

SKRIPSI

PERKAKAS COMMAND LINE KIRI



Alfred Aprianto Liaunardi

NPM: 6181801014

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

UNDERGRADUATE THESIS

KIRI COMMAND LINE TOOL



Alfred Aprianto Liaunardi

NPM: 6181801014

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022**

ABSTRAK

Project KIRI (atau KIRI saja) merupakan sebuah perkakas berbasis *web* yang dapat digunakan untuk membantu memudahkan penggunaanya untuk menggunakan angkutan kota (angkot). KIRI merealisasikan hal ini dengan cara menunjukkan kepada penggunaanya langkah-langkah yang harus dilakukan untuk pergi dari suatu lokasi ke lokasi lainnya, angkot-angkot mana yang harus dinaiki dalam rute tersebut, serta di mana pengguna harus menaiki atau turun dari angkot-angkot yang bersangkutan. Selain itu, KIRI juga dapat menunjukkan estimasi waktu dari rute-rute yang ditemukan.

Sementara itu, dalam komputer, salah satu dari sekian banyak tipe perangkat lunak adalah perangkat lunak berbasis *command line*. Perangkat-perangkat lunak ini selalu terdiri atas sebuah kotak (*window*) yang memuat teks berupa perintah-perintah, yang menerima masukan (berupa perintah-perintah juga) langsung dari pengguna dan menjalankannya. Indikator utama dari tipe perangkat lunak ini adalah bahwa perangkat-perangkat lunak jenis ini tidak memiliki tampilan dengan gambar grafis apapun—dalam artian bahwa tampilan perangkat tipe ini hanya berupa teks.

Dalam skripsi ini akan dibuat sebuah perangkat lunak berupa perkakas *command line* yang dapat menjalankan fungsi-fungsi API KIRI. Seperti jenis umumnya, perkakas ini akan dibuat murni sebagai perkakas yang dijalankan dari *command line*, dan tidak akan memiliki tambahan gambar grafis apapun dalam tampilannya. Perkakas ini akan dibuat dalam bahasa C, dengan mengutilisasikan berbagai macam *library*, seperti getopt, cJSON, cURL, dan CMake, serta mengimplementasikan fitur-fitur API KIRI serta fitur-fitur dasar perkakas *command line*, seperti mode bantuan, dan *man page* (untuk Linux). Perkakas ini nantinya akan diuji coba dengan cara menguji satu-satu fungsinya, dan juga dengan menguji kasus-kasus umum untuk integrasinya dengan perkakas-perkakas lainnya yang sudah ada.

Kata-kata kunci: Navigasi, angkot, *Project KIRI*, *command line*, C

ABSTRACT

Project KIRI (or just KIRI) is a web-based tool which could be used to assist its users in utilizing *angkutan kota* (or *angkot*). KIRI does this by showing to its users the steps needed to go from one location to the other, which *angkots* would have to be taken within said route, as well as where they would have to board or get off of these *angkots*. Aside from that, KIRI also has the ability to show the estimated durations of the available routes.

Meanwhile, in computers, one of the many types of softwares is command line softwares. These programs always consist of a box (window) with texts in the form of commands, which accepts direct inputs (also in the form of commands) from the users, and run them. The main indicator of this type is the fact that these softwares do not have any graphical images in its interface—in the sense that the interface of this type of programs contains only texts.

In this undergraduate thesis, a command line tool will be made, in which the tool would be able to run KIRI's API functions. Just like its general type, this tool will be run purely through the command line, and will not have any additional graphical images whatsoever in its interface. This newly-made tool will be made in C language, utilizing various libraries such as getopt, cJSON, cURL, and CMake, along with implementing the KIRI API features, as well as the general features of a command line tool, such as a help mode, and a man page (for Linux). This tool will later be tested by testing each of its functions, as well as the general cases of its integration with other, existing command line tools.

Keywords: Navigation, *angkot*, Project KIRI, command line, C

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR KODE PROGRAM	xiii
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	8
2.2 KIRI	11
2.2.1 Tampilan	11
2.2.2 API ¹	12
2.3 Fungsi dan <i>Library</i> Bahasa C	18
2.3.1 getopt [1]	19
2.3.2 libcurl [2]	22
2.3.3 cJSON ²	24
2.3.4 CMake [3]	29
3 ANALISIS	35
3.1 Analisis Aplikasi Sejenis	35
3.1.1 Chrome Web Store Item Property CLI	35
3.1.2 iTunes Search API	37
3.1.3 Uber CLI	38
3.1.4 Google Maps Direction CLI	40
3.2 Analisis API KIRI	41
3.2.1 Search Place	41
3.2.2 Routing	42
3.3 Analisis Perkakas yang Akan Dibuat	43
3.3.1 Analisis Fitur Perkakas	43
3.3.2 Analisis Use Case	46
4 PERANCANGAN	49
4.1 Rancangan Alur Kerja Perkakas	49

4.1.1	Mencari lokasi menggunakan kata kunci pencarian	49
4.1.2	Mencari rute dengan angkot menggunakan <i>latitude</i> dan <i>longitude</i> lokasi	50
4.1.3	Mencari rute dengan angkot menggunakan kata kunci pencarian lokasi	51
4.2	Rancangan Implementasi Perkakas	52
4.2.1	Cara Kerja Perkakas	52
4.2.2	Tipe Data Tambahan	53
4.2.3	Variabel Global	53
4.2.4	<code>print_help()</code>	56
4.2.5	<code>replace_space()</code>	56
4.2.6	<code>build_url_searchplace()</code>	56
4.2.7	<code>build_url_findroute()</code>	57
4.2.8	<code>reset_url()</code>	57
4.2.9	<code>execute_curl()</code>	57
4.2.10	<code>print_curl_error()</code>	57
4.2.11	<code>write_malloc()</code>	58
4.2.12	<code>write_searchplace()</code>	58
4.2.13	<code>write_findroute()</code>	58
4.2.14	<code>write_searchplace_noreturns()</code>	58
4.2.15	Fungsi utama (<code>main</code>)	58
5	IMPLEMENTASI DAN PENGUJIAN	67
5.1	Implementasi Kode	67
5.1.1	<code>print_help()</code>	67
5.1.2	<code>replace_space()</code>	67
5.1.3	<code>build_url_searchplace()</code>	67
5.1.4	<code>build_url_findroute()</code>	67
5.1.5	<code>reset_url()</code>	68
5.1.6	<code>execute_curl()</code>	68
5.1.7	<code>print_curl_error()</code>	68
5.1.8	<code>write_malloc()</code>	68
5.1.9	<code>write_searchplace()</code>	69
5.1.10	<code>write_findroute()</code>	69
5.1.11	<code>write_searchplace_noreturns()</code>	69
5.1.12	Fungsi utama (<code>main</code>)	69
5.1.13	CMakeLists	70
5.1.14	Halaman manual (<i>man page</i>)	70
5.2	Pengujian	70
5.2.1	Lingkungan Perangkat Keras	70
5.2.2	Lingkungan Perangkat Lunak	71
5.2.3	Pembangunan dan Instalasi	71
5.2.4	Pengujian	73
6	KESIMPULAN	89
6.1	Kesimpulan	89
6.2	Saran	89
DAFTAR REFERENSI		91
A KODE PERKAKAS <i>Command Line KIRI</i>		93

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.	9
2.4	Tampilan awal halaman web KIRI	12
2.5	Tampilan akhir halaman web KIRI	12
2.6	Halaman web <i>API Keys</i> KIRI.	14
2.7	Penggunaan API KIRI untuk layanan pencarian lokasi	15
2.8	Penggunaan API KIRI untuk layanan pencarian rute	17
2.9	Penggunaan API KIRI untuk layanan <i>smart direction</i>	18
2.10	Tampilan aplikasi cmake-gui	31
2.11	Tampilan aplikasi ccmake	32
3.1	Contoh penggunaan perkakas Chrome <i>Web Store Item Property CLI</i>	36
3.2	Contoh penggunaan perkakas <i>iTunes Search API</i>	38
3.3	Contoh penggunaan perkakas Uber CLI (<i>time</i>)	39
3.4	Contoh penggunaan perkakas Uber CLI (<i>price</i>)	39
3.5	Contoh penggunaan perkakas Google <i>Maps Direction CLI</i>	41
3.6	Diagram <i>use case</i> perkakas yang akan dibangun	46
4.1	<i>Activity diagram</i> fitur pencarian lokasi menggunakan kata kunci lokasi	50
4.2	<i>Sequence diagram</i> fitur pencarian lokasi menggunakan kata kunci lokasi	51
4.3	<i>Activity diagram</i> fitur pencarian rute angkot menggunakan koordinat lokasi	52
4.4	<i>Sequence diagram</i> fitur pencarian rute angkot menggunakan koordinat lokasi	53
4.5	<i>Activity diagram</i> fitur pencarian rute angkot menggunakan kata kunci lokasi	54
4.6	<i>Sequence diagram</i> fitur pencarian rute angkot menggunakan kata kunci lokasi	55

DAFTAR KODE PROGRAM

2.1	Contoh sederhana penggunaan getopt	20
2.2	Contoh sederhana penggunaan getopt-long	21
2.3	Loop sederhana dari penggunaan <i>multi handle curl</i>	24
2.4	Kumpulan implementasi penggunaan <i>multi socket handle curl</i>	24
2.5	Struktur data cJSON	26
2.6	Kode utama operasional CMake	30
2.7	Contoh kode pembangunan CMake lebih dari satu mode	33
A.1	CMakeLists.txt	93
A.2	main.c	93
A.3	kiritool.1 (<i>Source Code man page</i>)	102
A.4	Bantuan penggunaan perkakas	104
A.5	man page Perkakas <i>Command Line KIRI</i>	104

1

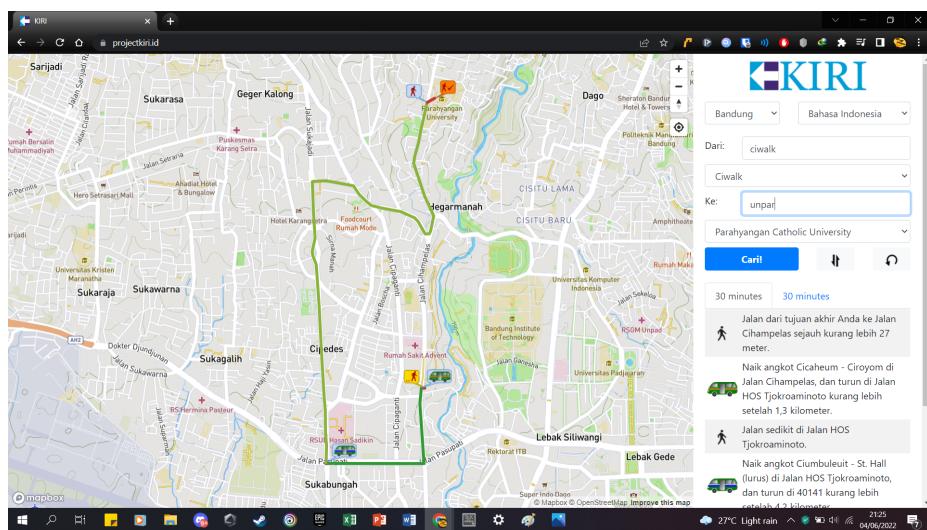
BAB 1

2

PENDAHULUAN

3 1.1 Latar Belakang

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



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

- 16 Sementara itu, dalam komputer, salah satu dari sekian banyak tipe perangkat lunak adalah
17 perangkat lunak *command line*. *Command line (command line interpreter*, atau *command line*

¹<https://projectkiri.id>

1 *interface*) adalah sebuah perangkat lunak berupa sebuah kotak/*window* yang memuat teks berupa
2 perintah-perintah,² yang menerima masukan dari pengguna dan menjalankannya.[4] Perintah-
3 perintah ini hanya berupa gabungan dari teks and simbol-simbol berupa karakter, tanpa ada
4 tambahan gambar grafis apapun. Singkatnya, tipe perangkat lunak ini bukan merupakan tipe yang
5 paling indah untuk dilihat oleh para pengguna, tetapi jika digunakan dengan tepat, maka jenis
6 perangkat lunak ini bisa menyuruh komputer untuk melakukan banyak sekali perintah-perintah
7 dengan sangat cepat dan sangat efektif.

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

13 1.2 Rumusan Masalah

14 Rumusan masalah yang akan dibahas dalam skripsi ini adalah sebagai berikut:

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

19 1.3 Tujuan

20 Tujuan dari skripsi ini adalah sebagai berikut:

- 21 1. Membangun perkakas *command line* yang dapat mengimplementasikan fitur-fitur API KIRI
22 dalam bahasa C.
- 23 2. Melakukan integrasi perkakas *command line* KIRI dengan perkakas-perkakas *command line*
24 lainnya.

25 1.4 Batasan Masalah

26 Batasan masalah dalam skripsi ini adalah sebagai berikut:

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

30 1.5 Metodologi

31 Metodologi yang akan diikuti dalam skripsi ini adalah sebagai berikut:

- 32 1. Melakukan studi dan eksplorasi terhadap fungsi-fungsi yang dimiliki perangkat lunak KIRI
33 serta cara implementasi API KIRI.

²<https://ubuntu.com/tutorials/command-line-for-beginners#3-opening-a-terminal>

- 1 2. Melakukan analisis dan desain perangkat lunak yang akan dibangun.
- 2 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.
- 5 4. Melakukan analisis kebutuhan fitur-fitur perangkat lunak dan melakukan eksplorasi *library* yang dapat digunakan dan memenuhi spesifikasi dalam pembuatan perangkat lunak.
- 7 5. Membangun perangkat lunak berdasarkan rancangan yang sudah dibuat, dengan megimplementasikan seluruh modul dan *library* yang telah ditentukan di tahap sebelumnya dalam bahasa C.
- 10 6. Melakukan pengujian fungsional, perbaikan *bug*, serta rekomendasi perbaikan berdasarkan pengujian yang sudah dilakukan.
- 12 7. Menyelesaikan pembuatan dokumen-dokumen yang berkaitan, seperti dokumen skripsi dan dokumentasi perangkat lunak.

14 1.6 Sistematika Pembahasan

15 Setiap bab dalam skripsi ini mengikuti sistematika yang terdiri atas poin-poin sebagai berikut:

16 1. Bab 1: Pendahuluan

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

19 2. Bab 2: Dasar Teori

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

22 3. Bab 3: Analisis

23 Bab ini berisi analisis perkakas-perkakas sejenis, analisis API KIRI, serta analisis kebutuhan perkakas yang akan dibuat.

25 4. Bab 4: Perancangan

26 Bab ini berisi pembahasan mengenai rancangan cara kerja tiap-tiap fitur perkakas yang akan dibuat.

28 5. Bab 5: Implementasi dan Pengujian

29 Bab ini berisi dua bagian utama, yaitu:

- 30 • Implementasi

31 Bagian ini meliputi struktur kelas dan penjelasan tiap-tiap fungsi di dalamnya.

- 32 • Pengujian

33 Bagian ini meliputi cara instalasi, cara menggunakan perkakas, serta pengujian fungsional terhadap fitur-fitur dari perkakas yang telah dibuat.

