

SKRIPSI

PERKAKAS COMMAND LINE KIRI



Alfred Aprianto Liaunardi

NPM: 6181801014

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022**

DAFTAR ISI

DAFTAR ISI	iii
DAFTAR GAMBAR	v
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	3
2 LANDASAN TEORI	5
2.1 <i>Command Line</i>	5
2.1.1 <i>Command Line Interface</i> dan <i>Graphical User Interface</i>	5
2.1.2 <i>Command Line</i> di Linux	6
2.1.3 <i>Command Line</i> di Windows	7
2.2 KIRI	8
2.2.1 Tampilan	9
2.2.2 API	9
2.3 Skripsi	12
2.4 L ^A T _E X	12
2.5 Template Skripsi FTIS UNPAR	12
2.5.1 Tabel	13
2.5.2 Kutipan	13
2.5.3 Gambar	14
2.5.4 Kode Program	16
2.5.5 Notasi	17
DAFTAR REFERENSI	19
A KODE PROGRAM	21
B HASIL EKSPERIMEN	23

DAFTAR GAMBAR

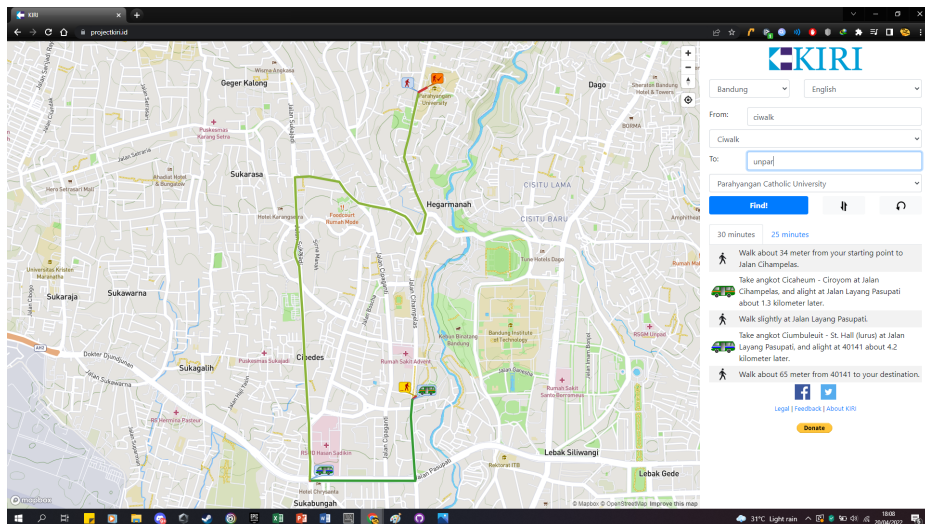
1.1	Tampilan halaman web KIRI	1
2.1	Dua jenis tampilan perangkat lunak	5
2.2	Baris <i>shell prompt</i> terminal di sistem operasi Linux.	6
2.3	Tampang kedua antarmuka <i>command line</i> bawaan di sistem operasi Windows.	8
2.4	Tampilan awal halaman web KIRI	9
2.5	Tampilan akhir halaman web KIRI	10
2.6	Gambar <i>Serpentes</i> dalam format png	15
2.7	Ular kecil	15
2.8	<i>Serpentes</i> betina	16
B.1	Hasil 1	23
B.2	Hasil 2	23
B.3	Hasil 3	23
B.4	Hasil 4	23

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Project KIRI¹ (akan disingkat sebagai KIRI dalam dokumen ini) adalah sebuah perangkat lunak berbasis web yang dibuat untuk membantu mengurangi efek dari kemacetan. KIRI mengurangi dampak kemacetan dengan membantu penggunanya, baik masyarakat maupun turis, dalam menggunakan salah satu sarana transportasi umum yang ada di Indonesia, yaitu angkutan kota (angkot). Cara KIRI mempermudah penggunaan angkot adalah dengan menunjukkan rute yang akan ditempuh, beserta langkah-langkah yang harus dilakukan oleh pengguna yang ingin berpergian dari satu titik ke titik lain, mulai dari seberapa jauh pengguna harus berjalan untuk menaiki angkot yang bersangkutan, di mana pengguna harus naik atau turun, seberapa jauh lagi pengguna harus berjalan sampai ke titik tujuan, dan seberapa lama estimasi waktu perjalanan yang akan ditempuh. Untuk kebutuhan pembuatan perangkat lunak yang memanfaatkan fitur dari KIRI, tersedia juga REST API KIRI yang dapat digunakan secara praktis. Adapun tampilan dari halaman web ini dapat dilihat di gambar 1.1.



Gambar 1.1: Tampilan halaman web KIRI, yang menunjukkan rute dari Cihampelas Walk ke Universitas Katolik Parahyangan.

