SKRIPSI

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

Nicholas Aditya Halim

NPM: 2017730018

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»

Nicholas Aditya Halim

NPM: 2017730018

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

Nicholas Aditya Halim

NPM: 2017730018

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, Nugroho, M.Comp.	**«pembimbing pendamping/2»**

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	**«penguji 2»**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Nicholas Aditya Halim
NPM: 2017730018

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

SharIF Judge (dengan IF kapital) adalah modifikasi dari aplikasi Sharif Judge buatan Mohammad Javad Naderi yang berfungsi untuk menilai kode program yang diunggah secara otomatis berdasarkan kunci jawaban yang disediakan. SharIF Judge digunakan pada beberapa kuliah di Informatika Unpar untuk mempermudah proses pengumpulan dan penilaian kode program, terutama saat ujian.

Dengan adanya situasi pandemi, seluruh kegiatan kuliah wajib dilaksanakan secara daring. Hal ini menyebabkan berbagai kesulitan, terutama dalam pelaksanaan ujian. Saat ujian sedang berlangsung, umumnya terdapat pengawas yang mengawasi mahasiswa secara fisik untuk mencegah kecurangan. Namun, pengawasan saat ujian menjadi sangat sulit untuk dilakukan saat kuliah dilaksanakan secara daring. Diperlukan sebuah cara untuk merekam tindakan-tindakan mahasiswa selama ujian daring berlangsung.

Maka, pada skripsi ini akan diimplementasikan editor kode pada SharIF Judge, yang sudah memiliki kemampuan untuk mengompilasi dan menjalankan kode. Dengan demikian, SharIF Judge dapat menjadi sebuah *Integrated Development Environment* yang mampu memfasilitasi seluruh proses pembuatan kode serta merekamnya.

Integrated Development Environment (IDE) adalah sebuah aplikasi yang menyediakan fasilitas untuk pembangunan perangkat lunak. Sebuah IDE memiliki kemampuan untuk mengedit, mengompilasi, dan menjalankan kode program. Pada umumnya, mahasiswa menggunakan aplikasi IDE seperti Netbeans untuk membuat kode program yang kemudian diunggah ke SharIF Judge untuk dinilai.

Dengan mengimplementasikan IDE berbasis web pada SharIF Judge, pengawasan terhadap mahasiswa saat ujian dapat dipermudah dengan merekam ketikan pada editor kode dan mendeteksi bila mahasiswa sedang melihat aplikasi selain SharIF judge.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada skripsi ini adalah sebagai berikut:

- Bagaimana mengimplementasikan *Integrated Development Environment* sehingga mahasiswa dapat mengetik dan menjalankan kode dalam SharIF Judge?

1.3 Tujuan

Tujuan yang ingin dicapai skripsi ini adalah sebagai berikut:

- Mengimplementasikan *Integrated Development Environment* sehingga mahasiswa dapat mengetik dan menjalankan kode dalam SharIF Judge.

1.4 Batasan Masalah

Batasan masalah pada skripsi ini adalah sebagai berikut:

1. Perangkat lunak diuji pada mata kuliah Dasar-dasar Pemrograman.

1.5 Metodologi

Metodologi pengerjaan skripsi ini adalah sebagai berikut:

1. Melakukan studi mengenai komponen yang diperlukan untuk membuat IDE berbasis web.
2. Mempelajari struktur SharIF Judge.
3. Merancang IDE berbasis web untuk SharIF Judge.
4. Mengimplementasikan IDE pada SharIF Judge.
5. Melakukan pengujian dan eksperimen.
6. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika pembahasan skripsi ini adalah sebagai berikut:

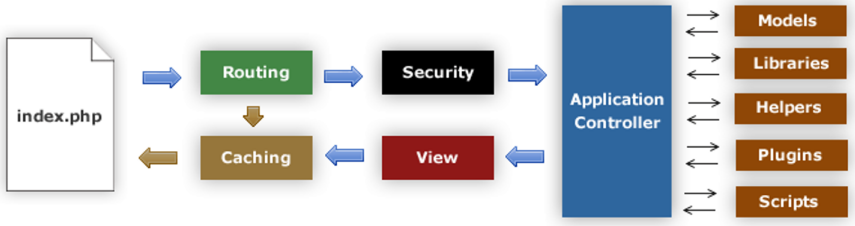
- Bab 1 Pendahuluan membahas mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
- Bab 2 Landasan Teori

BAB 2

LANDASAN TEORI

2.1 CodeIgniter

CodeIgniter adalah sebuah *framework* untuk membangun situs web menggunakan PHP. Tujuan utamanya adalah untuk mempercepat pembuatan proyek dengan menyediakan *library* yang lengkap untuk fungsi-fungsi yang umum digunakan, serta antarmuka yang sederhana dan struktur yang logis untuk mengakses *library* tersebut.