35 6. Bab 6: Kesimpulan dan Saran

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

BAB 2

LANDASAN TEORI

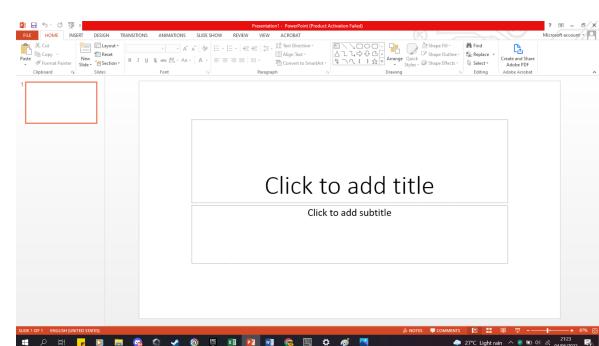
3 2.1 *Command Line*

4 *Command line* (atau *command line interface*) dapat diartikan sebagai tampilan antarmuka/*interface*
 5 yang memproses perintah dari pengguna dan meneruskannya langsung ke sistem operasi untuk
 6 dijalankan.^[5] Seluruh sistem operasi komputer yang ada memiliki sebuah *command line interface*
 7 dalam bentuk *shell*, yang dapat digunakan oleh penggunanya untuk langsung mengakses fungsi
 8 atau servis yang disediakan oleh sistem operasi.^[6]

9 2.1.1 *Command Line Interface* dan *Graphical User Interface*

10 Ada beberapa dari tipe antarmuka yang masih banyak digunakan di zaman sekarang, tetapi dua tipe
 11 yang paling banyak muncul adalah *command line interface* dan *graphical user interface*. Perangkat
 12 lunak berbasis *command line* sendiri bisa memiliki berbagai macam tampilan, tetapi semuanya
 13 selalu mengikuti satu bentuk antarmuka umum. Bentuk yang dimaksud adalah sebuah area/*window*
 14 yang memuat teks berupa perintah-perintah dari user untuk dilakukan oleh komputer, beserta
 15 keluarannya yang juga berupa teks, seperti dapat dilihat pada gambar 2.1a. Jenis perangkat lunak
 16 seperti ini disebut memiliki antarmuka jenis *command line interface* (CLI). Adapun dekorasi visual
 17 yang dimiliki oleh jenis tampilan ini hanya berupa warna pada teks-teks yang ada, tanpa tambahan
 18 gambar apapun. Jika perangkat lunak tersebut memiliki dekorasi dan/atau tombol interaktif berupa
 19 gambar grafis, seperti pada gambar 2.1b, maka perangkat lunak tersebut dikategorikan sebagai
 20 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 3 devasc devasc 4096 Sep 18 2021 downloads
drwxr-xr-x 4 devasc devasc 4096 Dec 14 22:44 Labs
drwxr-xr-x 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
Utique latet et non videntur. Non enim est in nobis
Loren ipsum dolor sit amet, consectetur adipisciing elit. Collatio igitur ista te nihil
tuuat. Honesta oratio, Socratica, Platonis etiam. Primum in nostrane potestate est, qui
d meminerimus? Duo Reges: constructio interrete. Quid, si etiam lucunda memoria est pra
eteritorum malorum? Si quidem, inquit, tollerem, sed relinqui. An nisi populari fama?
Quamquam id quidem licetit lis existimare, qui legerint. Summum a vobis bonum voluptas
dictur. At hoc in eo M. Refert tamen, quo modo. Quid sequatur, quid repugnet, vident.
Iam id ipsum absurdum, maximum malum neglegit.
devasc@labvm:~/labs/personal/samples$
```



(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.[6]

- 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.

2.1.2 *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*).[7]

Tampilan

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

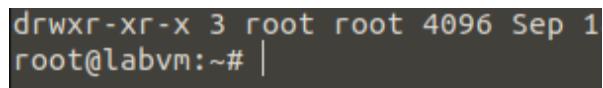
```
<nama pengguna>@<nama perangkat>:<direktori yang sedang diproses>$
```

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*.[5]



```
drwx----- 5 devasc devasc 4096 Sep 1
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 [5]

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.

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

- pwd

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

- ls

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

- cd

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

```
cd <direktori yang diinginkan>
```

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

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

33 man [5]

Untuk sistem-sistem operasi berbasis Linux, ada sebuah konvensi bahwa semua perangkat lunak *executable* yang dimaksudkan untuk dijalankan melalui *command line* perlu disertai dengan sebuah dokumentasi formal yang disebut *manual* atau *man page*. Di sistem-sistem operasi ini sudah ada perangkat lunak khusus, bernama “*man*” yang fungsinya adalah untuk menampilkan dokumentasi-dokumentasi ini. Adapun *man* digunakan dengan perintah berikut:

man <nama perangkat lunak>

- ¹ *Man page* tidak memiliki format umum, tapi pada umumnya memiliki bagian-bagian berikut:
- ² • judul—perangkat lunak mana yang sedang ditampilkan halaman *man*-nya,
 - ³ • perintah penggunaan perangkat lunak,
 - ⁴ • deskripsi singkat dari fungsi perangkat lunak, serta
 - ⁵ • daftar fitur-fitur perangkat lunak serta cara penggunaannya.
- ⁶ Walaupun begitu, perlu diingat bahwa halaman *man* umumnya tidak mengikutkan contoh penggunaan. Selain itu, perlu juga diingat bahwa halaman *man* ini dibagi ke delapan bagian, di mana bagian-bagiannya dapat dilihat di tabel 2.1.

Bagian	Kategori <i>manual</i>
1	Perintah pengguna
2	Panggilan langsung ke <i>kernel</i> sistem
3	Panggilan langsung ke <i>library C</i>
4	File-file spesial (contohnya <i>driver</i>)
5	Format file
6	Permainan
7	Lain-lain
8	Perintah administrasi sistem

Tabel 2.1: Jumlah kategori dan tes yang dilakukan.

- ⁹ Adapun contoh penggunaan perintah `man` ini ada di gambar <REF!>.

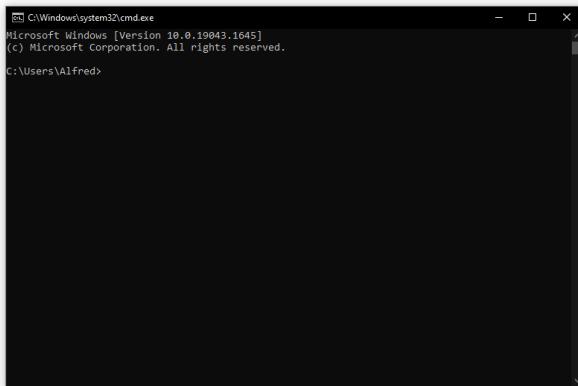
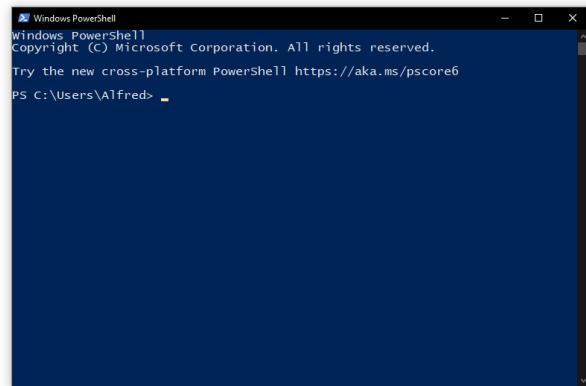
¹⁰ 2.1.3 *Command Line* di Windows

¹¹ Cara kerja *command line* di Windows serupa dengan cara kerja *command line* di Linux, dalam ¹² arti bahwa untuk bekerja dengan *command line* di Windows, penggunanya juga akan langsung ¹³ berinteraksi dengan utilitas yang disediakan oleh sistem operasi. *Command line* di Windows juga ¹⁴ dapat digunakan untuk hal-hal yang serupa dengan *command line* di Linux, seperti menulis (dan ¹⁵ menjalankan) *script*, menjalankan perintah yang diinginkan pengguna secara otomatis, atau melihat ¹⁶ status dari sistem operasi.[6]

¹⁷ Masih sama dengan Linux, ada banyak sekali command yang bisa digunakan, sehingga susah ¹⁸ untuk menghafal seluruh command-command yang ada—termasuk masukan, parameter-parameter ¹⁹ yang dibutuhkan, serta keluarannya. Untuk melihat dokumentasi, atau penjelasan detail untuk ²⁰ masukan, keluaran, parameter, serta opsi-opsi dari perintah tertentu, pengguna dapat memasukkan ²¹ perintah tersebut, diikuti dengan /?.[6]

²² Tampilan

²³ Di sistem operasi Windows, ada dua jenis antarmuka *command line*, yaitu cmd (*Command Prompt*) ²⁴ dan *PowerShell*. Keduanya memiliki tampilan yang kurang lebih sama—hanya saja awalnya cmd ²⁵ memiliki latar belakang hitam, sedangkan *PowerShell* memiliki latar belakang biru tua, seperti ²⁶ terlihat di gambar 2.3.

(a) Antarmuka Windows *Command Prompt* (*cmd*)(b) Antarmuka Winodws *PowerShell*Gambar 2.3: Tampang kedua antarmuka *command line* bawaan di sistem operasi Windows.

1 Navigasi

2 Untuk navigasi di antarmuka *command line* Windows, ada dua perintah penting yang dipakai ketika
 3 pengguna sedang berurusan dengan file-file dan navigasi dalam direktori sistem. Kedua perintah
 4 tersebut adalah **cd** dan **dir**.

5 • **cd** (**chdir**) [8]

6 **cd** merupakan sebuah perintah yang memiliki tiga fungsi utama, yaitu menampilkan *drive*
 7 tempat sedang *command line* berada (jika pengguna hanya memasukkan **cd** tanpa parameter
 8 apapun), menampilkan direktori tempat *command line* sedang berada (jika pengguna hanya
 9 memasukkan *drive* sebagai parameter, atau fungsi yang paling umumnya, untuk mengganti
 10 *working directory* dari *command line*.

11

12 Adapun format dari perintah **dir** adalah sebagai berikut.

13 **cd** [/d] [<drive>:] [<path>]

14 Dengan fungsi dari semua opsi dan parameter yang ada sebagai berikut.

15 – /d

16 Opsi yang menandakan bahwa pengguna ingin mengganti *drive* (partisi) dan juga *working*
 17 *directory* dari *command line*.

18 – <drive>:

19 Kode huruf dari partisi yang akan diproses.

20 – <path>

21 Direktori yang akan diproses. Parameter ini harus diikutkan beserta kode huruf partisi
 22 (tidak dapat berdiri sendiri.)

23 • **dir**

24 **dir** merupakan sebuah perintah yang mengeluarkan/menampilkan sebuah daftar berisi file-file
 25 yang ada di suatu direktori, termasuk subdirektori. Jika tidak disertai parameter apapun,
 26 perintah ini akan menampilkan label volume dan nomor serial *disk*, dilanjutkan dengan daftar
 27 direktori dan file di dalamnya. Untuk file, akan ditampilkan nama beserta ukurannya. Perintah
 28 ini juga akan menampilkan jumlah direktori dan file yang didaftarkan, ukuran kumulatifnya,

1 dan sisa dari *disk* yang tidak terpakai (dalam *bytes*).[8]

2
3 Adapun format dari perintah **dir** adalah sebagai berikut.[6]

4 **dir** [<drive:>] [<path>] [<filename>] [/A[:<attributes>]] [/B]
5 [/C] [/D] [/L] [/N] [/O[:<sortorder>]] [/P] [/Q] [/R] [/S]
6 [/T[:<timefield>]] [/W] [/X] [/4]

7 Untuk perintah ini, seperti terlihat di atas, memiliki banyak sekali opsi dan parameter.
8 Tiap-tiap dari parameter tersebut memiliki fungsi tersendiri, yaitu:

9 – /A[:<attributes>]

10 Menampilkan file-file dengan atribut tertentu, seperti file yang disembunyikan, file sistem,
11 file *read-only*, dan sebagainya.

12 – /B

13 Menghilangkan *heading* dan ringkasan informasi dari keluaran, atau dengan kata lain,
14 hanya menampilkan file-file dan direktori, tanpa informasi tambahan apapun.

15 – /C

16 Menggunakan separator koma untuk tiap angka ribuan di ukuran file. Jika opsi yang
17 dimasukkan adalah /-C, separator koma justru akan dihilangkan.

18 – /D

19 Menampilkan keluaran dengan format yang lebih lebar. Jika opsi ini diikutkan, keluaran
20 akan ditampilkan dengan urutan berdasarkan kolom.

21 – /L

22 Seluruh teks dalam keluaran akan menggunakan huruf kecil. Jika opsi ini tidak digunakan,
23 keluaran akan mengandung huruf besar dan huruf kecil *mixed case*.

24 – /N

25 Menampilkan daftar dengan format panjang, dengan nama file berada di ujung paling
26 kanan.

27 – /O[:<sortorder>]

28 Menampilkan daftar direktori yang terurut berdasarkan urutan tertentu, seperti ber-
29 dasarkan ekstensi file, berdasarkan tanggal dibuat, berdasarkan nama, dan sebagainya.
30 Jika tidak diikutkan tanda minus (-) sebelum huruf O pada perintah, daftar yang muncul
31 akan terurut secara menaik.

32 – /P

33 Memberhentikan keluaran selama beberapa waktu singkat (memberi jeda kecil) setelah
34 setiap halaman informasi.

35 – /Q

36 Menambahkan informasi mengenai pemilik file dalam keluaran.

37 – /R

38 Menampilkan *data stream* alternatif, jika ada.

39 – /S

40 Mendaftarkan seluruh file di direktori dan subdirektori yang diproses. Tiap-tiap direktori
41 akan memiliki *header* tersendiri dalam keluarannya.

42 – /T[:<timefield>]

1 Menspesifikasi *time field* mana yang akan tampil dan digunakan sebagai urutan, jika
2 aturan pengurutan lain tidak ditentukan. *Time field* yang dapat digunakan adalah waktu
3 pembuatan file, kapan terakhir file diakses, dan kapan file terakhir dimodifikasi. Jika
4 parameter ini tidak dispesifikasi, *time field* yang digunakan adalah kapan file terakhir
5 dimodifikasi.

6 – /W

7 Menampilkan keluaran dengan format yang lebih lebar. Opsi ini hampir sama dengan /D,
8 hanya saja untuk /W, jika opsi ini diikutkan, keluaran akan ditampilkan dengan urutan
9 berdasarkan baris, dan bukan kolom.

10 – /X

11 Menampilkan nama pendek yang dibuat untuk nama-nama file non-8.3. Opsi ini memiliki
12 format tampilan yang sama seperti opsi /N, hanya saja nama pendeknya ditampilkan di
13 keluaran sebelum nama panjangnya.

14 – 4

15 Menampilkan angka tahun dengan format angka empat digit.

16 2.2 KIRI

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

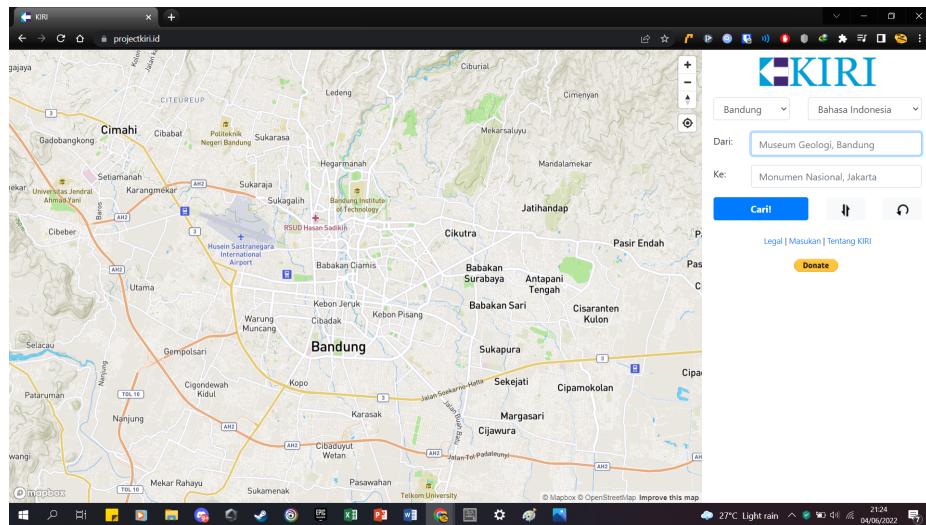
26 Di halaman web KIRI, pengguna dapat memasukkan input berupa lokasi awal dan lokasi tujuan
27 dan KIRI akan menghasilkan seluruh langkah yang harus ditempuh oleh pengguna untuk sampai
28 ke lokasi tujuan, dengan menggunakan angkot. Keluaran ini sudah meliputi kode angkot mana saja
29 yang harus dinaiki, dan juga seberapa jauh pengguna harus berjalan kaki untuk sampai ke lokasi
30 rute angkot berikutnya.