Sementara itu, dalam komputer, salah satu dari sekian banyak tipe perangkat lunak adalah *command line*. *Command line* adalah perangkat lunak paling sederhana, yang sudah ada sejak

¹<https://projectkiri.id>

pertama kali komputer diciptakan. Perangkat lunak selalu memiliki tampilan berupa *command line interface* (CLI), yang tidak memiliki tampilan apapun selain sebuah kotak yang memuat teks berupa perintah-perintah tertentu, baik perintah yang meminta masukan dari user untuk dilakukan oleh komputer, maupun perintah yang menampilkan keluaran dari komputer, tanpa ada tambahan gambar grafis apapun, seperti pada perangkat lunak dengan tampilan *graphical user interface* (GUI). Singkatnya, tipe perangkat lunak ini bukan merupakan tipe yang paling indah untuk dilihat oleh para pengguna, tetapi jika digunakan dengan tepat, maka jenis perangkat lunak ini bisa menyuruh komputer untuk melakukan banyak sekali perintah-perintah dengan sangat cepat dan sangat efektif.

Pada skripsi ini akan dibuat sebuah perangkat lunak berupa perkakas *command line* (*command line tool*) yang dapat menjalankan fungsi-fungsi API dari KIRI. Perangkat lunak ini, seperti jenisnya, akan dibuat murni sebagai perkakas yang dijalankan dari *command line* (terminal, cmd, PowerShell, dll.), dan tampilan akhir dari perangkat lunak akan berupa *command line interface* tanpa tambahan *graphical user interface*. Keseluruhan dari perangkat lunak ini akan dibangun dalam bahasa C.

1.2 Rumusan Masalah

1. Bagaimana membangun perkakas *command line* yang dapat mengimplementasikan fitur-fitur API KIRI dalam bahasa C?
2. Bagaimana integrasi perkakas *command line* KIRI dapat dilakukan dengan perkakas-perkakas *command line* lainnya di Linux?

1.3 Tujuan

Batasan masalah dalam skripsi ini adalah sebagai berikut:

1. Membangun perkakas *command line* yang dapat mengimplementasikan fitur-fitur API KIRI dalam bahasa C.
2. Melakukan integrasi perkakas *command line* KIRI dengan perkakas-perkakas *command line* lainnya di Linux.

1.4 Batasan Masalah

Batasan masalah dalam skripsi ini adalah sebagai berikut:

1. Perangkat lunak dibuat murni dalam bentuk CLI, tanpa tambahan GUI.
2. Perangkat lunak yang dibuat tidak menyelesaikan batasan (lokasi tidak terdeteksi, rute tidak berhasil ditemukan, dsb.) yang sudah sejak awal terdapat dalam KIRI.

1.5 Metodologi

Metodologi yang akan diikuti dalam skripsi ini adalah sebagai berikut:

1. Melakukan studi dan eksplorasi terhadap fungsi-fungsi yang dimiliki perangkat lunak KIRI serta cara implementasi API KIRI.
2. Melakukan analisis dan desain perangkat lunak yang akan dibangun.

3. Melakukan studi dan eksplorasi terhadap seluruh kemungkinan *library-library* yang memenuhi spesifikasi dalam pembuatan perangkat lunak, berdasarkan analisis dan desain yang telah dilakukan sebelumnya.
4. Melakukan analisis kebutuhan fitur-fitur perangkat lunak dan melakukan eksplorasi *library* yang dapat digunakan dan memenuhi spesifikasi dalam pembuatan perangkat lunak.
5. Membangun perangkat lunak berdasarkan rancangan yang sudah dibuat, dengan mengimplementasikan seluruh modul dan *library* yang telah ditentukan di tahap sebelumnya dalam bahasa C.
6. Melakukan pengujian fungsional, perbaikan *bug*, serta rekomendasi perbaikan berdasarkan pengujian yang sudah dilakukan.
7. Menyelesaikan pembuatan dokumen-dokumen yang berkaitan, seperti dokumen skripsi dan dokumentasi perangkat lunak.

1.6 Sistematika Pembahasan

Setiap bab dalam skripsi ini memiliki sistematika pembahasan dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.

2. Bab 2: Dasar Teori

Bab ini berisi pembahasan-pembahasan teoritis mengenai aspek-aspek yang akan dirujuk di dalam skripsi ini, seperti *command line*, bahasa C, dan juga KIRI.

3. Bab 3: Analisis dan Perancangan

Bab ini berisi pembahasan mengenai rancangan perangkat lunak serta seluruh analisis yang dilakukan terhadap kebutuhan fitur perangkat lunak.

4. Bab 4: Implementasi dan Pengujian

Bab ini berisi pembahasan mengenai pembuatan perangkat lunak, implementasi seluruh modul-modul yang telah ditentukan di bab 3, serta pengujian fitur-fitur dari perangkat lunak.

5. Bab 5: Kesimpulan dan Saran

Bab ini berisi kesimpulan hasil pembuatan perangkat lunak dan saran-saran terhadap hasil perangkat lunak yang diberikan selama pengerjaan skripsi.