Gambar 2.1: *Flow Chart* CodeIgniter

Gambar 2.1 mengilustrasikan bagaimana data mengalir pada sistem CodeIgniter.

1. *File* index.php berfungsi sebagai *front controller*, menginisialisasi *resource* utama untuk menjalankan CodeIgniter.
2. Router meneliti *request* HTTP dan menentukan apa yang harus dilakukan.
3. Jika terdapat *file cache*, maka langsung dikirimkan ke *browser*.
4. Sebelum *controller* dimuat, seluruh *request* HTTP dan data dari user disaring terlebih dahulu untuk keamanan.
5. *Controller* memuat *model*, *library* utama, dan *resource* lainnya yang diperlukan.
6. *View* akhir lalu dikirim ke browser untuk dilihat. *Cache* akan dibuat terlebih dahulu bila diaktifkan.

2.1.1 Model-View-Controller

CodeIgniter menggunakan pola arsitektur MVC (*Model-View-Controller*) sebagai dasarnya. MVC memisahkan proses logika aplikasi dari presentasi. Dengan demikian, halaman web dapat memuat sedikit *script* karena presentasinya terpisah dari *scripting* PHP.

Model

Model merepresentasikan struktur data. Biasanya *model* memiliki fungsi-fungsi yang membantu dalam mengambil, memasukkan, dan memperbarui informasi pada *database*. Pada CodeIgniter, *model* adalah sebuah kelas yang mengekstensi `CI_Model` dan terletak di direktori `application/models/`.

Kode 2.1: Contoh *model*

```
class Blog_model extends CI_Model {

        public $title;
        public $content;
        public $date;

        public function get_last_ten_entries()
        {
                $query = $this->db->get('entries', 10);
                return $query->result();
        }

        public function insert_entry()
        {
                $this->title    = $_POST['title']; // please read the below note
                $this->content  = $_POST['content'];
                $this->date     = time();

                $this->db->insert('entries', $this);
        }

        public function update_entry()
        {
                $this->title    = $_POST['title'];
                $this->content  = $_POST['content'];
                $this->date     = time();

                $this->db->update('entries', $this, array('id' => $_POST['id']));
        }

}
```

Kode 2.1 merupakan contoh sebuah kelas *model* pada CodeIgniter. Kelas tersebut mengekstensi `CI_Model` dan memiliki fungsi untuk mengambil, memasukkan, dan memperbarui *database*.

View

View adalah informasi yang ditampilkan kepada pengguna. Pada CodeIgniter, *view* merupakan sebuah halaman web atau sebagian dari halaman web yang terletak di direktori `application/view/`.

Kode 2.2: Contoh *view*

```
<html>
<head>
        <title>My Blog</title>
</head>
<body>
        <h1>Welcome to my Blog!</h1>
</body>
</html>
```

Kode 2.2 merupakan contoh sebuah *view*. *View* pada CodeIgniter harus dipanggil melalui *Controller* dan tidak pernah dipanggil secara langsung.

Controller

Controller adalah perantara dari *model* dan *view*, serta *resource* lainnya yang diperlukan untuk memproses *request* HTTP dan menghasilkan sebuah halaman web. Pada CodeIgniter, *controller* adalah sebuah kelas yang mengekstensi `CI_Controller` dan terletak di direktori `application/controllers/`.

Kode 2.3: Contoh *controller*

```
<?php
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }

        public function comments()
        {
                echo 'Look at this!';
        }
}
```

Kode 2.1 merupakan contoh sebuah kelas *controller* pada CodeIgniter. Kelas tersebut mengekstensi `CI_Controller` dan memiliki fungsi `index()` dan `comments()`. Fungsi `index()` akan dipanggil secara otomatis jika tidak ada fungsi lain yang dipanggil.