31 2.2.1 Tampilan

32 Pada saat pertama kali dibuka, hal pertama yang paling mencolok di halaman awal web KIRI adalah
33 sebuah peta besar di sebelah kiri yang dapat diperbesar ataupun diperkecil. Sedangkan, bagian
34 kanan dari halamannya terdiri atas beberapa bagian. Di bagian paling atas terdapat logo KIRI,
35 beserta sepasang menu *dropdown*—yang pertama merupakan pilihan kota tempat pengguna berada
36 (untuk sekarang hanya tersedia pilihan kota Jakarta dan Bandung), dan yang kedua merupakan
37 pilihan bahasa, entah bahasa Indonesia atau Inggris. Di bawahnya merupakan sepasang menu
38 *dropdown* yang merupakan tempat di mana pengguna memasukkan lokasi awal dan tujuan yang

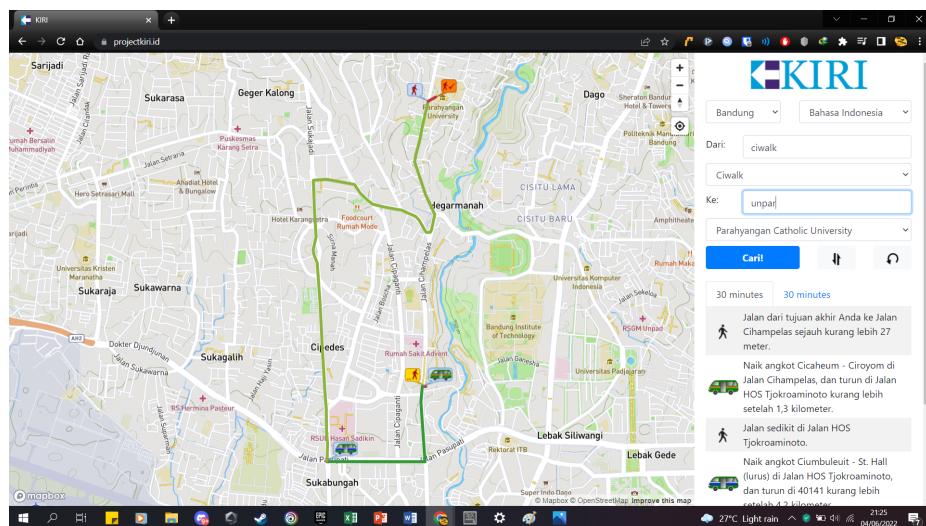
¹<https://projectkiri.github.io/#about-kiri>

- 1 akan diproses oleh KIRI. Terakhir, di bawahnya ada sebuah bagian kosong, yang nantinya akan
 2 menjadi tempat di mana KIRI akan meletakkan keluaran dari prosesnya. Adapun tampilan awal
 3 dari halaman web ini dapat dilihat di gambar 2.4.



Gambar 2.4: Tampilan awal halaman web KIRI.

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



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

11 2.2.2 API²

²<https://github.com/projectkiri/Tirtayasa/wiki/KIRI-API-v2>

1 KIRI juga memiliki sebuah API yang dapat digunakan untuk keperluan pengembangan perangkat
2 lunak. API ini menyediakan tiga buah jenis layanan web (*webservice*), yang ketiganya dapat
3 dilakukan dengan mengirim permintaan (*request*) tipe GET melalui API tersebut. Isi dari permintaan
4 yang perlu dikirimkan serta respon dari API yang akan dikembalikan berbeda tergantung dari jenis
5 layanan yang digunakan. Adapun ketiga jenis layanan tersebut adalah pencarian tempat (*search*
6 *place*), pencarian rute (*routing*), dan *smart direction*.

7 **Search Place**

8 Layanan pencarian lokasi (*search place*) adalah layanan web pada API KIRI yang berfungsi untuk
9 mencari suatu lokasi berdasarkan kata kunci yang diberikan oleh pengguna. Untuk menggunakan
10 layanan ini, pengguna harus mengirim permintaan GET ke alamat <https://projectkiri.id/api>.
11 Adapun permintaan tersebut harus memiliki parameter-parameter seperti terlihat di bawah ini.

- 12 • **version**

13 **Kemungkinan nilai:** 2

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

- 15 • **mode**

16 **Kemungkinan nilai:** searchplace

17 Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna.
18 Untuk penggunaan layanan pencarian lokasi, variabel ini harus diisi dengan **searchplace**.

- 19 • **region**

20 **Kemungkinan nilai:** cgk, bdo, mlg, atau sub

21 Parameter ini merupakan kode bandara IATA tiga huruf yang merepresentasikan daerah mana
22 tempat lokasi yang ingin dicari berada. Kode yang dapat diproses oleh API ini meliputi **cgk**
23 (Cengkareng/Jakarta), **bdo** (Bandung), **mlg** (Malang), dan **sub** (Surabaya).

- 24 • **querystring**

25 **Kemungkinan nilai:** string berisi teks apapun dengan panjang minimal satu karakter

26 Parameter ini berisi kata kunci yang akan digunakan untuk menentukan lokasi yang ingin
27 dicari pengguna.

- 28 • **apikey**

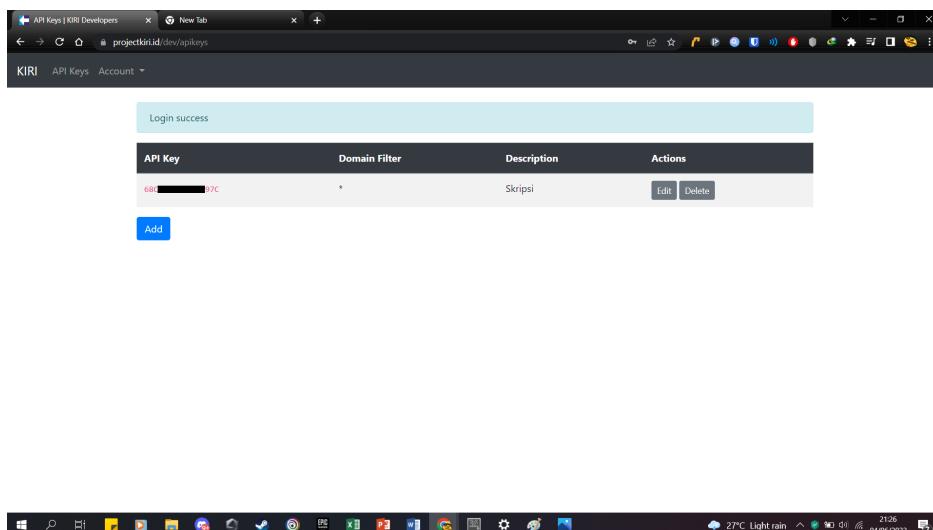
29 **Kemungkinan nilai:** angka heksadesimal 16-digit

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

32 Perlu diperhatikan bahwa salah satu dari parameter yang harus diikutkan dalam pesan tersebut
33 merupakan parameter yang meminta kunci API. Kunci tersebut harus digenerasikan terlebih dahulu
34 sebelum API KIRI dapat digunakan, melalui halaman *API Keys* KIRI,³ yang dapat dilihat di
35 gambar 2.6.

36 Untuk mengakses halaman tersebut, pengguna harus membuat sebuah akun terlebih dahulu.
37 Ketika akun sudah dibuat, maka pengguna baru akan dapat membuat kunci API yang dibutuhkan,
38 sekaligus membuat filter *domain*, yang membatasi di *domain* mana saja kunci tersebut dapat
39 digunakan, serta memberikan deskripsi untuk kunci API tersebut. Kunci ini kemudian dapat

³<https://projectkiri.id/dev/apikeys>



Gambar 2.6: Halaman web *API Keys* KIRI.

digunakan sebagai nilai dari parameter `apikey` yang diperlukan dalam permintaan tadi.

Sebelum membahas keluaran dari layanan API ini, perlu ditegaskan dulu apa definisi dari nilai *latitude* dan *longitude* suatu lokasi. *Latitude* merupakan berapa derajat sebuah tempat berada dari garis ekuator, dengan lokasi-lokasi yang berada maksimum 90 derajat di atas ekuator memiliki nilai *latitude* positif, sedangkan lokasi-lokasi yang berada maksimum 90 derajat di bawah ekuator memiliki nilai *latitude* negatif. Sedangkan, *longitude* merupakan berapa derajat lokasi sebuah tempat berada dari garis meridian (bujur) utama Bumi, dengan rentang nilai dari -180 derajat di sisi kiri (barat) meridian utama, hingga 180 derajat di kanan (timur) bujur tersebut.⁴ Kedua nilai ini merupakan salah satu dari dua variabel yang dikembalikan dalam respon API untuk layanan ini, dengan variabel lainnya berupa nama dari lokasi yang ditemukan itu sendiri. Adapun respon yang diberikan oleh API akan 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`.

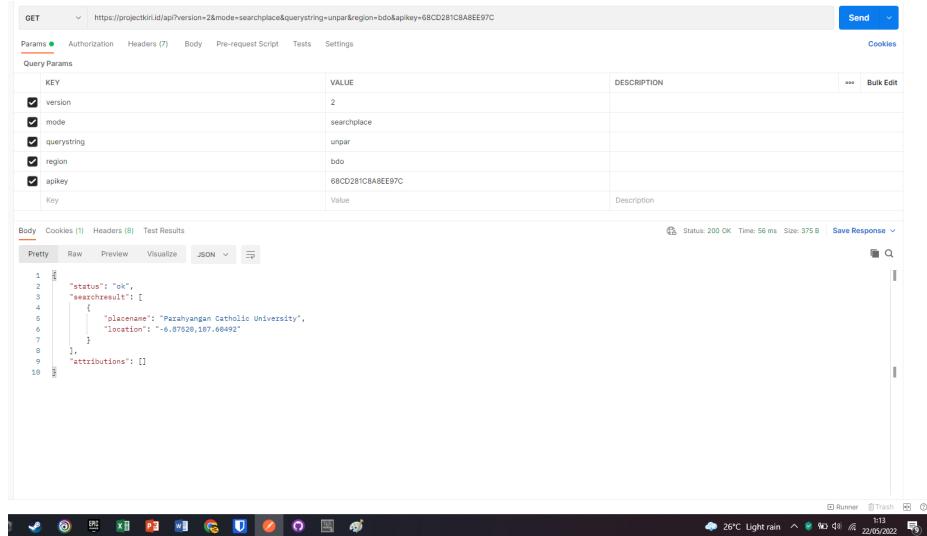
- **message**

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 pengguna berhasil diproses, variabel ini akan mengalami dua perubahan utama. Pertama, nama variabel ini akan berubah menjadi `searchresult`, dan kedua, isi dari variabel ini akan menjadi sebuah `array` yang merupakan respon dari API KIRI berupa keluaran yang akan dilihat oleh pengguna. `Array` ini sendiri akan memiliki variabel berikut.

- **placename**

Variabel ini berisi nama lokasi yang ditemukan berdasarkan kata kunci yang diberikan oleh pengguna.

⁴https://gsp.humboldt.edu/olm/Lessons/GIS/01%20SphericalCoordinates/Latitude_and_Longitude.html



Gambar 2.7: Penggunaan API KIRI untuk layanan pencarian lokasi menggunakan Postman. Gambar ini menunjukkan hasil pencarian lokasi “unpar” di daerah Bandung.

- 1 – **location**
2 Variabel ini berisi nilai *latitude* dan *longitude* dari lokasi yang ditemukan dalam pencarian.

3 Contoh dari penggunaan API KIRI untuk layanan ini dapat dilihat di gambar 2.7.

4 **Routing**

5 Layanan pencarian rute (*routing*) adalah layanan web pada API KIRI yang memiliki fungsi yang
6 sama dengan fungsi utama dari perangkat lunak KIRI sendiri, yaitu menunjukkan rute serta
7 langkah-langkah yang harus ditempuh untuk pergi dari satu lokasi ke lokasi lainnya, dengan
8 menggunakan angkot yang tersedia. Untuk menggunakan layanan ini, pengguna harus mengirim
9 permintaan GET ke alamat <https://projectkiri.id/api>. Adapun permintaan tersebut harus memiliki
10 parameter-parameter seperti terlihat di bawah ini.

11 • **version**

12 **Kemungkinan nilai:** 2

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

14 • **mode**

15 **Kemungkinan nilai:** findroute

16 Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna.
17 Untuk penggunaan layanan pencarian rute dengan angkot, variabel ini diisi dengan **findroute**.

18 • **locale**

19 **Kemungkinan nilai:** en atau id

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

23 • **start**

24 **Kemungkinan nilai:** lat, lng; dalam bentuk desimal 10-digit

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

- 2 • **finish**

3 **Kemungkinan nilai:** `lat`, `lng`; dalam bentuk desimal 10-digit

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

- 5 • **presentation** (opsional)

6 **Kemungkinan nilai:** `desktop`

7 Parameter ini hanya digunakan untuk fitur *backwards compatibility*.

- 8 • **apikey**

9 **Kemungkinan nilai:** angka heksadesimal 16-digit

10 Parameter ini berisi kunci API pribadi yang harus digenerasi terlebih dahulu sebelum API

11 dapat digunakan.

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

- 14 • **status**

15 **Kemungkinan nilai:** `ok` atau `error`

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

- 18 • **message**

19 Mirip dengan fitur `searchplace`, jika permintaan dari pengguna tidak berhasil diproses,
20 variabel ini akan berupa *string* yang berisi pesan error dari API. Jika permintaan dari
21 pengguna berhasil diproses, nama variabel ini akan berubah menjadi `routingresults`, dan
22 isi dari variabel ini akan menjadi sebuah *array* JSON yang berisi variabel-variabel sebagai
23 berikut:

- 24 – **steps**

25 **Tipe:** *array*

26 Variabel ini merepresentasikan satu buah langkah yang harus ditempuh oleh pengguna.

27 Adapun *array* ini sendiri berisi variabel-variabel berikut:

- 28 * Tipe transportasi

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

- 32 * Kode angkot

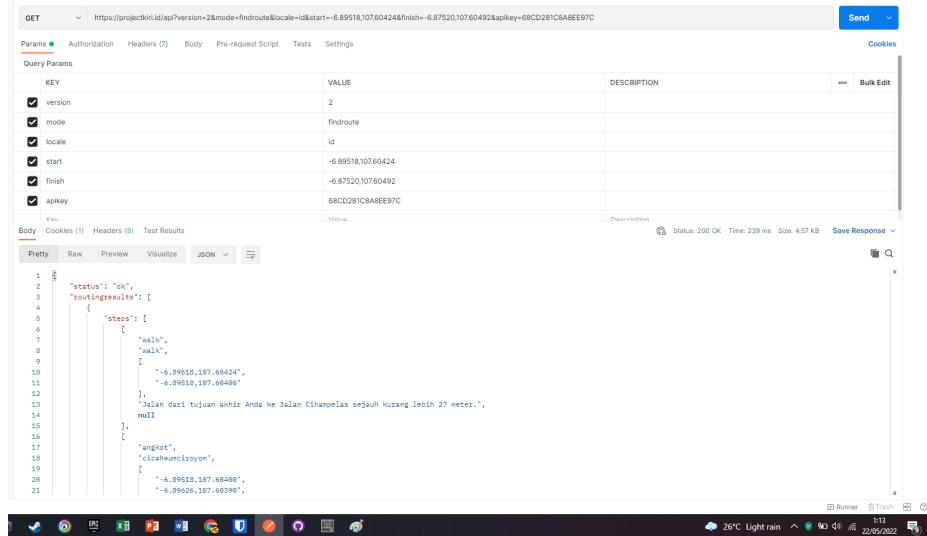
33 Variabel ini menunjukkan angkot mana yang harus dinaiki oleh pengguna di langkah
34 tersebut. Jika penggunaan angkot tidak dimungkinkan pada langkah ini (pengguna
35 harus berjalan kaki), variabel ini akan berisi `walk`.

- 36 * *Array latitude* dan *longitude* lokasi

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

- 39 * Deskripsi langkah

40 Deskripsi langkah yang harus ditempuh, dalam bahasa natural. Bahasa yang digu-
41 nakan tergantung parameter `locale` yang diatur dalam masukan.



Gambar 2.8: Penggunaan API KIRI untuk layanan pencarian rute menggunakan Postman. Gambar ini menunjukkan hasil pencarian rute dari Cihampelas Walk ke UNPAR.

- * 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 melakukan modifikasi rute, jika dimungkinkan. Jika rute tidak bisa dimodifikasi, variabel ini akan berisi `null`.

– **traveltime**
Tipe: string
Variabel ini berisi estimasi jangka waktu yang diperlukan untuk menyelesaikan langkah tersebut.

Adapun gambar 2.8 menunjukkan penggunaan API KIRI untuk layanan pencarian rute dari Cihampelas Walk ke Universitas Katolik Parahyangan.

Smart Direction

Layanan terakhir dari API ini adalah layanan *smart direction*, yang merupakan gabungan dari kedua layanan sebelumnya. Berbeda dengan kedua layanan tadi, yang harus mengakses API secara manual (dengan mengirimkan permintaan GET), layanan ini tidak memerlukan pengguna untuk mengirim permintaan apapun—layanan ini sudah otomatis menangani permintaan pengguna. Berbeda dengan kedua layanan sebelumnya juga, layanan ini tidak memerlukan dibuatnya kunci API terlebih dahulu.