BAB 2

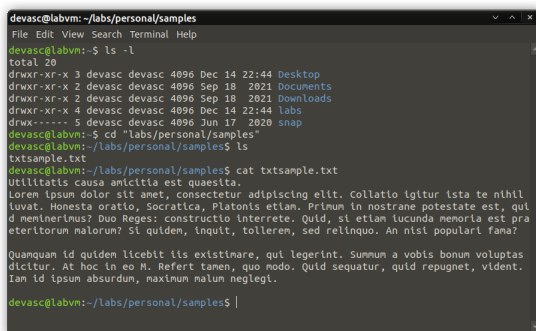
LANDASAN TEORI

2.1 *Command Line*

Command line (atau *command line interface*) dapat diartikan sebagai tampilan antarmuka/*interface* yang memproses perintah dari pengguna dan meneruskannya langsung ke sistem operasi untuk dijalankan.[1] Seluruh sistem operasi komputer yang ada memiliki sebuah *command line interface* dalam bentuk *shell*, yang dapat digunakan oleh penggunanya untuk langsung mengakses fungsi atau servis yang disediakan oleh sistem operasi.[2]

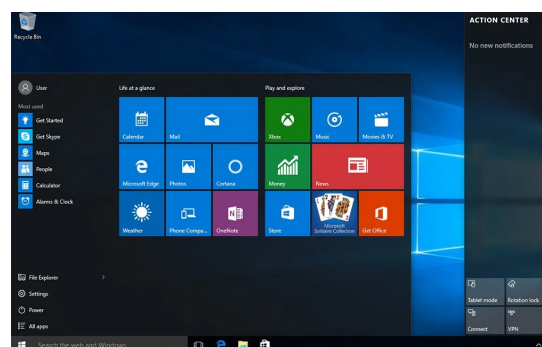
2.1.1 *Command Line Interface dan Graphical User Interface*

Ada beberapa dari tipe antarmuka yang masih banyak digunakan di zaman sekarang, tetapi dua tipe yang paling banyak muncul adalah *command line interface* dan *graphical user interface*. Perangkat lunak berbasis *command line* sendiri bisa memiliki berbagai macam tampilan, tetapi semuanya selalu mengikuti satu bentuk antarmuka umum. Bentuk yang dimaksud adalah sebuah area/*window* yang memuat teks berupa perintah-perintah dari user untuk dilakukan oleh komputer, beserta keluarannya yang juga berupa teks, seperti dapat dilihat pada gambar 2.1a. Jenis perangkat lunak seperti ini disebut memiliki antarmuka jenis *command line interface* (CLI). Adapun dekorasi visual yang dimiliki oleh jenis tampilan ini hanya berupa warna pada teks-teks yang ada, tanpa tambahan gambar apapun. Jika perangkat lunak tersebut memiliki dekorasi dan/atau tombol interaktif berupa gambar grafis, seperti pada gambar 2.1b, maka perangkat lunak tersebut dikategorikan sebagai perangkat lunak berbasis *graphical user interface*.



```
devasc@labvm: ~/labs/personal/samples
File Edit View Search Terminal Help
devasc@labvm:~$ ls -l
total 20
drwxr-xr-x 3 devasc devasc 4096 Dec 14 22:44 Desktop
drwxr-xr-x 2 devasc devasc 4096 Sep 18 2021 Documents
drwxr-xr-x 2 devasc devasc 4096 Sep 18 2021 Downloads
drwxr-xr-x 4 devasc devasc 4096 Dec 14 22:44 Labs
drwx----- 5 devasc devasc 4096 Jun 17 2020 snap
devasc@labvm:~$ cd ~/labs/personal/samples
devasc@labvm:~/labs/personal/samples$ ls
txtsample.txt
devasc@labvm:~/labs/personal/samples$ cat txtsample.txt
Utilitatis causa amicitia est quaesita.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Collatio igitur ista te nihil
luvat. Honestas oratio, Socratica, Platonis etiam. Primum in nostram potestatem est, qui
d meminerimus? Duo Reges: constructio interrete. Quid, si etiam lucunda memoria est pra
eteritum malorum? Si quidem, inquit, tollerem, sed relinquo. An nisi populari fano?
Quamquam id quidem licebit illis existinare, qui legerint. Summun a vobis bonum voluptas
dicitur. At hoc in eo M. Refert tamen, quo modo. Quid sequatur, quid repugnet, vident.
Iam id ipsum absurdum, maximum malum neglegi.
```

(a) Antarmuka perangkat lunak berbasis *command line interface*.



(b) Antarmuka perangkat lunak berbasis *graphical user interface*.

Gambar 2.1: Contoh dua jenis antarmuka (*interface*) perangkat lunak.

Selain dari tampilannya sendiri, ada beberapa perbedaan utama lain antara perangkat-perangkat lunak berbasis *command line interface* dengan perangkat lunak berbasis *graphical user interface*. Adapun perbedaan-perbedaan utama dari kedua jenis antarmuka ini adalah sebagai berikut.

