanyak `n+1` kali, dan penambahan `(i++)` yang dieksekusi sebanyak n kali. Jika demikian, maka total *running time* dari baris 2 adalah `1+(n+1)+n = 2n+2`. Baris 3 sendiri terdiri dari 3 operasi, yakni 2 operasi perkalian dan 1 operasi penambahan, sehingga *running time*-nya bernilai 3.

Namun, mengingat baris ini berada di dalam sintaks perulangan, maka baris tersebut dieksekusi sepanjang perulangan bernilai *True* (yang berarti **n** kali), sehingga *running time* seluruhnya adalah **3n**.

Dari uraian di atas, diperoleh *running time* per baris:

Baris 1 = **[1]**

Baris 2 = **[2n+2]**

Baris 3 = **[3n]**

Baris 4 = **[1]**.

Running time keseluruhan program **(T(n))** sama dengan total *running time* dari semua baris yang membentuk program tersebut, yang dalam kasus ini adalah:

T(n) = 1+(2n+2)+3n+1 = 5n+4 unit waktu.

Pada program dengan menggunakan *array*, penghitungan *running time*nya sama dengan program sebelumnya karena operasi di dalamnya sama. Namun, jumlah memori yang digunakan pada program dengan *array* lebih banyak.

Sumber hasil analisis *running time* dinyatakan pada tabel 2.1 dan 2.2:

Menurut Singh A, dkk (2013)

Tabel 2.1: *Case Table* menurut Singh A, dkk

Name	Best	Average	Worst	Memory	Method
Bubble sort	n	n^2	n^2	1	Exchanging
Insertion sort	n	n^2	n^2	1	Insertion sort
Selection sort	n^2	n^2	n^2	1	Selection
Heap sort	n log n	n log n	n log n	1	Selection
Merge sort	n log n	n log n	n log n	Depends	Merging
Quick sort	n log n	n log n	n2	log n	Partitioning

Menurut Rao D, dkk (2013)

Tabel 2.2: *Case Table* menurut Rao D,dkk

Sort	Order	Worst case	Memory	Stable	Data type	Complexity
Bubble sort	n^2	n^2	nk+np	yes	all	very low
Selection sort	n^2	n^2	nk+np	yes	all	very low
Insertion sort	n^2	n^2	nk+np	yes	all	very low
Counting sort	n	n	nk+np	yes	all	very low
Gnome	n	n^2	nk+np	yes	all	very low
Cokatail	n	n^2	nk+np	yes	all	very low
Shell	$n(\log n)^2$	n	nk+np	no	all	low
Merge sort	n log n	n log n	nk+np+stack	yes	all	medium
Quick sort	n log n	n^2	nk+np+stack	no	all	high

2.13 Kompleksitas Algoritma

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

Terdapat beberapa definisi matematis terkait analisis kompleksitas waktu algoritma yaitu:

T(n) = O(f(n)) (BigO), jika konstanta c dan n_0 maka *T(n)* $\leq cf$(n) ketika $n \geq n_0$.

T(n) = Ω *(g(n)) (Big Omega)*, jika konstanta c dan n_0 maka *T(n)* $\geq cg$(n) ketika $n \geq n_0$.

T(n) = Θ *(h(n)) (Big Theta)*, jika dan hanya jika *T(n) = O(h(n))* dan *T(n)*= Ω*(h(n))*.

T(n) = o*(p(n)) (Little o)*, jika *T(n) = O(p(n))* dan *T(n)* $\neq$ Θ*(p(n))*. (Weiss, 2007)

Misalkan, T(n) = 5n+4 dan g(n)= n. Sekarang, dicari hubungan antara keduanya, apakah *Big Oh*, *Big Omega*, *Big Theta*, atau *Little Oh*. Nilai c yang dipakai adalah 9.

```
5n+4    ... c.n
= 5n+4  ... 9n
dimana,
n = 1 -->9  ... 9
n = 2 -->14 ... 18
n = 3 -->19 ... 27
n = 4 -->24 ... 36
n = 5 -->29 ... 45
```

Dari hasil perhitungan di atas, maka dapat dilihat bahwa, setelah kedua persamaan mencapai hasil yang sama, pertumbuhan nilai dari persamaan pertama tersebut lebih kecil dari persamaan kedua. Dengan demikian dapat disimpulkan bahwa 5n+4 = O(n) .

Notasi *O* memiliki beberapa bentuk (Azizah, 2013):

1. *O*(1), merupakan algoritma konstan yang artinya *running time* algoritma tersebut tetap, tidak bergantung pada n.

2. *O*(n), disebut algoritma linier yang artinya bila n menjadi 2n maka *running time* algoritma tersebut akan menjadi dua kali semula

3. $O(n^2)$, disebut algoritma kuadratik yang biasanya hanya digunakan untuk kasus dengan n yang berukuran kecil. Sebab, bila n dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi empat kali semula.

4. $O(n^3)$, disebut algoritma kubik dimana bila n dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi delapan kali semula.

5. $O(2^n)$, disebut algoritma eksponensial dimana bila n dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi kuadrat kali semula

6. *O*(log n), disebut algoritma logaritmik dimana laju pertumbuhan waktu lebih lambat dari pada pertumbuhan n. Algoritma yang termasuk algoritma logaritmik adalah algoritma yang memecahkan persoalan besar dengan mentransformasikannya menjadi beberapa persoalan yang lebih kecil dengan ukuran sama. Basis algoritma tidak terlalu penting, sebab bila misalkan n dinaikkan menjadi dua kali semula, log n meningkat sebesar jumlah tetapan.

7. *O*(n log n), terdapat pada algoritma yang membagi persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan setiap persoalan secara independen, kemudian menggabungkan solusi masing-masing persoalan.

8. *O*(n!), diebut algoritma faktorial dimana algoritma jenis ini akan memproses setiap masukan dan menghubungkannya dengan n-1 masukan lainnya. Bila n menjadi dua kali semula, maka *running time* algoritma akan menjadi faktorial dari 2n.