Layanan ini bekerja dengan mengalihkan pengguna langsung ke halaman web KIRI yang sudah langsung menunjukkan rute yang perlu ditempuh untuk pergi dari lokasi satu ke lokasi lainnya. Untuk melakukan hal ini, layanan ini memerlukan sebuah URL, yang memiliki format sebagai berikut:

`https://projectkiri.id?start=<lokasi awal>&finish=<lokasi akhir>&locale=<locale>`

Dapat dilihat bahwa URL tersebut memiliki tiga buah parameter, yaitu:

1 • **start**

2 **Kemungkinan nilai:** Nilai *latitude* dan *longitude* lokasi, atau nama lokasi tersebut
 3 Parameter ini berisi lokasi yang ingin digunakan sebagai lokasi mulainya pencarian rute.

4 • **finish**

5 **Kemungkinan nilai:** Nilai *latitude* dan *longitude* lokasi, atau nama lokasi tersebut
 6 Parameter ini berisi lokasi yang merupakan tujuan akhir yang ingin dicapai dalam pencarian
 7 rute.

8 • **locale (opsional)**

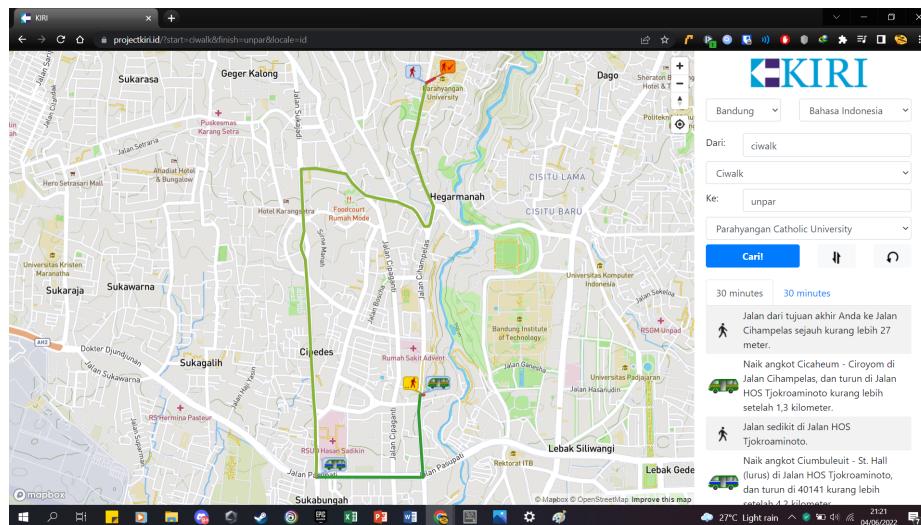
9 **Kemungkinan nilai:** *id* atau *en*

10 Menentukan dalam bahasa apa hasil pencarian rutennya akan ditampilkan (bahasa Indonesia
 11 atau bahasa Inggris). Jika parameter ini tidak diberikan oleh pengguna, maka bahasa yang
 12 akan digunakan adalah bahasa yang terakhir dipakai di halaman web KIRI sendiri.

13 Misalkan pengguna ingin mencari rute dari Cihampelas Walk ke Universitas Katolik Parahyangan,
 14 dan menampilkan langkah-langkah yang harus ditempuh dalam rutennya dalam bahasa Indonesia.
 15 Pengguna dapat memasukkan URL berikut ke peramban mereka.

16 <https://projectkiri.id?start=ciwalk&finish=unpar&locale=id>

17 Jika URL tersebut sudah dimasukkan ke kotak *link* pada peramban, halaman yang akan ditampilkan
 18 akan terlihat seperti pada gambar 2.9.



Gambar 2.9: Penggunaan API KIRI untuk layanan *smart direction*, dari Cihampelas Walk ke UNPAR.

19 **2.3 Fungsi dan *Library* Bahasa C**

20 Di bagian ini akan dilakukan studi literatur terhadap seluruh fungsi bawaan serta *library-library*
 21 bahasa pemrograman C yang akan digunakan dalam pembuatan perkakas ini.

1 2.3.1 getopt [1]

2 getopt merupakan sebuah fungsi yang dapat mengautomasi pekerjaan-pekerjaan yang berhubungan
3 dengan penerimaan opsi-opsi untuk *command line* berbasis UNIX.

4

5 Fungsi getopt dapat dipanggil dengan format sebagai berikut.

6 getopt (argc, argv, <options>)

7 Seluruh kode ini dapat dimasukkan ke suatu variabel berupa sebuah karakter yang merepresentasikan opsi yang ingin digunakan. argc merupakan jumlah argumen yang terdapat dalam masukan, sedangkan argv merupakan sebuah *array* yang berisi argumen-argumen tersebut.

10

11 Selain itu, penggunaan getopt juga akan memakai variabel-variabel tertentu, yang nilainya akan diisi oleh fungsi getopt tersebut sendiri. Variabel-variabel ini beserta penjelasannya dapat dilihat di daftar berikut.⁵

12 • opterr

13 Isi dari variabel ini akan memberi sinyal ke perangkat lunak/perkakas yang menentukan apakah getopt akan mengirim pesan ke *error stream* atau tidak. Jika variabel ini bukan bernilai 0, maka pesan *error* akan dikirim. Sebaliknya, jika variabel ini bernilai 0, getopt tidak akan mengirim pesan *error* apapun, tetapi tetap akan mengembalikan sebuah karakter tanda tanya (?) sebagai tanda bahwa sebuah *error* telah terjadi.

14 • getopt

15 Ketika getopt menemukan sebuah karakter yang tidak didefinisikan dalam kumpulan opsi, atau sebuah opsi yang tidak disertai argumen yang diperlukan, karakter tersebut akan disimpan di variabel ini.

16 • optind

17 Variabel ini digunakan oleh getopt sebagai indeks untuk *array* argv. Jika seluruh argumen sudah diproses, nilai variabel ini dapat digunakan untuk menentukan argumen mana yang merupakan arguman tambahan yang tidak terpakai. Nilai dari variabel ini dimulai dari 1.

18 • optarg

19 Jika opsi yang sedang diproses memerlukan argumen, variabel ini adalah tempat dimana argumen tersebut akan disimpan.

20 • <options>

21 Variabel ini merupakan salah satu variabel yang tertera di format pemanggilan getopt diatas. Variabel ini berupa *string* yang menandakan karakter-karakter apa saja yang menjadi opsi yang mungkin dalam perkakas tersebut, beserta tipenya. Jika karakter opsi:

- 22 – Diikuti dengan titik dua (:), maka opsi tersebut memiliki argumen yang bersifat wajib.
- 23 – Diikuti dengan titik dua ganda (::), maka opsi tersebut memiliki argumen yang bersifat opsional.
- 24 – Tidak diikuti apa-apa, maka opsi tersebut merupakan opsi tidak berarguman.

25 Untuk memberikan gambaran yang lebih baik mengenai penggunaan fungsi getopt, berikut merupakan contoh perangkat lunak berbasis *command line* sederhana yang menggunakan fungsi tersebut.

5 https://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html

- 1 Perangkat lunak ini akan menerima masukan berupa opsi tidak berparameter **-a** dan/atau opsi
 2 berparameter **-b**. Untuk opsi **-a**, perangkat lunak ini akan mengeluarkan nilai 0 jika opsi **-a** tidak
 3 dipakai, dan 1 jika opsi tersebut dipakai. Sebagai keluaran keduanya, perangkat lunak ini akan
 4 mengeluarkan isi parameter dari opsi **-b**, atau **NULL** jika opsi **-b** tidak dipakai.

Kode 2.1: Contoh sederhana penggunaan getopt

```

5   #include <ctype.h>
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <unistd.h>
9
10
11  int main(int argc, char **argv) {
12      int aflag = 0;
13      char *bvalue = NULL;
14      int index;
15      int args;
16
17      opterr = 0;
18
19      while ((args = getopt(argc, argv, ":ab:")) != -1)
20          switch (args) {
21              case 'a':
22                  aflag = 1;
23                  break;
24              case 'b':
25                  bvalue = optarg;
26                  break;
27              case ':':
28                  if (optopt == 'b') {
29                      fprintf(stderr, "Option_-%-c_-requires_-an_-argument.\n", optopt);
30                  }
31                  return 1;
32              case '?':
33                  if (isprint(optopt)) {
34                      fprintf(stderr, "Unknown_option_-%-c_-.\n", optopt);
35                  }
36                  else fprintf(stderr, "Unknown_option_character_-'\\"\\x%'"_.\n", optopt);
37                  return 1;
38              default:
39                  abort();
40          }
41
42      printf("aflag_=_%d,_bvalue_=_%s\n", aflag, bvalue);
43      for (index = optind; index < argc; index++)
44          printf("Non-option_argument_-%s\n", argv[index]);
45
46      return 0;
47 }

```

49 getopt-long

50 Ada pula versi **getopt** yang memungkinkan perangkat lunak untuk menerima dua jenis opsi—opsi
 51 versi pendek berupa sebuah karakter singular, seperti pada **getopt** biasa, dan/atau opsi panjang
 52 bergaya GNU, berupa sebuah kata. Perlu diingat juga bahwa perkakas yang menerima opsi sebagai
 53 masukannya sebaiknya memiliki versi panjang dari opsi-opsi tersebut, karena tidak hanya hal ini
 54 tidak sulit untuk diimplementasikan, tetapi implementasi versi panjang dari opsi-opsi yang ada juga
 55 akan memudahkan pengguna untuk mengingat cara kerja perkakas tersebut.

56 **getopt-long** juga memiliki seluruh variabel-variabel yang dimiliki oleh **getopt**, hanya saja
 57 **getopt-long** memiliki sebuah variabel tambahan berupa struktur, yaitu **long_options**. Variabel
 58 ini merupakan sebuah struktur berupa *array* yang berisi beberapa *array* lainnya, di mana *array-array*
 59 lain ini merupakan masing-masing opsi dari fungsi **getopt-long** tersebut. Tiap-tiap *array* tersebut
 60 memiliki variabel-variabel berikut:

- 61 • name

1 Variabel ini merupakan nama panjang dari opsi.

2 • **has_arg**

3 Variabel ini merupakan penanda apakah opsi memerlukan argumen atau tidak. Nilai untuk
4 variabel ini adalah `no_argument`, `required_argument`, atau `optional_argument`.

5 • **flag & val**

6 Kedua variabel ini menandakan bagaimana sebuah opsi akan diberlakukan ketika diterima
7 oleh `getopt-long`. Variabel `flag` dapat diisi dengan penunjuk (*pointer*) ke suatu variabel
8 lain yang akan diisi dengan isi dari variabel `val` untuk menandakan bahwa `getopt-long` telah
9 berhasil memroses opsi tersebut. Di lain sisi, jika variabel ini berisi *null pointer*, maka fungsi
10 `getopt-long` akan mengembalikan isi dari variabel `val`.

11 Struktur ini harus diakhiri dengan sebuah *array* tambahan yang seluruh variabelnya bernilai 0.

12

13 Untuk menjelaskan lebih lanjut mengenai cara penggunaan `getopt-long`, berikut merupakan
14 contoh perangkat lunak berbasis *command line* sederhana yang menggunakan fungsi tersebut.

15 Adapun perangkat lunak ini memiliki spesifikasi sebagai berikut:

- 16 • Perangkat lunak ini akan menerima masukan dari penggunaan opsi-opsi serta parameternya.
- 17 • Keluaran dari perangkat lunak ini adalah opsi apa saja yang dipilih, serta parameter yang
18 diberikan (jika ada).
- 19 • Opsi pertama yang disediakan adalah `-a` atau `--args` yang merupakan opsi tidak berargumen.
- 20 • Opsi kedua yang disediakan adalah `-n` atau `--noargs`, yang merupakan opsi yang membu-
21 tuhkan sebuah argumen.
- 22 • Opsi ketiga dan keempat merupakan penanda mode keluaran, yaitu `--short` dan `--long`.
23 Jika opsi `--long` digunakan, maka perangkat lunak ini akan mengeluarkan versi panjang dari
24 keluaran, sedangkan jika opsi `--short` digunakan, maka perangkat akan mengeluarkan versi
25 pendek dari keluaran.

Kode 2.2: Contoh sederhana penggunaan getopt-long

```

27 1 #include <stdio.h>
28 2 #include <stdlib.h>
29 3 #include <getopt.h>
30 4
31 5 int main(int argc, char **argv) {
32 6     int option;
33 7     static int verbose;
34 8
35 9     while (1) {
36 10         static struct option long_options[] = {
37 11             {"long", 0, &verbose, 1},
38 12             {"short", 0, &verbose, 0},
39 13             {"noargs", 0, 0, 'n'},
40 14             {"args", 1, 0, 'a'},
41 15             {0, 0, 0, 0}};
42 16         int option_index = 0;
43 17         option = getopt_long(argc, argv, "na:", long_options, &option_index);
44 18
45 19         if (option == -1)
46 20             break;
47 21         switch (option) {
48 22             case 0:
49 23                 if (verbose) {
50 24                     printf("Print_mode_is_set_to:_%s", long_options[option_index].name);
51 25                 }
52 26             else
53 27                 printf("Print_mode_is_set_to:_%s", long_options[option_index].name);
54 28                 putchar('\n');
55 29             break;
56 30
57 31         case 'n':
58 32     }
59 33 }
```

```

B2      if (verbose == 1) {
B3          printf("Option '%s' was_picked_. This_option_does_not_require_any_arguments.", long_options[option_index].name)
B4          ;
B5      }
B6      else
B7          printf("Argumentless_option '%s' was_picked.", long_options[option_index].name);
B8          putchar('\n');
B9          break;
B10
B11      case 'a':
B12          if (verbose == 1) {
B13              printf("Option '%s' was_picked_with_argument '%s'.", long_options[option_index].name, optarg);
B14          }
B15          else
B16              printf("a=%s", optarg);
B17          putchar('\n');
B18          break;
B19
B20      case '?':
B21          break;
B22
B23      default:
B24          abort();
B25      }
B26
B27  if (optind < argc)
B28  {
B29      printf("Arguments_passed_without_a_corresponding_option_(argv):");
B30      while (optind < argc) {
B31          printf("%s", argv[optind++]);
B32      }
B33      putchar('\n');
B34  }
B35
B36  exit(0);
B37 }
```

39 2.3.2 libcurl [2]

40 libcurl merupakan sebuah *library* yang berisi fungsi-fungsi yang disediakan dalam bentuk API bahasa
 41 C, untuk digunakan oleh aplikasi-aplikasi bahasa C. Libcurl didesain dengan berorientasi transfer
 42 (biasanya transfer berkas), tanpa memerlukan para penggunaanya untuk mengerti protokol-protokol
 43 yang digunakan dalam proses pentransferan tersebut. Fitur transfer ini dapat dibuat sesederhana
 44 mungkin, dan seluruh aturan dan opsi seputar pemindahan berkas tersebut dapat diatur nantinya
 45 secara manual melalui opsi-opsi yang ada.

46 Untuk mulai melakukan transfer berkas, diperlukan juga sebuah *handle* yang perlu diinisialisasi
 47 terlebih dahulu. Adapun cURL memiliki dua jenis *handle*, yaitu *easy handle* dan *multi handle*.

48 *Easy Handle*

49 *Easy handle* dari cURL dapat diinisialisasi dengan memanggil fungsi `curl_easy_init()`. Setelah
 50 itu, untuk mengatur opsi-opsi yang perlu diatur sesuai kebutuhan pengguna, seperti URL yang
 51 dituju, protokol yang ingin dipakai, koneksi ke port spesifik, dan sebagainya,⁶ pengguna harus
 52 mengurnya dengan fungsi `curl_easy_setopt()`. *Handle* ini berhasil diatur opsinya apabila
 53 fungsi `curl_easy_setopt()` tadi mengembalikan `CURLE_OK`. Terakhir, untuk menjalankan transfer-
 54 nya, fungsi yang perlu dipanggil adalah `curl_easy_perform(<easy handle>)`, dengan variabel
 55 `<easy handle>` diisi dengan nama dari *easy handle* yang ingin dimulai transfernya.

56 *Handle* yang telah diatur ini dapat digunakan berulang kali dengan konfigurasi yang sama,
 57 sampai entah pengguna mengganti konfigurasi opsi-opsinya kembali, atau atau *handle*-nya direset

⁶https://curl.se/libcurl/c/curl_easy_setopt.html

1 dengan pemanggilan fungsi `curl_easy_reset()`.

2 ***Multi Handle***

3 *Multi handle* merupakan sebuah *handle* yang dapat memfasilitasi beberapa transfer yang dilakukan
 4 secara paralel. Metode pentransferan filenya masih sama dengan *easy handle*, hanya saja untuk
 5 *multi handle*, diperlukan sebuah *handle* tambahan yang dapat menampung seluruh *easy handle*
 6 yang akan digunakan. Adapun *handle* tipe ini dapat diinisialisasi dengan memanggil fungsi
 7 `curl_multi_init()`, dan untuk mengatur opsi-opsi seputar *multi handle* tersebut, pengguna dapat
 8 memanggil fungsi `curl_multi_setopt()`.