- Penggunaan sumber daya sistem untuk menjalankan perangkat lunak berbasis *command line interface* lebih rendah dibandingkan dengan perangkat lunak berbasis *graphical user interface*.
- Bagi pengguna pemula (atau pengguna awam pada umumnya), perangkat lunak berbasis *command line interface* akan lebih sulit digunakan karena tidak adanya bantuan apapun dalam bentuk visual, sehingga satu-satunya cara untuk tahu bagaimana cara menggunakan fitur-fiturnya adalah melalui dokumentasi perangkat lunak yang ada. Karena alasan yang sama pula, perangkat lunak berbasis *command line interface* lebih sulit untuk dibiasakan penggunaannya.
- Automasi perintah yang bersifat berulang-ulang jauh lebih mudah dilakukan pada perangkat lunak berbasis *command line interface*. Hal ini dikarenakan perangkat lunak berbasis *command line interface* tidak hanya lebih mudah untuk dibuat *script*-nya, tetapi juga lebih efisien untuk digunakan ketika ada banyak sekali perintah yang harus dilakukan pada suatu saat tertentu.[2]

2.1.2 *Command Line* di Linux

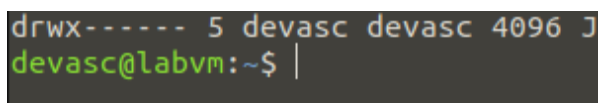
Linux merupakan sebuah sistem operasi yang sangat modular, jadi ada banyak sekali *shell* yang dapat dijalankan dan digunakan di dalamnya. Walaupun begitu, ada satu *shell* yang selalu datang ter-*install* di dalam semua sistem operasi Linux, yaitu "*bash*" (GNU *Bourne Again Shell*).[3]

Tampilan

Ketika terminal di Linux dijalankan, akan keluar kotak dialog, beserta sebuah baris. Baris ini biasanya berisi sebuah teks dengan format sebagai berikut.

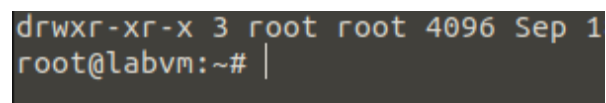
```
<nama pengguna>@<nama perangkat>:<direktori yang sedang diproses>$
```

Tanda dolar di ujung baris ini menandakan bahwa baris tersebut merupakan baris *shell prompt*, yang merupakan waktu di mana terminal sudah siap menerima masukan dari pengguna untuk diproses. Perlu diingat bahwa di posisi tanda dolar ini, terkadang justru terdapat tanda pagar (#). Tanda pagar di akhir baris *shell prompt* menandakan bahwa terminal tersebut dijalankan dengan tingkat akses *superuser*, yang berarti bahwa entah pengguna masuk ke sistem sebagai user *root*, atau terminal memiliki izin tingkat *superuser/administrator*. [1]



```
drwx----- 5 devasc devasc 4096 J
devasc@labvm:~$ |
```

(a) *Shell prompt* terminal dengan tingkat izin normal.



```
drwxr-xr-x 3 root root 4096 Sep 1
root@labvm:~# |
```

(b) *Shell prompt* terminal dengan tingkat izin *superuser*.

Gambar 2.2: Baris *shell prompt* terminal di sistem operasi Linux.

1 Navigasi

2 Sama seperti di Windows, Linux menyimpan file-filenya di sebuah struktur direktori yang bersifat
3 hierarkial. Hal ini berarti bahwa file-file tersebut disimpan dalam direktori-direktori (atau *folder-*
4 *folder*) yang tersusun seperti sebuah pohon. dalam arti bahwa satu *folder* bisa jadi berada di dalam
5 satu *folder* lain, atau berisi beberapa *folder* lainnya.[1]

6 Untuk navigasi, terminal Linux memiliki beberapa perintah utama. Adapun perintah-perintah
7 tersebut adalah sebagai berikut.

- 8 • **pwd** [1]

9 **pwd** merupakan singkatan dari *print working directory*, yang berarti bahwa perintah ini akan
10 mengeluarkan *working directory*, atau direktori tempat terminal sekarang sedang bekerja/ber-
11 jalan, sebagai keluaran dari perintah tersebut. Ketika pengguna pertama kali menjalankan
12 terminal, *working directory*-nya selalu merupakan direktori *home* dari perangkat.

- 13 • **ls** [1]

14 **ls** digunakan untuk menghasilkan keluaran berupa isi dari folder yang dispesifikasi. Biasanya
15 digunakan ketika pengguna sudah memasuki folder yang diinginkan, walaupun dengan perintah
16 ini, pengguna bisa saja mengintip isi dari folder manapun di direktori manapun, dengan
17 mengikutkan direktori yang diinginkan sebagai parameter dari perintah tersebut. Adapun
18 Isi dari folder yang diikutkan sebagai parameter tidak hanya berupa folder lain, tetapi juga
19 seluruh file-file yang ada, walaupun untuk file-file yang disembunyikan (nama file diawali
20 dengan tanda titik), perlu ditambahkan opsi **-a** agar file-file tersebut muncul pula dalam
21 keluarannya.