Kode 2.4: Contoh memuat *model* dan menampilkan *view*

```php
<?php
class Blog_controller extends CI_Controller {

        public function blog()
        {
                $this->load->model('blog');

                $data['query'] = $this->blog->get_last_ten_entries();

                $this->load->view('blog', $data);
        }
}
}
```

Pada CodeIgniter, *model* dan *view* hanya dapat dimuat melalui controller. Pada contoh kode 2.4, fungsi `blog()` pada *controller* memuat *model* untuk mengambil data dari *database*, lalu menampilkan *view* yang memuat data tersebut.

2.1.2 URL CodeIgniter

URL pada CodeIgniter menggunakan *segment-based approach* yang dirancang untuk lebih mudah dibaca oleh *search engine* dan manusia. Berikut ini adalah contoh sebuah URL pada CodeIgniter:

`example.com/class/function/ID`

- Bagian pertama, `class` merepresentasikan kelas *controller* yang akan dipanggil.
- Bagian kedua, `function` merepresentasikan fungsi yang akan dipanggil.
- Bagian ketiga dan seterusnya, `ID` merepresentasikan variabel yang akan digunakan.

2.2 Twig

Twig adalah sebuah *template engine* untuk PHP. Sebuah *template* Twig memuat *variable* atau *expression* yang nantinya akan diubah menjadi *value* saat template dievaluasi, serta *tag* yang mengontrol logika template.

Kode 2.5: Contoh template Twig

```twig
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
        </ul>

        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

Kode 2.5 merupakan contoh sebuah template Twig. Terdapat dua jenis *delimiter*, yaitu `{% ... %}` dan `{{ ... }}`. *Delimiter* `{% ... %}` digunakan untuk menjalankan *statement* seperti `for` dan `if`, sementara *delimiter* `{{ ... }}` digunakan untuk menampilkan nilai dari *variable* atau *expression*.

2.3 PDF.js

PDF.js adalah sebuah library JavaScript yang berfungsi untuk menampilkan *file* Portable Document Format (PDF) menggunakan HTML5 *Canvas*. PDF.js terdiri dari 3 *layer*:

- ***Core*** merupakan bagian dimana proses *parse* dan *interpret* dilakukan terhadap *binary* PDF.
- ***Display*** mengambil *layer core* sebagai API yang lebih mudah digunakan untuk menampilkan PDF dan mengambil informasi lainnya dari sebuah dokumen.
- ***Viewer*** membangun *layer display* sebagai halaman website dengan *user interface* yang dapat ditampilkan di browser.

Salah satu cara untuk menampilkan *file* PDF menggunakan PDF.js adalah dengan *embed* layer *viewer* yang sudah tersedia melalui `web/viewer.js` pada sebuah `iframe`. Contoh kode untuk *embed* PDF.js untuk menampilkan file `sample.pdf` dapat dilihat pada kode 2.6.

Kode 2.6: Contoh kode untuk menggunakan PDF.js

```
<!DOCTYPE html>
<html>
    <iframe src="/web/viewer.html?file=sample.pdf"></iframe>
</html>
```

2.4 Ace

Ace adalah sebuah library JavaScript yang berfungsi sebagai *code editor*. Ace memiliki fitur-fitur yang dapat ditemukan di *code editor* pada umumnya. Kode 2.7 merupakan contoh kode untuk menempatkan editor Ace pada `div` dengan id `editor`. Terdapat berbagai konfigurasi pada Ace editor, pada contoh ini digunakan tema *monokai* dan mode *syntax highlighting* untuk JavaScript.

Kode 2.7: Contoh kode untuk menggunakan Ace

```
<!DOCTYPE html>
<html>
<head>
<title>ACE in Action</title>
</head>
<body>

<div id="editor">
function foo(items) {
    var x = "All this is syntax highlighted";
    return x;
}
</div>