9 Untuk memulai transfer paralel, tentunya perlu diinisialisasi dulu masing-masing *easy handle*-nya.
 10 Setelah *handle-handle* tersebut diinisialisasi, *handle* tersebut dapat dimasukkan ke dalam sebuah
 11 *multi handle* dengan memanggil fungsi berikut.

12 `curl_multi_add_handle(<multi handle>, <easy handle>);`

13 Dengan `<easy handle>` merupakan nama dari *easy handle* yang ingin dimasukkan ke dalam *multi*
 14 *handle* tertentu dan `<multi handle>` merupakan nama dari *multi handle*-nya sendiri. Sedangkan,
 15 untuk menghapus sebuah *easy handle* dari dalam *multi handle*, dapat dipanggil fungsi berikut.

16 `curl_multi_remove_handle(<multi handle>, <easy handle>);`

17 Setelah seluruh *easy handle* yang ingin dijalankan dimasukkan ke dalam *multi handle*, *multi*
 18 *handle* tersebut dapat dijalankan dengan menggunakan sebuah *loop* transfer. Adapun isi dari loop
 19 ini meliputi tiga langkah utama, yaitu:

20 1. Inisialisasi variabel `transfers_running`

21 `transfers_running` merupakan sebuah variabel *integer* yang menjadi penanda bagi pengguna
 22 mengenai berapa banyak *handle* yang sedang melakukan proses transfer di suatu waktu. Selama
 23 nilai variabel ini bukan 0, artinya ada *easy handle* yang belum selesai melakukan proses transfer
 24 berkas.

25 2. Menjalankan transfer dalam *multi handle*

26 Langkah selanjutnya adalah menjalankan transfer dalam *mutli handle*, dengan memanggil
 27 fungsi `curl_multi_perform`. Adapun fungsi tersebut harus dipanggil dengan format berikut,
 28 yaitu:

29 `curl_multi_perform(<multi handle>, <transfers_running>)`

30 3. Menunggu transfer untuk selesai sebelum data diekstraksi

31 Tentunya sebelum data hasil transfer dapat diekstraksi untuk dipakai, proses transfernya sendiri
 32 harus selesai terlebih dahulu—untuk kasus dalam *multi handle* libcurl, seluruh *handle* di dalam
 33 *multi handle* harus selesai melakukan transfer terlebih dahulu, atau sampai terjadi *timeout*.
 34 Adapun langkah ini dapat diselesaikan entah dengan menggunakan `curl_multi_wait()`
 35 atau `curl_multi_poll()`, atau dengan cara manual, yaitu dengan memasukkan deskriptor-
 36 deskriptor file serta nilai *timeout* secara manual, dan kemudian menggunakan `select()`,
 37 walaupun metode ini tidak dianjurkan karena keterbatasan jumlah deskriptor yang dapat
 38 digunakan.⁷

⁷https://curl.se/libcurl/c/curl_multi_poll.html

- 1 Implementasi singkat dari ketiga langkah ini dapat dilihat di potongan kode berikut.

Kode 2.3: Loop sederhana dari penggunaan *multi handle curl*

```

2 31 do {
3 32     curl_multi_wait (multi_handle, NULL, 0, 1000, NULL);
3 33     curl_multi_perform (multi_handle, &transfers_running);
3 34 } while (transfers_running);

```

8 Multi Socket Handle

9 Ada mode lain dari *multi handle*, yaitu *multi socket handle*. Bedanya dengan *multi handle* biasa
10 adalah *multi socket handle* memperhatikan *socket-socket* yang ada dan memberitahu *handle-handle*
11 jika ada *socket* yang sudah siap untuk digunakan dalam proses *read/write*. Cara operasinya hampir
12 sama dengan *multi handle*—hal utama yang berbeda adalah dengan *multi socket handle*, diperlukan
13 satu buah parameter tambahan, yaitu kumpulan *socket* yang ingin diperhatikan.

14 *Socket* ini, beserta apa aksi yang ingin ditunggu dalam *socket* tersebut, diperhatikan dengan
15 implementasi sebuah fungsi *callback*, yaitu *socket_callback()*. Handle mana yang ingin digunakan,
16 *socket* mana yang ingin diperhatikan, serta aksi apa yang ditunggu diatur dalam fungsi ini. Jika
17 ada beberapa *socket* yang ingin diperhatikan, fungsi ini harus dipanggil lagi untuk setiap *socket*-nya.
18 Selain itu, perlu juga dipanggil fungsi *timer_callback()*, yang berfungsi untuk mengatur seberapa
19 lama aplikasi akan menunggu *socket*, sebelum terjadi *timeout*. Kedua *callback* ini dapat diatur ke
20 dalam suatu *multi handle* dengan menggunakan fungsi *curl_multi_setopt()* biasa—untuk *callback*
21 *socket*, fungsi ini ditandai dengan menggunakan parameter *CURLMOPT_SOCKETFUNCTION*, sedangkan
22 untuk *callback timer*, fungsi tersebut ditandai dengan parameter *CURLMOPT_TIMERFUNCTION*.

23 Adapun seluruh *handle* ini dapat dipanggil dengan memanggil fungsi *curl_multi_socket_action()*,
24 dan cara untuk melihat apakah seluruh transfer sudah selesai atau masih ada transfer yang berlangs-
25 sung pada suatu waktu sama dengan *multi transfer* biasa.

26 Contoh implementasi singkat dari seluruh fungsi-fungsi tersebut dapat dilihat di potongan kode
27 berikut.

Kode 2.4: Kumpulan implementasi penggunaan *multi socket handle curl*

```

28 /* socket callback */
29 30 int socket_callback(CURL *easy,      /* easy handle */
30 31     curl_socket_t s, /* socket */
30 32     int what,        /* aksi apa yang ditunggu */
30 33     void *userp,     /* penunjuk callback privat */
30 34     void *socktp)    /* penunjuk socket privat */
30 35
30 36 curl_multi_setopt(multi_handle, CURLMOPT_SOCKETFUNCTION, socket_callback);
30 37
30 38 /* timer callback */
30 39 int timer_callback(multi_handle, /* multi handle */
30 40     timeout_ms,   /* waktu tunggu dalam milidetik */
30 41     userp)        /* penunjuk callback privat */
30 42
30 43 curl_multi_setopt(multi_handle, CURLMOPT_TIMERFUNCTION, timer_callback);
30 44
30 45 /* multi socket action */
30 46 curl_multi_socket_action(multi, CURL_SOCKET_TIMEOUT, 0, &running);
30 47
30 48

```

48 2.3.3 cJSON⁸

⁸<https://github.com/DaveGamble/cJSON>

- 1 cJSON merupakan sebuah *library* yang berfungsi sebagai *parser* JSON untuk perangkat-perangkat
- 2 lunak bahasa C. *Library* ini sendiri terdiri atas sebuah file C dan sebuah file header.

3 Instalasi

- 4 cJSON dapat diinstal dengan beberapa cara, yaitu:

- Manual

Instalasi manual hanya membutuhkan pengembang perangkat lunak untuk menyalin kedua file *library* cJSON ke dalam direktori perangkat lunak tersebut.

- CMake

Untuk penggunaan cJSON dengan CMake, perlu dibuat sebuah direktori bernama `build`, dan kemudian CMake harus dijalankan di dalam direktori tersebut, dengan mengeksekusi perintah `cmake`. Dengan melakukan ini, Makefile akan dibuat di dalam direktori tersebut, yang nantinya akan dapat di-*compile* dan diinstal dengan perintah `make install`, atau `cmake --install`. Selain file-file *header* dan *library*, proses ini juga akan menginstal file-file untuk `pkg-config`, untuk memudahkan deteksi instalasi CMake sebelumnya.

Proses pembangunan cJSON juga memiliki beberapa opsi yang dapat diatur sedemikian rupa sesuai dengan kebutuhan pembuat perangkat lunak. Adapun opsi-opsi yang dapat diatur dapat dilihat di daftar berikut.

- `-DENABLE_CJSON_TEST`

Nilai awal: On

Jika opsi ini dinyalakan (diberi nilai “On”), maka tes-tes cJSON akan dibuat dan dijalankan bersamaan dengan instalasi.

- `-DENABLE_CJSON_UTILS`

Nilai awal: Off

Jika opsi ini dinyalakan, maka file-file utilitas cJSON akan diinstal bersamaan dengan proses instalasi utama.

- `-DENABLE_TARGET_EXPORT`

Nilai awal: On

Mengekspor target-target ekspor cJSON. Opsi ini dapat dimatikan jika terjadi masalah saat instalasi.

- `-DENABLE_CUSTOM_COMPILER_FLAGS`

Nilai awal: On

Mengaktifkan properti-properti untuk *compiler* non-standar (Clang, GCC, MSVC). Opsi ini dapat dimatikan jika terjadi masalah saat instalasi.

- `-DENABLE_VALGRIND`

Nilai awal: Off

Menjalankan tes-tes yang ada menggunakan Valgrind.

- `-DENABLE_SANITIZERS`

Nilai awal: Off

Meng-*compile* cJSON dengan menyalakan AddressSanitizer dan UndefinedBehaviorSanitizer.

- `-DENABLE_SAFE_STACK`

Nilai awal: Off

- 1 Menyalakan SafeStack. Pada saat skripsi ini dibuat, fitur ini hanya didukung untuk
 2 *compiler* Clang.
- 3 – **-DBUILD_SHARED_LIBS**
 4 **Nilai awal:** Off
 5 Membangun semua *shared library* yang tersedia.
- 6 – **-DBUILD_SHARED_AND_STATIC_LIBS**
 7 **Nilai awal:** On
 8 Membangun *shared library* dan *static library* yang tersedia.
- 9 – **-DCMAKE_INSTALL_PREFIX**
 10 **Nilai awal:** -
 11 Mengatur *prefix* direktori tempat instalasi cJSON.
 12 – **-DENABLE_LOCALES**
 13 **Nilai awal:** On
 14 Memungkinkan penggunaan metode *localeconv*.
 15 – **-DCJSON_OVERRIDE_BUILD_SHARED_LIBS**
 16 **Nilai awal:** On
 17 Memungkinkan penimpaan nilai dari opsi **-BUILD_SHARED_LIBS** menggunakan nilai dari
 18 opsi **-DCJSON_BUILD_SHARED_LIBS**.
 19 – **-DENABLE_CJSON_VERSION_SO**
 20 **Nilai awal:** On
 21 Menyalakan versi so dari cJSON.
- 22 • Makefile
 23 Jika CMake tidak tersedia, cJSON juga dapat dibangun dengan menggunakan GNU Make,
 24 dengan menggunakan perintah **make all**, dan menginstal *library-library* yang sudah ter-
 25 *compile* dengan perintah **make install**. Akan tetapi, perlu diingat bahwa metode instalasi
 26 ini sudah tidak lagi diperbarui (*deprecated*), dan dukungannya hanya sebatas pembetulan *bug*.
 - 27 • vcpkg
 28 Melalui vcpkg, cJSON dapat diunduh dan diinstal secara langsung. Versi vcpkg dari cJSON
 29 terus diperbarui oleh tim Microsoft dan kontributor-kontributor dari komunitas publik, jadi
 30 *library* cJSON yang diinstal melalui vcpkg kemungkinan besar akan selalu merupakan versi
 31 terbarunya.

32 Penggunaan

33 Jika cJSON diinstal melalui CMake atau Makefile, cJSON dapat digunakan dengan mengikutkan
 34 baris ini dalam kode program:

```
35 #include <cjson/cJSON.h>
```

36 Struktur Data

37 cJSON merepresentasikan sebuah nilai JSON dengan struktur data **cJSON**, yang dapat dilihat di
 38 potongan kode 2.5.

```

1 1 typedef struct cJSON
2 2 {
3 3     struct cJSON *next;
4 4     struct cJSON *prev;
5 5     struct cJSON *child;
6 6     int type;
7 7     char *valuestring;
8 8     int valueint;
9 9     double valuedouble;
10 10    char *string;
11 11 } cJSON;
12 12

```

14 Dengan variabel-variabel dalam struktur tersebut sebagai berikut:

- 15 • **next** dan **prev**

16 Variabel ini merupakan penunjuk ke struktur cJSON lainnya yang merupakan nilai JSON
17 selanjutnya (untuk **next**) dan sebelumnya (untuk **prev**).

- 18 • **child**

19 Variabel ini merupakan penunjuk ke struktur cJSON lainnya yang merupakan elemen JSON
20 di dalam struktur cJSON tersebut.

- 21 • **type**

22 Variabel ini menandakan tipe dari nilai JSON yang terdapat di dalam struktur cJSON tersebut.
23 Adapun tipe-tipe yang mungkin adalah sebagai berikut.

- 24 – **cJSON_Invalid**

25 Merepresentasikan sebuah objek yang tidak valid dan tidak bernilai. Jika seluruh *field*
26 diatur sehingga panjangnya 0 *byte*, maka variabel **type** akan otomatis berisi tipe ini.

- 27 – **cJSON_True**

28 Merepresentasikan nilai boolean *true*.

- 29 – **cJSON_False**

30 Merepresentasikan nilai boolean *false*.

- 31 – **cJSON_Number**

32 Merepresentasikan nilai numerik apapun. Jika nilainya merupakan nilai *double*, maka
33 nilai tersebut akan disimpan di dalam variabel **valuedouble**. Sedangkan jika nilainya
34 merupakan nilai *integer*, maka nilai tersebut akan disimpan di dalam variabel **valueint**.

- 35 – **cJSON_String**

36 Merepresentasikan nilai *string* apapun. Nilainya disimpan sebagai *string* yang dipisah
37 dengan *null terminator* ('\0' atau \u0000).

- 38 – **cJSON_Array**

39 Merepresentasikan nilai *array*. *Array* dalam cJSON diimplementasikan dengan menun-
40 jukkan isi dari variabel **child** tadi ke sebuah *linked list* dari objek-objek cJSON. Jika
41 objek cJSON ini merupakan elemen dalam sebuah *array*, maka isi dari variabel **prev** dan
42 **next** merupakan salah satu dari elemen-elemen lain dalam *array* yang menjadi elemen
43 sebelum dan sesudah elemen ini.

- 44 – **cJSON_Object**

45 Merepresentasikan nilai objek general. Implementasinya sama dengan *array*, hanya saja
46 untuk objek general, kuncinya diletakkan di dalam variabel **string**.

- 47 – **cJSON_Raw**

48 Merepresentasikan objek JSON apapun yang disimpan dalam bentuk *array* yang ditermi-
49 nisi/dipisah oleh *null terminator*.

1 – `cJSON_NULL`

2 Merepresentasikan sebuah nilai *null*.

3 • `valuestring/valuestring/valuestring/string`

4 Merupakan variabel-variabel yang menyimpan nilai dari struktur cJSON. Variabel apa yang
5 diisi dan apa isinya tergantung dari tipe data yang disimpan.

6 **Fungsi-Fungsi Dasar**

7 • Tipe-tipe data dasar

8 Untuk setiap tipe data cJSON, ada sebuah fungsi `cJSON_Create`.... Semua fungsi tersebut akan
9 mengalokasikan sebuah struktur cJSON yang nantinya dapat dihapus dengan `cJSON_Delete`.
10 Perlu diingat juga bahwa seluruh struktur yang dibuat harus dihapus agar tidak terjadi kebo-
11 coran memori. Adapun fungsi-fungsi `cJSON_Create` yang dapat digunakan untuk tipe-tipe
12 data yang tersedia adalah sebagai berikut.

13 – *null* dibuat dengan `cJSON_CreateNull`.

14 – *boolean* dibuat dengan `cJSON_CreateBool`, atau jika ingin membuat data *boolean* langsung dengan nilainya, dapat menggunakan `cJSON_CreateTrue` atau `cJSON_CreateFalse`.

15 – Bilangan apapun dibuat dengan `cJSON_CreateNumber`. Penggunaan fungsi ini akan langsung mengisi nilai variabel `valueint` dan `valuedouble`.

16 – *string* apapun dibuat dengan `cJSON_CreateString` (membuat sebuah string langsung sebagai nilai data JSON), atau `cJSON_CreateStringReference` (mereferensikan *string* yang sudah ada sebagai nilai data JSON).