- 22 • **cd** [1]

23 **cd** adalah perintah yang berfungsi untuk mengganti *working directory* dari terminal. Untuk
24 melakukan hal tersebut, perintah yang perlu dimasukkan adalah sebagai berikut:

25 **cd <direktori yang diinginkan>**

26 Direktori yang diinginkan dapat berupa direktori absolut, atau direktori relatif. Perbedaanannya
27 adalah direktori absolut selalu dimulai dari folder *root*, mengikuti folder-folder apapun yang
28 ada di antara *root* sampai ke folder yang diinginkan.

29 Sedangkan, direktori relatif selalu dimulai dari *working directory*. Untuk penggunaan direktori
30 relatif, diperlukan dua buah notasi spesial, yaitu titik (**.**), yang merepresentasikan *working*
31 *directory* sekarang itu sendiri, dan dua titik (**..**), yang merepresentasikan *parent folder* dari
32 *working directory*.

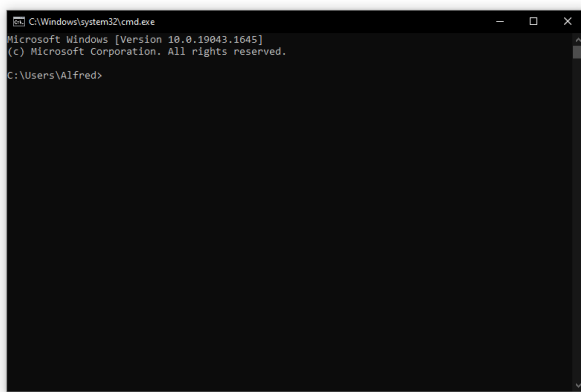
33 2.1.3 Command Line di Windows

34 Cara kerja *command line* di Windows serupa dengan cara kerja *command line* di Linux, dalam
35 arti bahwa untuk bekerja dengan *command line* di Windows, penggunaanya juga akan langsung
36 berinteraksi dengan utilitas yang disediakan oleh sistem operasi. *Command line* di Windows juga
37 dapat digunakan untuk hal-hal yang serupa dengan *command line* di Linux, seperti menulis (dan
38 menjalankan) *script*, menjalankan perintah yang diinginkan pengguna secara otomatis, atau melihat
39 status dari sistem operasi.[2]

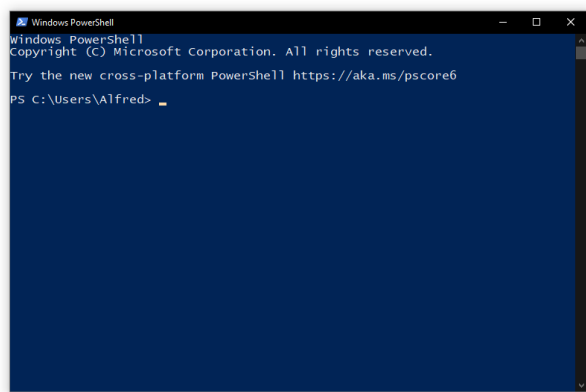
Masih sama dengan Linux, ada banyak sekali command yang bisa digunakan, sehingga susah untuk menghafal seluruh command-command yang ada—termasuk masukan, parameter-parameter yang dibutuhkan, serta keluarannya. Untuk melihat dokumentasi, atau penjelasan detail untuk masukan, keluaran, parameter, serta opsi-opsi dari perintah tertentu, pengguna dapat memasukkan perintah tersebut, diikuti dengan `/?`.^[2]

Tampilan

Di sistem operasi Windows, ada dua jenis antarmuka *command line*, yaitu `cmd` (*Command Prompt*) dan *PowerShell*. Keduanya memiliki tampilan yang kurang lebih sama—hanya saja awalnya `cmd` memiliki latar belakang hitam, sedangkan *PowerShell* memiliki latar belakang biru tua, seperti terlihat di gambar 2.3.



(a) Antarmuka Windows *Command Prompt* (`cmd`)



(b) Antarmuka Windows *PowerShell*

Gambar 2.3: Tampilan kedua antarmuka *command line* bawaan di sistem operasi Windows.

Navigasi

2.2 KIRI

KIRI merupakan sebuah perangkat lunak berbasis web yang berfungsi untuk menyelesaikan (atau setidaknya mengurangi) dampak dari masalah-masalah yang dapat diselesaikan oleh transportasi umum/publik di Indonesia, seperti pemanasan global, kemacetan, atau peningkatan harga bensin. Selain itu, turis mancanegara juga memilih untuk menaiki transportasi umum, karena sarana transportasi tersebut tidak hanya jauh lebih murah, tetapi juga memberikan kesempatan kepada mereka untuk melihat seluk-beluk dari kota-kota yang mereka kunjungi. Walaupun begitu, masih banyak masyarakat lokal sendiri yang segan untuk menaiki transportasi publik, umumnya karena transportasi publik lebih rumit persiapannya dibandingkan dengan transportasi privat, seperti kendaraan pribadi.¹