<script src="/ace-builds/src-noconflict/ace.js" type="text/javascript" charset="utf-8"></script>
<script>
    var editor = ace.edit("editor");
    editor.setTheme("ace/theme/monokai");
    editor.session.setMode("ace/mode/javascript");
</script>
</body>
</html>
```

Berikut ini beberapa fungsi Ace yang digunakan:

- `setTheme()`
  Mengubah tema editor kode.
- `setMode()`
  Mengubah mode *syntax highlighting* untuk bahasa pemrograman.
- `setValue()`
  Mengubah isi editor kode.
- `getValue()`
  Mengambil isi editor kode.
- `setReadOnly()`
  Mengatur editor kode menjadi *read only*.

BAB 3

ANALISIS

3.1 Analisis Sistem Kini

SharIF Judge menggunakan *framework* CodeIgniter. Seperti yang dibahas pada bagian 2.1.1, *framework* CodeIgniter menerapkan pola arsitektur MVC, dengan komponen-komponen *model*, *view*, dan *controller* yang terdapat pada `\application`.

3.1.1 Model

Berikut ini adalah seluruh *model* pada SharIF Judge yang terletak pada `\application\models`:

- `Assignment_model`
  Model untuk menangani database *assignments*. Fungsi yang dimiliki:
  - `add_assignment($id, $edit = FALSE)`
    Menambah atau mengubah sebuah *assignment*.
  - `delete_assignment($assignment_id)`
    Menghapus sebuah *assignment*.
  - `all_assignments()`
    Mengambil seluruh *assignment*.
  - `new_assignment_id()`
    Menentukan *integer* terkecil yang dapat digunakan sebagai id *assignment* baru.
  - `all_problems($assignment_id)`
    Mengambil seluruh *problem* dari *assignment*.
  - `problem_info($assignment_id, $problem_id)`
    Mengambil sebuah *problem*.
  - `assignment_info($assignment_id)`
    Mengambil sebuah *assignment*.
  - `is_participant($participants, $username)`
    Mengembalikan `TRUE` jika `$username` terdapat dalam `$participants`.
  - `increase_total_submits($assignment_id)`
    Meningkatkan jumlah total *submit* sebuah *assignment* sebanyak satu.
  - `set_moss_time($assignment_id)`
    Mengubah *"Moss Update Time"* untuk sebuah assignment.
  - `get_moss_time($assignment_id)`
    Mengambil *"Moss Update Time"* untuk sebuah assignment.
  - `save_problem_description($assignment_id, $problem_id, $text, $type)`
    Menambah atau mengubah deskripsi sebuah *problem*.
  - `_update_coefficients($assignment_id, $extra_time, $finish_time, $new_late_rule)`
    Mengubah koefisien seluruh *submission* pada sebuah *assignment*. Fungsi ini dipanggil oleh `add_assignment($id, TRUE)`.
- `Hof_model`
  Model untuk menangani informasi *hall of fame*.

- `Logs_model`
  Model untuk menangani database *logins*.
- `Notifications_model`
  Model untuk menangani database *notifications*.
- `Queue_model`
  Model untuk menangani database *queue*.
- `Scoreboard_model`
  Model untuk menangani database *scoreboard*.
- `Settings_model`
  Model untuk menangani database *settings*.
- `Submit_model`
  Model untuk menangani database *submissions*.
- `User`
  Model untuk menangani informasi preferensi setiap *user*.
- `User_model`
  Model untuk menangani database *users*.

3.1.2 View

View pada SharIF Judge yang terletak pada `\application\views` terbagi menjadi beberapa kategori:

- `errors`
  Menyimpan tampilan halaman error.
- `pages`
  Menyimpan tampilan utama halaman.
- `templates`
  Menyimpan komponen-komponen dasar halaman.

3.1.3 Controller

Berikut ini adalah seluruh *controller* pada SharIF Judge yang terletak pada `\application\controllers`:

- `Assignments`
  Controller untuk menangani *assignments*. Fungsi yang dimiliki:
  - `select()`
    Memilih *assignment* yang sedang ditampilkan.

BAB 4

TEMPLATE

4.1 Template Skripsi FTIS UNPAR

Akan dipaparkan bagaimana menggunakan template ini, termasuk petunjuk singkat membuat referensi, gambar dan tabel. Juga hal-hal lain yang belum terpikir sampai saat ini.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

4.1.1 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh LaTeX, perhatikan contoh di file bab2.tex untuk melihat bagaimana cara memaksa tabel ditempatkan sesuai keinginan kita.

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel 4.1 berikut ini:

Tabel 4.1: Tabel contoh

	v_{start}	$\mathcal{S}_1$	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel 4.2 dan Tabel 4.3 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 4.2: Tabel bewarna(1)

	v_{start}	$\mathcal{S}_2$	$\mathcal{S}_1$	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 4.3: Tabel bewarna(2)

	v_{start}	$\mathcal{S}_1$	$\mathcal{S}_2$	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

4.1.2 Kutipan

Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

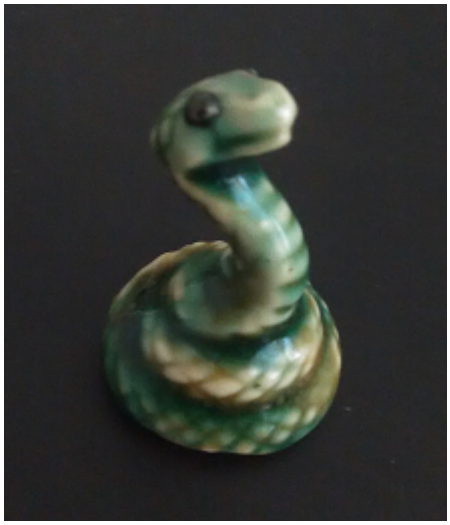
- Buku: [1]
- Bab dalam buku: [2]
- Artikel dari Jurnal: [3]
- Artikel dari prosiding seminar/konferensi: [4]
- Skripsi/Thesis/Disertasi: [5] [6] [7]
- Technical/Scientific Report: [8]
- RFC (Request For Comments): [9]
- Technical Documentation/Technical Manual: [10] [11] [12]
- Paten: [13]
- Tidak dipublikasikan: [14] [15]
- Laman web: [16]
- Lain-lain: [17]

4.1.3 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)
- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar 4.1), tipe JPG (Gambar 4.2) dan tipe PDF (Gambar 4.3)
- Besarnya gambar dapat diatur dengan *field* SCALE.
- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini.
  Jika tidak menggunakan *placement specifier*, LaTeX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
  - Gambar 4.1 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
  - Gambar 4.2 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
  - Gambar 4.3 ditempatkan menggunakan *specifier* **h**.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis,



Gambar 4.1: Gambar *Serpentes* dalam format png

rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Gambar 4.2: Ular kecil

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Gambar 4.3: *Serpentes* jantan

4.1.4 Kode Program

Kode program dalam bahasa tertentu seringkali harus ditulis di dalam bab, bukan hanya dilampirkan di bagian Lampiran. Kode 4.1 menampilkan penggunaan karakter-karakter yang umum digunakan dalam sebuah program yang ditulis dengan bahasa C.

Kode 4.1: Kode untuk menampilkan karakter-karakter aneh