21 • *Array*

22 Sebuah *array* kosong dapat dibuat dengan menggunakan fungsi `cJSON_CreateArray`. Selain fungsi tersebut, pembuatan *array* juga dapat dilakukan dengan memanggil sebuah fungsi alternatif, yaitu `cJSON_CreateArrayReference`. Bedanya adalah dengan fungsi alternatif ini, *array* yang dibuat tidak secara langsung “mengandung” nilai-nilainya, jadi ketika *array* tersebut dihapus dengan `cJSON_Delete`, nilai-nilai di dalamnya tidak terhapus juga. Hal yang sama juga dapat dilakukan dengan objek-objek di dalam *array* tersebut, dimana objek ini dapat ditambahkan dengan fungsi `cJSON.AddItemToArray`, yang akan langsung menambahkan objek tersebut ke dalam *array*-nya, atau dengan menggunakan `cJSON_AddItemReferenceToArray`, yang hanya akan menambahkan referensi dari objek yang ingin ditambahkan ke dalam *array*.

31 Jika ada sebuah objek yang ingin dihapus dari *array* tertentu, perangkat lunak dapat memanggil fungsi `cJSON_DetachItemFromArray`. Fungsi ini akan mengembalikan objek yang dihapus dari *array* tadi, jadi objek ini harus diberikan ke sebuah penunjuk lainnya (dimasukkan ke dalam variabel lain) agar tidak terjadi kebocoran memori. Selain fungsi tersebut, fungsi `cJSON_DeleteItemFromArray` juga dapat digunakan—bedanya adalah objek yang dihapus dari *array* tadi, jika dihapus menggunakan fungsi ini, akan langsung dihapus, seakan-akan fungsi `cJSON_Delete` dipanggil untuk objek tersebut.

38 Untuk menimpa/mengganti nilai suatu objek di dalam *array*, fungsi `cJSON_ReplaceItemIn`
39 *Array* dapat dipanggil dengan indeks dari objek yang ingin diganti sebagai parameternya.
40 Selain fungsi tersebut, fungsi `cJSON_ReplaceItemViaPointer` juga dapat digunakan—fungsi
41 ini bekerja dengan memutuskan (*detach*) objek lama yang ingin diganti, menghapus objek

1 tersebut, dan menyisipkan objek yang baru ke tempat objek lama yang sudah dihapus tadi.

2 Terakhir, untuk mendapatkan objek tertentu dalam *array* berdasarkan indeksnya, perangkat
3 lunak dapat memanggil fungsi `cJSON_GetArrayItem`. Ukuran dari *array*-nya sendiri juga
4 dapat dilihat dengan fungsi `cJSON_GetArraySize`.

5 • Objek

6 Sama seperti *array*, ada dua jenis fungsi pembuatan objek. Yang pertama adalah `cJSON_Create`
7 `Object` untuk membuat objek biasa, dan `cJSON_CreateObjectReference` untuk membuat
8 objek yang nilai-nilainya terdiri atas kumpulan referensi ke variabel-variabel lain.

9 Untuk menambahkan sebuah objek ke dalam suatu objek lainnya, pengguna dapat
10 memanggil fungsi `cJSON.AddItemToObject`. Jika objek ingin ditambahkan ke suatu objek
11 lain yang memiliki sebuah variabel konstan sebagai nama, atau sebuah referensi, prosesnya
12 harus menggunakan fungsi `cJSON.AddItemToObjectCS`.

13 Fungsi `cJSON_DetachItemFromObjectCaseSensitive` dapat dipanggil ketika ada sebuah
14 objek ingin dibuang dari objek lain. Sama seperti fungsi *detach* di *array* tadi, fungsi ini akan
15 mengembalikan objek yang dibuang tadi, dan objek tersebut harus dimasukkan ke variabel lain
16 untuk menghindari kebocoran memori. Selain itu, ada juga fungsi `cJSON_DeleteItemFrom`
17 `ObjectCaseSensitive`, yang bekerja dengan cara yang sama seperti fungsi penghapusan objek
18 untuk *array*.

19 Untuk pengeditan/penggantian nilai objek, lagi-lagi cara kerjanya sama dengan *array*.
20 Untuk penggantian nilai objek berdasarkan kuncinya, fungsi `cJSON_ReplaceItemInObject`
21 `CaseSensitive` dapat digunakan, sedangkan untuk penggantian objek langsung dengan
22 penunjuk ke elemen lainnya, fungsi `cJSON_ReplaceItemViaPointer` dapat digunakan.

23 Terakhir, untuk mengakses sebuah benda di dalam objek, pengguna dapat memanggil fungsi
24 `cJSON_GetObjectItemCaseSensitive`, dan untuk mengetahui ukuran dari objek tersebut,
25 fungsi yang dapat digunakan sama dengan fungsi yang dapat digunakan untuk *array*, yaitu
26 `cJSON_GetArraySize`.

27 • *Parsing*

28 Objek JSON yang dibatasi/diterminasi dengan *null terminator* dapat di-parse dengan
29 menggunakan fungsi berikut.

30 `cJSON_Parse(<JSON>)`

31 Sedangkan, jika objek JSON tersebut tidak (atau belum tentu) dibatasi oleh *null terminator*,
32 objek tersebut dapat di-parse dengan fungsi berikut.

33 `cJSON_Parse(<JSON>, <ukuran JSON yang ingin di-parse>)`

34 Kedua fungsi ini akan membuat sebuah struktur hierarkial dari objek-objek cJSON yang
35 merepresentasikan keseluruhan dari objek JSON tersebut. Setelah objek ini selesai digunakan,
36 penggunanya harus mendealokasikan objek tersebut dengan fungsi `cJSON_Delete`.

37 **2.3.4 CMake [3]**

38 CMake merupakan sebuah perkakas generator *build system* yang memungkinkan pengembang
39 perangkat lunak untuk menentukan parameter-parameter pembangunan perangkat di sebuah file

yang berupa teks biasa. File ini kemudian dipakai oleh CMake untuk membuat perkakas-perkakas pembangunan perangkat lunak yang dapat dibaca oleh IDE-IDE tertentu, seperti Microsoft Visual Studio, Apple Xcode, Linux, dan sebagainya. Selain itu, CMake juga menangani aspek-aspek rumit dari pembangunan perangkat lunak, seperti pembangunan perangkat lunak *cross-platform*, introspeksi sistem, dan juga pembangunan perangkat lunak yang dapat disesuaikan berdasarkan penggunanya.

Untuk proyek-proyek yang dibangun di satu platform, CMake memiliki fungsi sebagai berikut.

- Memberikan kemampuan untuk melakukan pencarian seluruh perangkat lunak, file-file *library*, dan file-file *header* yang dibutuhkan untuk proses pembangunan perangkat lunak. Pencarian ini juga dapat dilakukan sampai ke dalam *registry* sistem.
- Memungkinkan pembangunan perangkat lunak diluar folder tempat *source code* perangkat lunak tersebut sendiri.
- Memberikan kemampuan untuk membuat perintah-perintah khusus untuk file-file yang dibuat secara otomatis, seperti `moc()` dari Qt, atau generator pembungkus dari SWIG. Perintah-perintah ini dapat mengatur pembuatan file *source* baru yang nantinya akan diintegrasikan langsung ke dalam perangkat lunak akhirnya.
- Memberikan kemampuan untuk memilihkan file-file opsional dari *library* yang digunakan.
- Memungkinkan pengaturan pembuatan proyek dari sebuah file `.txt` sederhana.
- Kemampuan untuk dengan mudah beralih dari *static build* dan *shared build*.
- Membuat ketentuan keperluan (*dependency*) secara otomatis.

Sedangkan, untuk perangkat-perangkat lunak yang dibuat *cross-platform*, CMake memberikan fungsi-fungsi tambahan sebagai berikut.

- Memberikan kemampuan mengetes urutan *machine byte* dan karakteristik-karakteristik lainnya yang spesifik untuk suatu sistem operasi.
- File konfigurasi yang dibuat dapat berlaku untuk seluruh *platform*.
- Memberikan dukungan untuk membangun *shared build* untuk seluruh *platform* yang mendukungnya.
- Memberikan kemampuan untuk mengatur opsi-opsi yang spesifik untuk suatu sistem operasi, seperti misalnya lokasi file-file data utama.

Penggunaan Dasar

CMake mengambil satu (atau lebih) file-file CMakeLists sebagai masukan, dan sebagai keluaran akan menghasilkan file-file proyek atau Makefile yang dapat digunakan dengan dan oleh perkakas-perkakas pembangunan perangkat lunak lain yang ada.

Proses CMake umumnya terdiri atas langkah-langkah berikut.

1. Proyek yang ingin dibangun didefinisikan di salah satu file CMakeLists.

File-file CMakeLists (yang berupa file-file `.txt`) merupakan file-file *plain text* yang berisi deskripsi proyek dalam bahasa CMake. Tertera di kode 2.6 merupakan file CMakeLists yang dapat digunakan untuk membangun sebuah program “Hello World” sederhana dalam bahasa C, sebagai gambaran mengenai hal-hal apa saja yang minimal harus ada di dalam file CMakeLists.

```

1 cmake_minimum_required(VERSION 3.20)
2 project(Hello)
3 add_executable(Hello Hello.c)

```

Penjelasan dari baris-baris berikut adalah sebagai berikut.

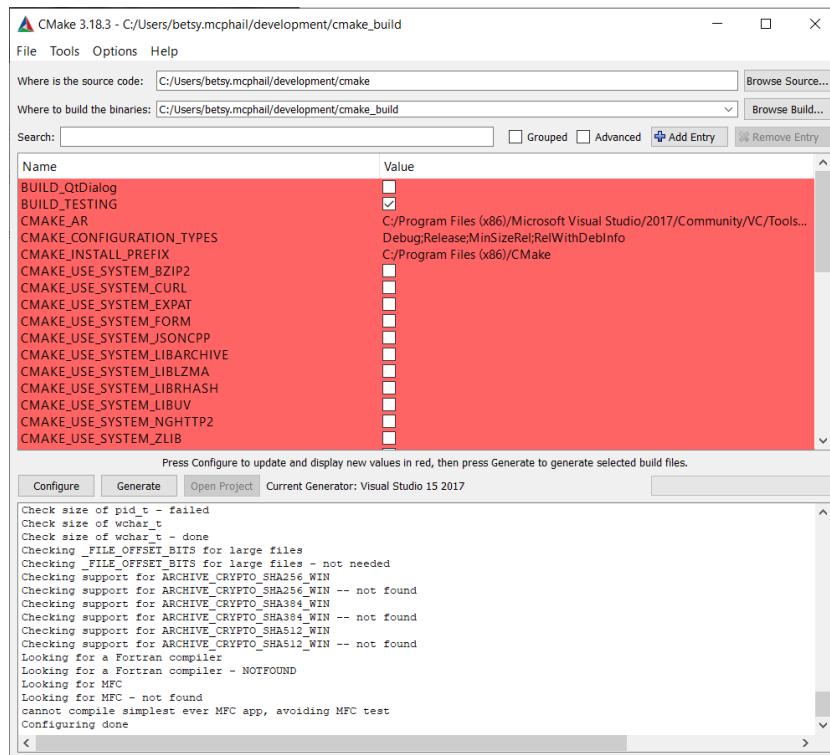
- Baris pertama harus selalu merupakan `cmake_minimum_required`. Hal ini mengharuskan proyek untuk menggunakan versi dari CMake yang dispesifikasi, dan juga memungkinkan *backwards compatibility*.
- Baris selanjutnya merupakan perintah `project`. Perintah ini mengatur nama proyek, dan juga bisa digunakan untuk mengatur aturan-aturan lainnya, seperti versi, atau bahasa dari proyek.
- Terakhir, perintah `add_executable` merupakan sebuah perintah yang akan menambahkan sebuah file *executable* untuk menjalankan proyek yang sudah dibangun tersebut.

2. CMake membuat dan mengkonfigurasi proyek tersebut.

Setelah file-file CMakeLists selesai dibuat, CMake akan memproses file-file tersebut dan membuat entri-entri dalam sebuah file *cache*. Pengembang perangkat lunak dapat mengedit file CMakeLists atau mengatur isi dari file *cache* tadi dengan GUI CMake, ccmake, atau untuk proyek-proyek skala kecil, langsung dari *command line*.

- GUI CMake

CMake memiliki perangkat lunak GUI berbasis Qt yang bisa digunakan di sebagian besar sistem operasi, seperti UNIX, Mac OS X, dan Windows. Perangkat lunak ini, `cmake-gui`, sudah terinstal bersama dengan CMake, tetapi membutuhkan instalasi Qt untuk dijalankan. Tampilan dari perangkat lunak ini dapat dilihat di gambar 2.10.



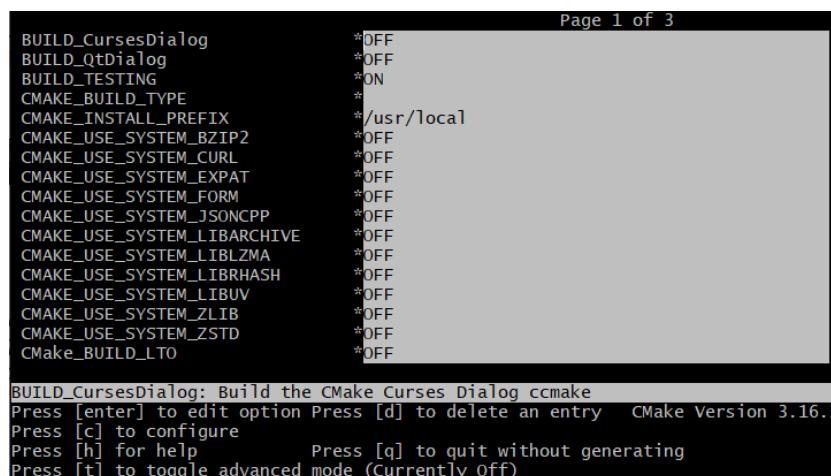
Gambar 2.10: Tampilan aplikasi `cmake-gui`.⁹

⁹<https://cmake.org/cmake/help/book/mastering-cmake/chapter/Getting%20Started.html>

1 Dua *field* paling atas adalah direktori dari *source code* dan direktori tempat file-file
 2 *binary* nantinya akan diletakkan setelah dibuat. Kedua *field* ini harus diisi secara
 3 manual, walaupun jika direktori *binary* ini sudah dikonfigurasi langsung melalui CMake
 4 sebelumnya, *field* direktori kedua akan secara otomatis terisi.

5 • **ccmake**

6 Di mayoritas sistem operasi berbasis UNIX, jika *library curses* didukung, maka CMake
 7 memiliki sebuah perangkat lunak lain yang dapat digunakan, yaitu **ccmake**. Untuk
 8 menjalankan **ccmake**, pengguna dapat menjalankannya melalui *command line*, dan
 9 direktori tempat **ccmake** dijalankan ini harus merupakan direktori tempat file-file *binary*
 10 nantinya ingin disimpan. Ketika aplikasi ini dijalankan, akan keluar tampilan seperti
 11 di gambar 2.11. Adapun instruksi singkat penggunaan dari aplikasi ini dapat dilihat di
 12 bagian bawah dari tampilan tersebut.



```

Page 1 of 3

BUILD_CursesDialog      *OFF
BUILD_qtDialog          *OFF
BUILD_TESTING           *ON
CMAKE_BUILD_TYPE        *
CMAKE_INSTALL_PREFIX    */usr/local
CMAKE_USE_SYSTEM_BZIP2  *OFF
CMAKE_USE_SYSTEM_CURL   *OFF
CMAKE_USE_SYSTEM_EXPAT  *OFF
CMAKE_USE_SYSTEM_FORM   *OFF
CMAKE_USE_SYSTEM_JSONCPP *OFF
CMAKE_USE_SYSTEM_LIBARCHIVE *OFF
CMAKE_USE_SYSTEM_LIBBLZMA *OFF
CMAKE_USE_SYSTEM_LIBRHASH *OFF
CMAKE_USE_SYSTEM_LIBUV   *OFF
CMAKE_USE_SYSTEM_ZLIB    *OFF
CMAKE_USE_SYSTEM_ZSTD    *OFF
CMake_BUILD_LTO          *OFF

BUILD_CursesDialog: Build the CMake Curses Dialog ccmake
Press [enter] to edit option Press [d] to delete an entry  CMake Version 3.16.2
Press [c] to configure
Press [h] for help             Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently off)

```

Gambar 2.11: Tampilan aplikasi **ccmake**.¹⁰

13 • Langsung dari *command line*

14 CMake juga dapat dijalankan melalui *command line*. Untuk menjalankan CMake dari
 15 *command line*, direktori tempat terminal berada lagi-lagi harus diatur ke direktori tempat
 16 file-file *binary* akan disimpan. Kemudian jalankan perintah **cmake** dengan opsi-opsi yang
 17 diinginkan, diikuti dengan direktori tempat *source code* dari perangkat lunak yang ingin
 18 dibangun berada. Walaupun begitu, perlu diingat bahwa metode ini direkomendasikan
 19 untuk digunakan hanya untuk proyek-proyek yang memiliki sedikit, atau bahkan tidak
 20 ada opsi sama sekali.