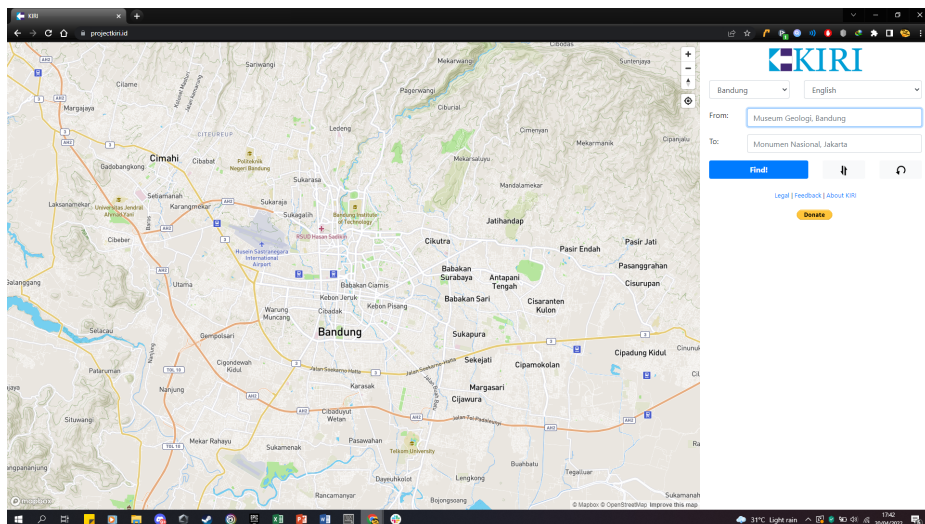
Di halaman web KIRI, pengguna dapat memasukkan input berupa lokasi awal dan lokasi tujuan dan KIRI akan menghasilkan seluruh langkah yang harus ditempuh oleh pengguna untuk sampai ke lokasi tujuan, dengan menggunakan angkot. Keluaran ini sudah meliputi kode angkot mana saja

¹<https://projectkiri.github.io/#about-kiri>

- 1 yang harus dinaiki, dan juga seberapa jauh pengguna harus berjalan kaki untuk sampai ke lokasi
- 2 rute angkot berikutnya.

3 2.2.1 Tampilan

4 Pada saat pertama kali dibuka, hal pertama yang paling mencolok di halaman awal web KIRI adalah
5 sebuah peta besar di sebelah kiri yang dapat diperbesar ataupun diperkecil. Sedangkan, bagian
6 kanan dari halamannya terdiri atas beberapa bagian. Di bagian paling atas terdapat logo KIRI,
7 beserta sepasang menu *dropdown* - yang pertama merupakan pilihan kota tempat pengguna berada
8 (untuk sekarang hanya tersedia pilihan kota Jakarta dan Bandung), dan yang kedua merupakan
9 pilihan bahasa, entah bahasa Indonesia atau Inggris. Di bawahnya merupakan sepasang menu
10 *dropdown* yang merupakan tempat di mana pengguna memasukkan lokasi awal dan tujuan yang
11 akan diproses oleh KIRI. Terakhir, di bawahnya ada sebuah bagian kosong, yang nantinya akan
12 menjadi tempat di mana KIRI akan meletakkan keluaran dari prosesnya. Adapun tampilan awal
13 dari halaman web ini dapat dilihat di gambar 2.4.



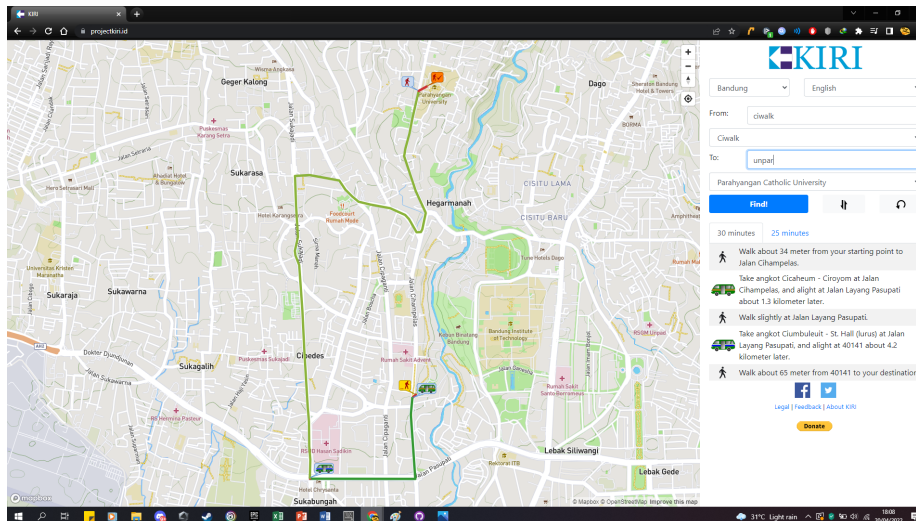
Gambar 2.4: Tampilan awal halaman web KIRI.