```
// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LaTeX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

4.1.5 Notasi

Simbol-simbol (matematika) yang sering digunakan sepanjang penulisan skripsi, dapat dimasukkan ke dalam "Daftar Notasi". Daftar ini ada di halaman depan sebelum Bab 1. Cara memasukkan sebuah simbol ke dalam Daftar Notasi adalah menggunakan perintah `\nomenclature`. Contoh:

```
\nomenclature[]{$A$}{luas kandang ular}
```

Argumen opsional digunakan untuk mengurutkan notasi. Silahkan lihat sendiri dokumentasi package `nomencl`

DAFTAR REFERENSI

[1] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.

[2] van Kreveld, M. J. (2004) Geographic information systems. Bagian dari Goodman, J. E. dan O'Rourke, J. (ed.), *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, Boca Raton.

[3] Buchin, K., Buchin, M., van Kreveld, M. J., Löffler, M., Silveira, R. I., Wenk, C., dan Wiratma, L. (2013) Median trajectories. *Algorithmica*, **66**, 595–614.

[4] van Kreveld, M. J. dan Wiratma, L. (2011) Median trajectories using well-visited regions and shortest paths. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, USA, 1-4 November, pp. 241–250. ACM, New York.

[5] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.

[6] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.

[7] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.

[8] van kreveld, M., van Lankveld, T., dan Veltkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.

[9] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, http://www.rfc-editor.org.

[10] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing*. International Telecommunications Union. Geneva, Switzerland.

[11] Version 9.0.0 (2016) *The Unicode Standard*. The Unicode Consortium. Mountain View, USA.

[12] Version 7.0 Nougat (2016) *Android API Reference Manual*. Google dan Open Handset Alliance. Mountain View, USA.

[13] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text massage communication between a server and a secure element*. Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.

[14] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.

[15] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. http://tinyurl.com/lionov. 30 September 2016.

[16] Erickson, J. (2003) CG models of computation? http://www.computational-geometry.org/mailing-lists/compgeom-announce/2003-December/000852.html. 30 September 2016.