21 3. Pengguna membangun proyek tersebut dengan perkakas pembangunan masing-masing.

22 CMake dapat memberikan beberapa opsi dalam pembangunan perangkat lunak yang dapat
 23 diatur oleh penggunanya masing-masing. Adapun dua opsi utama dari seluruh opsi-opsi
 24 tersebut adalah sebagai berikut.

25 • Menspesifikasi *compiler* yang akan digunakan

26 Di beberapa sistem, bisa jadi terdapat lebih dari satu *compiler*, atau *compiler* yang ada
 27 tidak berada di tempat *compiler* tersebut biasa diinstal. Di kasus-kasus ini, CMake perlu
 28 diberitahu secara manual dimana letak *compiler* yang ingin digunakan. Ada tiga cara

¹⁰<https://cmake.org/cmake/help/book/mastering-cmake/chapter/Getting%20Started.html>

untuk melakukan ini, yaitu dengan:

- menspesifikasikan *compiler* yang ingin dipakai ke generator,
- mengatur variabel *environment*, atau
- membuat entri *cache*.

Jika pengaturan *compiler* sudah selesai dan *cmake* sudah dijalankan setidaknya sekali, jika pengguna ingin mengganti *compiler*, pengguna harus memulai ulang dengan folder file *binary* yang kosong.

- Mengatur konfigurasi pembangunan perangkat

Konfigurasi pembangunan perangkat memungkinkan sebuah proyek untuk dibangun dalam beberapa cara dengan tujuan *debugging*, optimisasi, atau tujuan-tujuan sejenis lainnya. Secara *default*, CMake mendukung mode-mode berikut:

- **Debug**—Opsi-opsi *debug* dasar dinyalakan.
- **Release**—Fungsi optimisasi dasar dinyalakan.
- **MinSizeRel**—Ukuran kode akhir diusahakan sekecil mungkin.
- **RelWithDebinfo**—Membangun *build* optimal dan mengikutkan informasi-informasi untuk *debugging*.

Dengan generator-generator berbasis Makefile, hanya sebuah mode konfigurasi yang bisa aktif ketika CMake sedang dijalankan, dan mode tersebut diatur dengan variabel **CMAKE_BUILD_TYPE**. Jika variabel ini dikosongkan (atau tidak dimasukkan ke dalam kode), maka mode yang dipakai adalah mode *default*. Di lain kasus, jika pengembang ingin membangun perangkat dalam dua mode yang berbeda, perintah untuk menjalankan CMake (dengan variabel mode konfigurasi masing-masing) harus dijalankan untuk setiap modenya. Misal, jika pengguna ingin membangun perangkat dengan mode **Debug** dan **Release** sekaligus, maka perintah untuk menjalankan CMake harus ditulis seperti dapat dilihat di potongan kode 2.7.

Kode 2.7: Contoh kode pembangunan CMake lebih dari satu mode

```
1 cmake ../<direktori> -DCMAKE_BUILD_TYPE=Debug  
2 cmake ../<direktori> -DCMAKE_BUILD_TYPE=Release
```

Setelah CMake dijalankan, proyek tersebut siap untuk dibangun. Jika generator yang dipilih merupakan generator berbasis Makefiles, maka proyek tersebut dapat dibangun dengan mengganti *working directory* ke lokasi file-file *binary*, dan kemudian menjalankan perintah **make** atau **cmake --install**. Jika generator yang dipilih merupakan sebuah IDE, maka proyek tersebut dapat dibangun secara biasa melalui IDE tersebut.

¹

BAB 3

²

ANALISIS

³ 3.1 Analisis Aplikasi Sejenis

⁴ Untuk pembuatan perangkat lunak dalam skripsi ini, ada empat buah perkakas *command line*
⁵ yang akan diamati sebagai aplikasi sejenis. Dua dari empat aplikasi pertama adalah *Chrome Web*
⁶ *Store Item Property CLI* dan *iTunes Search API*. Selain dua perkakas tersebut, ada dua perkakas
⁷ lainnya yang bisa digunakan sebagai referensi, tetapi tidak dapat dieksplorasi, karena kedua aplikasi
⁸ tersebut tidak berhasil dijalankan dengan sempurna, yaitu *Uber CLI* dan *Google Maps Direction*
⁹ CLI

¹⁰ 3.1.1 *Chrome Web Store Item Property CLI*¹

¹¹ Perkakas *command line* ini merupakan ekstensi dari sebuah aplikasi lain yang memiliki fungsi
¹² yang sama, yaitu *Chrome Web Store Item Property*.² Perangkat lunak *Chrome Web Store Item*
¹³ *Property CLI* ini merupakan perangkat lunak yang akan memanggil fungsi API untuk mendapatkan
¹⁴ metadata dari sebuah ekstensi pada *web store* peramban Google Chrome. Perbedaan dari perkakas
¹⁵ ini dengan aplikasi dasarnya adalah bahwa perkakas ini dapat digunakan sebagai perkakas *command*
¹⁶ *line*, sedangkan aplikasi dasarnya hanya bisa digunakan dalam perangkat lunak lainnya sebagai
¹⁷ pemanggil fungsi API.

¹⁸ Penggunaan

¹⁹ Perkakas ini dapat digunakan melalui *command prompt* dengan cara mengetikkan perintah sebagai
²⁰ berikut.

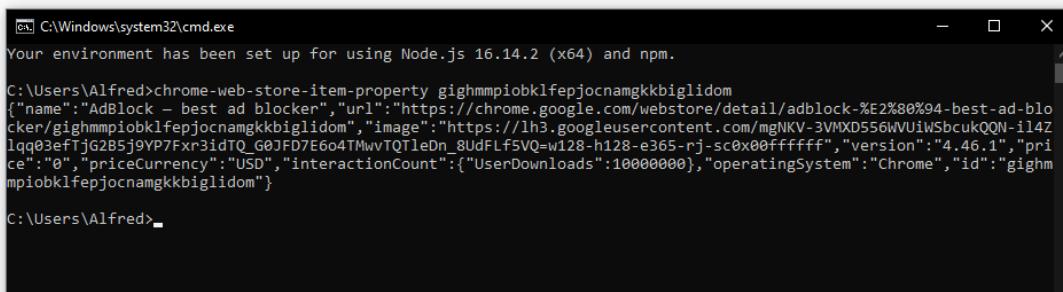
²¹ `chrome-web-store-item-property <identifier>`

²² Dengan *identifier* berupa ID dari ekstensi yang diinginkan. Jadi, misalkan pengguna mema-
²³ sukkan `gighmmypiobklfepjocnamgkkbiglidom` sebagai ID yang akan digunakan sebagai *identifier*,
²⁴ maka perkakas ini akan mengembalikan metadata dari ekstensi “AdBlock” sebagai keluarannya.
²⁵ Contoh penggunaan perkakas ini dapat dilihat di gambar 3.1.

²⁶ Sedangkan, keluaran dari perkakas ini merupakan sebuah objek JSON dengan properti-properti
²⁷ sebagai berikut.

¹<https://github.com/pandawing/node-chrome-web-store-item-property-cli>

²<https://github.com/pandawing/node-chrome-web-store-item-property>



```
C:\Windows\system32\cmd.exe
Your environment has been set up for using Node.js 16.14.2 (x64) and npm.

C:\Users\Alfred>chrome-web-store-item-property gighmmpioblkflepjocnamgkkbiglidom
{"name":"Adblock – best ad blocker","url":"https://chrome.google.com/webstore/detail/adblock-%E2%80%94-best-ad-blocker/gighmmpioblkflepjocnamgkkbiglidom","image":"https://lh3.googleusercontent.com/mgNKV-3VMXD556WVUiWSbcukQQN-iI4Zlqq03efTjG2B5j9YP7Fxr3idTQ_G0JFD7E6o4TMwvTQ1leDn_8UDfLf5VQ-w128-h128-e365-rj-sc0x0xffffffff","version":"4.46.1","price":0,"priceCurrency":"USD","interactionCount":{"UserDownloads":10000000}, "operatingSystem":"Chrome", "id":"gighmmpioblkflepjocnamgkkbiglidom"}

C:\Users\Alfred>
```

Gambar 3.1: Contoh penggunaan perkakas *Chrome Web Store Item Property* CLI.

- **name**
Nama dari ekstensi yang dicari metadatanya.
- **url**
URL halaman web dari ekstensi yang dicari di *web store* Google Chrome.
- **image**
Logo (dan ikon *thumbnail*) dari ekstensi yang dicari metadatanya.
- **version**
Nomor versi dari ekstensi.
- **price**
Harga dari ekstensi. Jika ekstensi tidak memiliki harga yang perlu dibayarkan (gratis), properti ini akan bernilai 0.
- **priceCurrency**
Kode mata uang dari harga ekstensi. Jika ekstensi tidak memiliki harga yang perlu dibayarkan, properti ini akan berisi “USD”.
- **interactionCount**
Properti ini berisi interaksi-interaksi pengguna yang tercatat sebagai data di halaman *web store* ekstensi. Pada saat pembuatan skripsi ini, properti ini hanya memiliki satu buah subproperti, yaitu **userDownloads**, yang menandakan berapa kali ekstensi ini telah diunduh oleh pengguna di manapun.
- **operatingSystems**
Menandakan di peramban mana ekstensi versi ini dapat diinstal. Karena ekstensi-ekstensinya berada di *web store* Chrome,
- **ratingValue** (tidak digunakan lagi)
Peringkat yang diberikan oleh para pengguna ekstensi ini. Nilai dari properti ini berupa skala desimal dari 0.00 sampai dengan 5.00. Di versi terbaru dari perkakas ini, properti ini tidak lagi tersedia dalam keluarannya.
- **ratingCount** (tidak digunakan lagi)
Jumlah pengguna yang telah menilai/memberi peringkat ke ekstensi ini. Di versi terbaru dari perkakas ini, properti ini tidak lagi tersedia dalam keluarannya.
- **id**
Properti ini mengandung ID dari ekstensi tersebut. Nilai dari properti ini akan sama dengan ID yang digunakan sebagai parameter masukan perkakas.

¹ 3.1.2 iTunes Search API³

² Perkakas *command line* ini berfungsi untuk melakukan pencarian melalui API iTunes, sehingga
³ seakan-akan pengguna langsung melakukan pencarian di iTunes sendiri. Hasil pencarian yang
⁴ dilakukan termasuk judul lagu, nama artis, ataupun nama album, dan pengguna dapat memilih
⁵ secara spesifik objek apa yang ingin dicari.

⁶ Penggunaan

⁷ Perkakas ini dapat digunakan melalui *command prompt* dengan cara mengetikkan perintah sebagai
⁸ berikut.

```
9 itunes-search-api <input> [<options>]
```

¹⁰ Dengan **input** berupa nama dari objek yang dicari. Perkakas ini juga memiliki opsi yang masing-
¹¹ masing memiliki parameter tersendiri untuk mempersempit hasil pencarian. Adapun opsi-opsi
¹² tersebut dapat dilihat di daftar di bawah ini.

- ¹³ • **--country**

Kemungkinan nilai: Kode negara dua huruf

Opsi ini menerima parameter berupa kode negara asal dari album atau artis yang dicari.

- ¹⁴ • **--entity**

Kemungkinan nilai: song, musicArtist, atau album

Menandakan jenis entitas (objek) yang ingin dicari. Opsi ini dapat bernilai **song** untuk pencarian berbasis judul lagu, **musicArtist** untuk pencarian nama artis, atau **album** untuk pencarian nama album. Jika opsi ini tidak dipakai, objek apapun yang memiliki kemiripan dengan **input** dalam salah satu dari ketiga properti ini akan muncul dalam hasil pencarian.

- ¹⁵ • **--limit**

Kemungkinan nilai: Bilangan bulat positif⁴

Batas maksimal hasil pencarian yang ingin ditampilkan dalam keluaran.

²⁵ Sedangkan, keluaran dari perkakas ini merupakan sebuah objek JSON yang memiliki dua properti
²⁶ utama, yaitu:

- ²⁷ • **resultCount**

Properti ini berisi bilangan bulat yang menandakan berapa buah objek yang terdapat dalam hasil pencarian.

- ²⁸ • **results**

Array yang berisi kumpulan objek yang terdapat di dalam hasil pencarian. Objek-objek ini akan dikembalikan berupa sebuah *array* lain yang berisi seluruh properti dari masing-masing objek. Apa saja properti yang diikutkan dalam *array* tersebut tergantung tipe dari objek dalam hasil pencarian.

³⁵ Adapun contoh penggunaan dan hasil keluaran perkakas ini dapat dilihat di gambar 3.2.

³<https://github.com/awcross/itunes-search-api>

⁴Opsi ini juga menerima bilangan bulat negatif, tetapi menggunakan sebuah bilangan bulat negatif akan menghilangkan pengaruh opsi ini terhadap hasil keluaran.

```
C:\WINDOWS\system32\cmd.exe
Your environment has been set up for using Node.js 16.14.2 (x64) and npm.

C:\Users\Alfred>itunes-search-api "the presets" --entity musicArtist
{
  resultCount: 3,
  results: [
    {
      wrapperType: 'artist',
      artistType: 'Artist',
      artistName: 'The Presets',
      artistLinkUrl: 'https://music.apple.com/us/artist/the-presets/78745197?uo=4',
      artistId: 78745197,
      amgArtistId: 748856,
      primaryGenreName: 'Electronic',
      primaryGenreId: 7
    },
    {
      wrapperType: 'artist',
      artistType: 'Artist',
      artistName: 'Too Many Presets',
      artistLinkUrl: 'https://music.apple.com/us/artist/too-many-presets/1453488885?uo=4',
      artistId: 1453488885,
      primaryGenreName: 'House',
      primaryGenreId: 1048
    },
    {
      wrapperType: 'artist',
      artistType: 'Artist',
      artistName: 'All Presets',
      artistLinkUrl: 'https://music.apple.com/us/artist/all-presets/1513653882?uo=4',
      artistId: 1513653882,
      primaryGenreName: 'Electronic',
      primaryGenreId: 7
    }
  ]
}
```

Gambar 3.2: Contoh penggunaan perkakas *iTunes Search API*, untuk mencari artis bernama “*The Presets*”.

1 3.1.3 Uber CLI⁵

2 Uber CLI merupakan sebuah perkakas *command line* yang dapat digunakan untuk dua fungsi
3 utama. Fungsi pertama dari perkakas ini adalah untuk mendapatkan estimasi untuk seberapa lama
4 waktu yang diperlukan untuk servis taksi *online* dari Uber untuk mencapai lokasi yang ingin dituju,
5 sedangkan fungsi keduanya adalah untuk mengestimasi berapa harga yang harus dibayarkan untuk
6 memakai servis tersebut.

7
8 Fungsi yang pertama dapat dilakukan memanggil perintah dengan format sebagai berikut.

uber merupakan perintah dasar dari perkakas ini. **time** merupakan parameter yang menandakan bahwa pengguna ingin menggunakan servis prediksi waktu dari perkakas ini. Selain itu, pengguna harus memasukkan alamat yang ingin dituju sebagai parameter akhir dari perintah yang akan digunakan sebagai masukan. Jika sintaksnya sudah benar, perintah tersebut akan bisa diproses oleh perkakas dengan cara mengirimkan pesan hasil konversi perintah tersebut ke API Uber. Setelah pemrosesan pesan tersebut berhasil, perkakas ini akan menampilkan sebuah keluaran dengan format yang dapat dilihat di gambar 3.3. Perlu diperhatikan juga bahwa keluaran yang dihasilkan oleh perkakas ini akan meliputi seluruh jenis servis yang disediakan oleh Uber.

⁵ <https://github.com/jaebradley/uber-cli>

21:00 \$ uber time '25 first street cambridge ma'	
25 First St, Cambridge, MA 02141, USA	
 	
2 min. uberPOOL,uberX	
3 min. uberXL	
5 min. UberBLACK,uberSUV	
7 min. TAXI	

Gambar 3.3: Contoh penggunaan fitur prediksi waktu perjalanan untuk perkakas Uber CLI.⁶

- 1 Sedangkan, untuk memanggil fungsi kedua dari perkakas ini, pengguna dapat dilakukan dengan
- 2 memanggil perintah dengan format berikut.