14 Ada dua area yang memiliki perbedaan yang signifikan ketika pengguna sudah memasukkan
15 masukan dan menyuruh KIRI untuk memprosesnya. Bagian yang pertama adalah bagian peta, yang
16 setelah pemrosesan masukan, akan memiliki garis-garis berwarna yang menandakan rute angkot
17 maupun tujuan perjalanan kaki yang harus ditempuh oleh pengguna. Bagian kedua adalah bagian
18 keluaran, yang tadinya kosong, sekarang akan berisi langkah-langkah yang harus ditempuh oleh
19 penggunanya untuk pergi dari lokasi awal ke lokasi tujuan. Spesifiknya, perbedaan-perbedaan ini
20 dapat dilihat di gambar 2.5.

21 2.2.2 API

22 KIRI juga memiliki sebuah API yang dapat digunakan untuk keperluan pengembangan perangkat
23 lunak. Seluruh permintaan (*request*) yang dilakukan melalui API KIRI harus dilakukan sebagai
24 permintaan tipe GET ke <https://projectkiri.id/api>, beserta parameter-parameter yang dibutuhkan.

25 Permintaan tersebut harus memiliki parameter-parameter seperti terlihat di bawah ini.



Gambar 2.5: Tampilan halaman web KIRI setelah pemrosesan masukan dari pengguna selesai.

- **version**

Kemungkinan nilai: 2

Parameter ini merupakan tanda bagi API untuk menggunakan protokol versi 2.

- **mode**

Kemungkinan nilai: findroute

Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna. Untuk mode **findroute**, jasa yang akan digunakan adalah jasa pencarian rute dengan angkot.

- **locale**

Kemungkinan nilai: en atau id

Parameter ini mengatur bahasa apa yang akan digunakan dalam keluaran API nantinya—**en** berarti keluaran akan menggunakan bahasa Inggris, dan **id** berarti keluaran akan menggunakan bahasa Indonesia.

- **start**

Kemungkinan nilai: lat, lng; dalam bentuk desimal

Parameter ini merupakan nilai *latitude* dan *longitude* dari titik awal perjalanan pengguna.

- **finish**

Kemungkinan nilai: lat, lng; dalam bentuk desimal

Parameter ini merupakan nilai *latitude* dan *longitude* dari titik akhir/tujuan perjalanan pengguna.

- **presentation**

Kemungkinan nilai: desktop

Parameter ini hanya digunakan untuk fitur *backwards compatibility*.

- **apikey**

Kemungkinan nilai: angka heksadesimal 16-digit

Parameter ini berisi kunci API pribadi yang harus digenerasi terlebih dahulu sebelum API dapat digunakan.

Sedangkan, respon yang diberikan oleh API berupa sebuah objek JSON yang selalu memiliki setidaknya dua variabel, yaitu:

- **status**

Kemungkinan nilai: `ok` atau `error`

Variabel ini manandakan apakah permintaan berhasil diproses atau tidak. Jika permintaan berhasil diproses, variabel ini akan bernilai `ok`, dan jika tidak, variabel ini akan bernilai `error`.

- **message**

Variabel ini bisa berisi dua macam objek. Jika permintaan dari user tidak berhasil diproses, atau dalam kata lain, terjadi sebuah *error*, maka variabel ini akan berisi string yang merupakan pesan *error* serta alasan spesifik mengapa *error* tersebut terjadi. Di lain sisi, jika permintaan dari user berhasil diproses, variabel ini akan mengalami dua perubahan utama. Pertama, nama variabel ini akan berubah menjadi `routingresults`, dan kedua, isi dari variabel ini akan menjadi sebuah *array* JSON yang merupakan respon dari API KIRI berupa keluaran yang akan dilihat oleh pengguna. *Array* JSON ini sendiri terbagi menjadi beberapa variabel lainnya, yang dapat dilihat di daftar di bawah ini.

- **steps**

Tipe: `array`

Variabel ini merepresentasikan satu buah langkah yang harus ditempuh oleh pengguna. Adapun *array* ini sendiri berisi variabel-variabel berikut:

- * Tipe transportasi

Tipe sarana transportasi yang harus dipakai oleh pengguna. Jika pengguna harus berjalan kaki, variabel ini akan berisi `walk`. Jika pengguna harus menaiki angkot, variabel ini akan berisi `angkot`.

- * Kode angkot

Variabel ini menunjukkan angkot mana yang harus dinaiki oleh pengguna di langkah tersebut. Jika penggunaan angkot tidak dimungkinkan pada langkah ini (pengguna harus berjalan kaki), variabel ini akan berisi `walk`.

- * Array *latitude* dan *longitude* lokasi

Array nilai-nilai desimal *latitude* dan *longitude* dari berbagai titik lokasi yang terdapat dalam rute.

- * Deskripsi langkah

Deskripsi langkah yang harus ditempuh, dalam bahasa natural. Bahasa apa yang digunakan untuk deskripsi ini tergantung parameter *locale* yang diatur dalam masukan.