[17] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

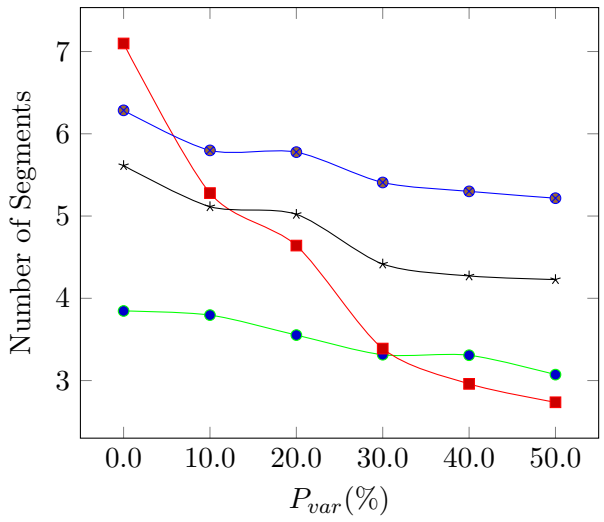
Kode A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

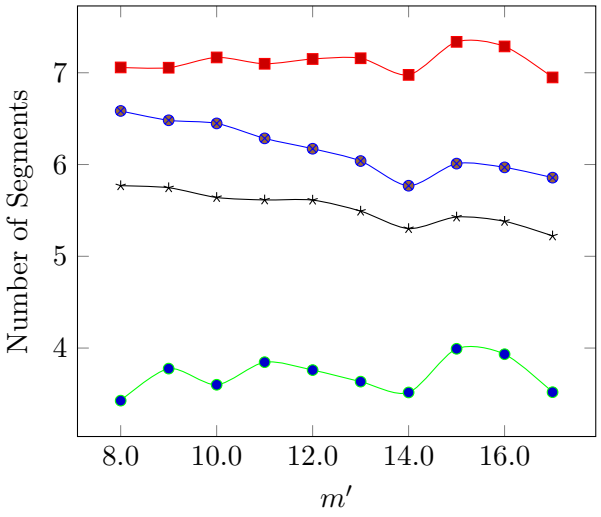
LAMPIRAN B

HASIL EKSPERIMEN

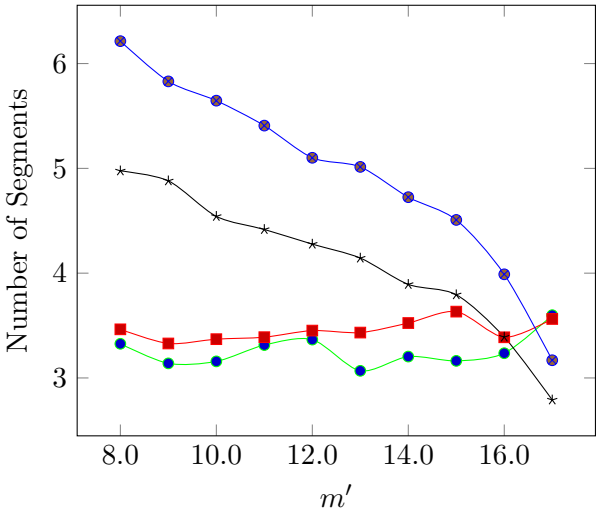
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



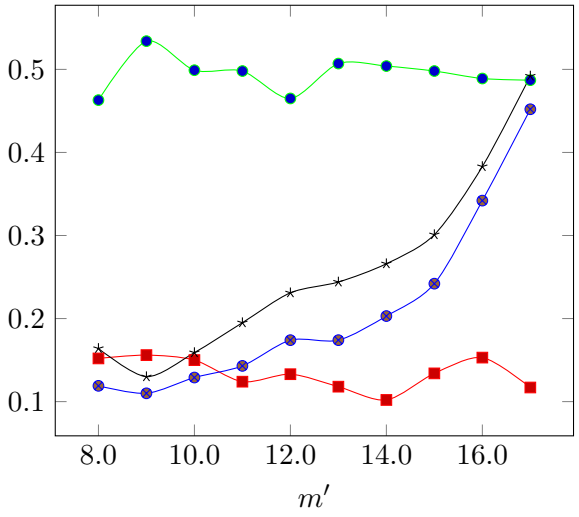
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4