- * URL untuk mendapatkan tiket kendaraan

Tautan untuk mendapatkan tiket angkutan umum, jika diperlukan. Jika transportasi pada langkah tersebut tidak memerlukan tiket, variabel ini akan berisi `null`.

- * URL editor rute

Tautan untuk meng-edit rute, jika situasinya memungkinkan. Jika tidak, variabel ini akan berisi `null`.

- **traveltime**

Tipe: `string`

Variabel ini berisi estimasi jangka waktu yang diperlukan untuk menyelesaikan langkah tersebut.

2.3 Skripsi

Rencananya akan diisi dengan penjelasan umum mengenai buku skripsi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetur eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetur tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

2.4 L^AT_EX

Mengapa menggunakan L^AT_EX untuk buku skripsi dan apa keunggulan/kerugiannya bagi mahasiswa dan pembuat template.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

2.5 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.

2.5.1 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh L^AT_EX, 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 2.1 berikut ini:

Tabel 2.1: 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 2.2 dan Tabel 2.3 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.2: 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 2.3: 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

2.5.2 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: [4]
- Bab dalam buku: [5]
- Artikel dari Jurnal: [6]
- Artikel dari prosiding seminar/konferensi: [7]
- Skripsi/Thesis/Disertasi: [8] [9] [10]

- Technical/Scientific Report: [11]
- RFC (Request For Comments): [12]
- Technical Documentation/Technical Manual: [13] [14] [15]
- Paten: [16]
- Tidak dipublikasikan: [17] [18]
- Laman web: [19]
- Lain-lain: [20]

2.5.3 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen L^AT_EX 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 2.6), tipe JPG (Gambar 2.7) dan tipe PDF (Gambar 2.8)
 - 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*, L^AT_EX 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 2.6 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
 - Gambar 2.7 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisan di dalam file bab2.tex!
 - Gambar 2.8 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, 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, consectetur 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

Gambar 2.6: Gambar *Serpentes* dalam format png

1 torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam
 2 elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl.
 3 Vestibulum sed nisl eu sapien cursus rutrum.

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

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



Gambar 2.7: Ular kecil

16 Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo
 17 lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac
 18 lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper
 19 sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit.

1 Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum
 2 sed, viverra at, consectetur quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in,
 3 suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.
 4 Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros.
 5 Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae
 6 nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac
 7 enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec
 8 vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh
 9 pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna
 10 tincidunt congue.



Gambar 2.8: *Serpentes* jantan

11 2.5.4 Kode Program

12 Kode program dalam bahasa tertentu seringkali harus ditulis di dalam bab, bukan hanya dilampirkan
 13 di bagian Lampiran. Kode 2.1 menampilkan penggunaan karakter-karakter yang umum digunakan
 14 dalam sebuah program yang ditulis dengan bahasa C.

Kode 2.1: Kode untuk menampilkan karakter-karakter aneh

```

15 // This does not make algorithmic sense,
16 // but it shows off significant programming characters.
17
18 #include<stdio.h>
19
20 void myFunction( int input, float* output ) {
21     switch ( array[i] ) {
22         case 1: // This is silly code
23             if ( a >= 0 || b <= 3 && c != x )
24                 *output += 0.005 + 20050;
25             char = 'g';
26             b = 2^n + ~right_size - leftSize * MAX_SIZE;
27             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
28             strcpy(a,"hello_$(?)");
29         }
30     count = ~mask | 0x00FF00AA;
31 }
32
33 // Fonts for Displaying Program Code in LATEX
34 // Adrian P. Robson, nepsweb.co.uk
35 // 8 October 2012

```

2.5.5 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 `nomenc1`

DAFTAR REFERENSI

- [1] Shotts Jr., W. E. (2019) *The Linux Command Line*, 5th internet edition. <https://www.linuxcommand.org/tlcl.php>.
- [2] Mueller, J. P. (2007) *Windows® Administration at the Command Line for Windows Vista™, Windows® 2003, Windows® XP, and Windows® 2000*, 1st edition. Wiley Publishing, Inc., Indiana.
- [3] Neil Matthew, R. S. (2007) *Beginning Linux® Programming*, 4th edition. Wiley Publishing, Inc., Indiana.
- [4] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.
- [9] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.
- [10] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.
- [11] van kreveld, M., van Lankveld, T., dan Velkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.
- [12] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, <http://www.rfc-editor.org>.
- [13] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing*. International Telecommunications Union. Geneva, Switzerland.
- [14] Version 9.0.0 (2016) *The Unicode Standard*. The Unicode Consortium. Mountain View, USA.
- [15] Version 7.0 Nougat (2016) *Android API Reference Manual*. Google dan Open Handset Alliance. Mountain View, USA.

- [16] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text message communication between a server and a secure element*. Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.
- [17] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.
- [18] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. <http://tinyurl.com/lionov>. 30 September 2016.
- [19] Erickson, J. (2003) CG models of computation? <http://www.computational-geometry.org/mailling-lists/compgeom-announce/2003-December/000852.html>. 30 September 2016.
- [20] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Kode A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


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