3 uber price -s <alamat awal> -e <alamat akhir>

- 4 Untuk sintaks ini, **uber** memiliki fungsi yang sama dengan sintaks untuk fungsi pertama dari
 5 perkakas. **price** merupakan penanda untuk perkakas bahwa pengguna ingin menggunakan servis
 6 untuk mengetahui perkiraan harga layanan Uber. Selanjutnya, perkakas akan meminta dua buah
 7 opsi beserta parameternya masing-masing. Pertama, opsi **-s**, berarti *start*, yang akan meminta
 8 sebuah parameter berupa lokasi yang ingin dipakai sebagai lokasi awal perkiraan harga layanan
 9 Uber. Sedangkan opsi **-e**, berarti *end*, akan meminta sebuah parameter berupa lokasi yang ingin
 10 dipakai sebagai lokasi akhir jasa perkiraan harga.

11

- 12 Adapun keluaran dari fungsi kedua ini dapat dilihat di gambar 3.4. Sama seperti keluaran untuk
 13 fungsi pertamanya, keluaran untuk fungsi kedua perkakas ini juga meliputi seluruh jasa yang
 14 disediakan oleh Uber.

21:00 \$ uber price -s '25 first street cambridge ma' -e '114 line street somerville ma'

					☀ Surge☀
uberPOOL	\$3-\$6	1.57 mi.	8 min.		
uberX	\$6-\$9	1.57 mi.	8 min.		
uberXL	\$10-\$13	1.57 mi.	8 min.		
UberBLACK	\$15-\$20	1.57 mi.	8 min.		
uberSUV	\$22-\$28	1.57 mi.	8 min.		
	25 First St, Cambridge, MA 02141, USA				
 END	114 Line St, Somerville, MA 02143, USA				

Gambar 3.4: Contoh penggunaan fitur prediksi harga perjalanan untuk perkakas Uber CLI.⁷

⁷Gambar diambil dari <https://github.com/jaebradley/uber-cli>

1 Permasalahan

2 Seperti telah dijelaskan di awal bab ini, perkakas ini tidak dapat digunakan. Kesimpulan yang
 3 diambil oleh penulis mengenai alasan perkakas ini tidak dapat dijalankan adalah dikarenakan oleh
 4 penggunaan API dan modul-modul yang telah usang (*deprecated*). Kesimpulan ini diambil oleh
 5 penulis karena dua alasan utama. Pertama, pada awalnya, perkakas ini tidak dapat dijalankan
 6 karena API Google *Maps* yang dipakai mengandung baris kode berikut didalamnya.

7 `exports.placesAutoCompleteSessionToken = require('uuid/v4');`

8 Kode ini merupakan kode yang dipakai untuk mengambil *subpath* dari paket *uuid*, tetapi penggu-
 9 naannya sudah berubah untuk versi yang lebih barunya. Akan tetapi, setelah diganti baris tersebut
 10 ke penggunaan versi barunya pun, perkakas ini masih tetap tidak dapat dijalankan—sekarang
 11 perkakas ini justru mengembalikan sebuah error. Singkatnya, error tersebut menunjukkan bahwa
 12 perkakas mencoba untuk mengakses API Uber dengan menggunakan kredensial OAuth 2.0 yang
 13 hanya berlaku untuk versi sebelumnya dari API tersebut. Permasalahan ini merupakan permasalah-
 14 an yang juga ditemukan oleh beberapa pengguna lain, seperti tertera di halaman *GitHub Issues*
 15 dari repositori ini.⁸ Oleh karena hal ini tidak lagi merupakan masalah kode perangkat lunak, maka
 16 perkakas ini dianggap tidak dapat dipakai.

17 3.1.4 Google *Maps Direction* CLI⁹

18 Google *Maps Direction* CLI merupakan sebuah perkakas *command line* yang memiliki kegunaan
 19 yang mirip dengan KIRI, hanya saja perkakas ini tidak secara spesifik mengharuskan penggunaan
 20 angkot, atau transportasi umum lainnya. Singkatnya, perkakas ini memiliki fungsi seperti sebuah
 21 GPS. Untuk menggunakannya, pengguna harus memasukkan perintah dengan bentuk sebagai
 22 berikut.

23 `direction <lokasi awal> <lokasi akhir> -k <kunci API>`

24 Setelah pengguna memasukkan perintah tersebut dengan benar, perkakas ini akan mengirim
 25 permintaan ke API Google *Maps*, di mana jika prosesnya berhasil, keluarannya akan berupa
 26 langkah-langkah yang harus ditempuh untuk sampai ke lokasi akhir, beserta di jarak berapa langkah
 27 tersebut harus diambil, relatif terhadap langkah sebelumnya. Perintah untuk perkakas ini juga
 28 dapat mengikutkan kunci API Google *Maps* melalui opsi `-k` (atau `--key`) untuk ramalan kemacetan
 29 yang lebih akurat. Adapun penggunaan dari perkakas ini dapat dilihat di gambar 3.5.

30 Permasalahan

31 Seperti tertulis di awal bab ini, perkakas ini juga tidak bisa digunakan. Alasan perkakas ini tidak
 32 dapat digunakan lagi-lagi merupakan masalah teknikal, yaitu diperbaruiinya API Google *Maps*.
 33 Lebih spesifiknya, semenjak 2018, Google tidak lagi memperbolehkan penggunaan API Google *Maps*
 34 tanpa kunci API, yang sayangnya tidak hanya mendasari perkakas ini, tetapi juga kunci API ini

⁸<https://github.com/jaebradley/uber-cli/issues/87>

⁹<https://github.com/yujinlim/google-maps-direction-cli>

¹⁰Gambar diambil dari <https://github.com/yujinlim/google-maps-direction-cli>

```

Route
Bukit Damansara, Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia → Petronas Twin Tower, Kuala Lumpur City Centre, 50088 Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia
Duration
18 mins
Routes
Head northeast on Jalan Medan Setia 2 (81 m)
Turn right toward Jalan Medan Setia (28 m)
Merge onto Jalan Medan Setia (45 m)
Turn left onto Jalan Setia Murni 1 (0.3 km)
Turn left onto Jalan Setia Murni 3 (53 m)
Turn left onto Jalan BeringinGo through 1 roundabout (1.0 km)
Take the ramp on the right to Jalan Duta/Kuala Lumpur (28 m)
Merge onto Damansara Link/Lebuhraya SPRINT/Sistem Penyiaran Trafik Kuala Lumpur Barat/E23 (0.4 km)
Continue onto Jalan Semantan (0.9 km)
Take the exit on the right toward K. Lumpur/Jln. Parlimen/PWTC (0.6 km)
Merge onto Jalan Tuanku Abdul Halim (0.7 km)
Take the exit toward PWTC/Jln. Parlimen/Jln. Tun Razak/Kuching/Dataran Merdeka (0.5 km)
Keep right at the fork, follow signs for Jalan Parlimen/Dataran Merdeka/Merdeka Square/Taman Botani Perdana/Perdana Botanical Garden (0.5 km)
Turn left onto Jalan Parlimen (1.1 km)
Turn left onto Bulatan Data Onn (91 m)
Take the ramp to Jalan Kuching/Route 1 (0.3 km)
Slight left onto Jalan Kuching/Route 1 (0.1 km)
Continue straight to stay on Jalan Kuching/Route 1 (0.7 km)
Take the Jln. Sultan Ismail exit on the right toward Menara KL/KLCC/Ampang/E12 (0.2 km)
Keep right to continue toward Jalan Sultan Ismail (0.3 km)
Continue onto Jalan Sultan Ismail (1.0 km)
Slight left onto the ramp to Ampang (0.4 km)
Continue onto Lebuhraya Bertingkat Ampang - Kuala Lumpur/E12 (0.8 km)
Take exit 1202A-Jln. Tun Razak toward KLCC (0.5 km)
Merge onto Lebuhraya Bertingkat Ampang - Kuala Lumpur/E12Toll road (0.3 km)
Take the exit toward KLCCPartial toll road (0.6 km)

```

Gambar 3.5: Contoh penggunaan perkakas Google Maps Direction CLI.¹⁰

- 1 tidak bisa didapatkan tanpa membayarkan biaya tertentu. Oleh karena itu, perkakas ini dianggap
- 2 tidak bisa dijalankan.

3.2 Analisis API KIRI

- 4 API KIRI memiliki tiga buah layanan, yaitu *search place*, *routing*, dan *smart direction*. Di antara
- 5 ketiga layanan ini, perlu diingat bahwa *smart direction* merupakan sebuah layanan yang bekerja
- 6 dengan langsung membuka halaman web KIRI sendiri, sehingga layanan ini tidak akan digunakan
- 7 dalam pembuatan perkakas *command line* untuk skripsi ini.

3.2.1 Search Place

- 9 Layanan pertama dari dua layanan yang tersisa merupakan layanan *search place*. Layanan ini
- 10 merupakan layanan API KIRI yang berfungsi untuk mencari sebuah lokasi, beserta nilai *latitude*
- 11 dan *longitude*-nya, berdasarkan kata kunci yang diberikan oleh pengguna. Untuk memanfaatkan
- 12 layanan ini, sebuah permintaan GET harus dikirimkan ke alamat API KIRI. Adapun permintaan
- 13 tersebut akan memiliki parameter-parameter sebagai berikut.

- 14 • **version**

Parameter ini merupakan tanda bagi API untuk menggunakan protokol versi 2. Karena kemungkinan nilai untuk parameter ini hanya satu, maka parameter ini pasti bernilai 2.

- 17 • **mode**

Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna. Untuk penggunaan layanan *search place*, parameter ini diisi dengan **searchplace**.

- 20 • **region**

Parameter ini mengatur daerah mana tempat lokasi yang ingin dicari berada. Isi dari parameter ini akan diberikan oleh pengguna pada saat pemakaian perkakas.

1 • **querystring**

2 Parameter ini akan berisi kata kunci yang diberikan oleh pengguna untuk mencari lokasi yang
3 ingin ditemukan.

4 • **apikey**

5 Parameter ini akan berisi sebuah nilai kunci API yang sudah dibuat sebelumnya, seperti
6 dijelaskan di subbab 2.2.2, parameter ini akan bernilai 68C...97C.¹¹

7 3.2.2 ***Routing***

8 Layanan kedua dari API ini merupakan fungsi utama dari KIRI sendiri, yaitu pencarian rute dengan
9 menggunakan angkot. Layanan ini akan menerima dua buah lokasi yang ingin dijadikan lokasi
10 awal dan lokasi akhir, dan menunjukkan langkah-langkah yang harus ditempuh untuk menempuh
11 perjalanan tersebut, dengan memanfaatkan jasa angkot yang ada. Sama seperti layanan *search*
12 *place*, layanan ini juga digunakan dengan mengirimkan permintaan GET ke alamat API KIRI.

13 • **version**

14 Sama seperti pada layanan *search place*, parameter ini hanya dapat diisi dengan nilai 2.

15 • **mode**

16 Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna.
17 Untuk penggunaan layanan *routing*, parameter ini diisi dengan **findroute**.

18 • **locale**

19 Isi dari parameter ini akan ditentukan pada saat pemakaian perkakas.

20 • **start**

21 Parameter ini merupakan nilai *latitude* dan *longitude* dari titik awal perjalanan pengguna,
22 yang akan diberikan sebagai masukan pada saat pemakaian perkakas.

23 • **finish**

24 Parameter ini merupakan nilai *latitude* dan *longitude* dari titik akhir/tujuan perjalanan
25 pengguna, yang akan diberikan sebagai masukan pada saat pemakaian perkakas.

26 • **presentation**

27 Parameter ini hanya digunakan untuk fitur *backwards compatibility*. Jika parameter ini ingin
28 digunakan, maka isi dari parameter ini hanya ada satu kemungkinan, yaitu **desktop**. Jika
29 parameter ini tidak dipakai

30 • **apikey**

31 Sama seperti pada layanan *search place*, parameter ini akan berisi sebuah nilai kunci API
32 yang sudah dibuat sebelumnya, yaitu 68C...97C.

33 Layanan ini dapat digunakan sebagai lanjutan dari layanan *search place*, terutama karena fitur
34 pencarian rute hanya menerima nilai *latitude* dan *longitude* dari lokasi-lokasi yang akan digunakan
35 sebagai masukan, yang tidak hanya akan menyusahkan pengguna dalam memakai perkakas *command*
36 *line* ini, tetapi juga kedua nilai tersebut juga merupakan salah satu dari keluaran layanan pencarian
37 lokasi. Jadi, dengan menggunakan kedua layanan tersebut secara berlangsung, maka pengguna
38 hanya perlu mencari lokasi tersebut dengan kata-kata kunci, dan pada akhirnya pengguna akan
39 dapat menerima langkah-langkah yang harus ditempuh dalam rute sebagai keluaran akhir dari

¹¹Bagian tengah dari kunci API diganti dengan “...” untuk alasan keamanan.

1 perkakas.

2 **3.3 Analisis Perkakas yang Akan Dibuat**

3 Di bagian ini akan dibahas analisis hal-hal yang berhubungan dengan perkakas yang akan dibangun
4 sebagai proyek skripsi ini.

5 **3.3.1 Analisis Fitur Perkakas**

6 Pada skripsi ini, akan dibangun sebuah aplikasi berupa perkakas *command line* yang merupakan
7 ekstensi dari sebuah aplikasi berbasis web lain, yaitu KIRI. Perkakas ini akan menjadi ekstensi KIRI
8 dengan cara memungkinkan penggunaanya untuk mengakses fungsi-fungsi API KIRI dari *command*
9 *line* milik perangkat mereka masing-masing. Fungsi utama dari perkakas ini tentunya sama dengan
10 KIRI sendiri, yaitu membantu navigasi dari satu lokasi ke lokasi lain dengan menggunakan angkot.

11 Walaupun begitu, aplikasi ini akan memiliki dua fungsi utama yang berhubungan satu sama
12 lain, yaitu pencarian lokasi, dan penunjukkan rute dengan angkot.

- 13 • Pencarian lokasi

14 Pencarian lokasi merupakan fungsi utama pertama dari perkakas ini. Untuk fungsi ini,
15 masukan langsung dari pengguna yang akan diterima oleh perkakas adalah kata kunci dari
16 sebuah lokasi yang ingin dicari. Kemudian, perkakas akan memprosesnya melalui API KIRI,
17 lalu mengembalikan nama lokasi tersebut, serta nilai *latitude* dan *longitude*-nya, sebagai
18 keluaran akhir dari proses tersebut.

- 19 • Pencarian rute

20 Pencarian rute dengan angkot merupakan fungsi utama kedua, sekaligus fungsi umum dari
21 perkakas ini. Dalam fungsi ini, perkakas akan meminta masukan langsung pengguna berupa
22 nilai *latitude* dan *longitude* dari lokasi awal serta lokasi akhir dari perjalanan pengguna.
23 Setelah masukan ini berhasil diterima dan diproses, perkakas akan mengeluarkan keluaran
24 akhir berupa langkah-langkah yang harus diambil oleh pengguna untuk pergi dari lokasi awal
25 ke lokasi akhir, dengan memanfaatkan jasa angkot yang ada.

26 Selain itu, perkakas ini juga akan memiliki fitur-fitur standar yang harus dimiliki oleh perkakas-
27 perkakas *command line*, seperti dokumentasi singkat (bantuan penggunaan perkakas) sebagai salah
28 satu mode operasional perkakas, dan juga *man page* (khusus implementasi di Linux).

29 **Aturan Penamaan**

30 Sedangkan, untuk memenuhi fitur-fitur tersebut, perkakas akan perlu berbagai opsi yang masing-
31 masing merepresentasikan fitur (mode operasional) perkakas, serta variabel-variabel yang dibutuhkan
32 oleh API. Setelah dilakukan analisis perkakas-perkakas sejenis yang memerlukan opsi-opsi dalam
33 masukannya, dibuat seperangkat aturan yang mendasari penamaan dari opsi-opsi yang akan dimiliki
34 perkakas yang akan dibuat. Berikut merupakan aturan-aturan tersebut.

- 35 1. Nama pendek (satu huruf) opsi umumnya akan dinamakan sesuai dengan huruf pertama dari
36 fitur/variabel yang direpresentasikan oleh opsi tersebut.
- 37 2. Nama panjang dari opsi umumnya akan dinamakan sesuai dengan nama fitur/variabel yang
38 direpresentasikan oleh opsi tersebut.

- 1 3. Argumen dari opsi yang merepresentasikan fitur perkakas akan dinamakan sesuai dengan
- 2 nama dari mode operasionalnya masing-masing di API KIRI, kecuali `direct`, karena fitur ini
- 3 merupakan mode tambahan yang bukan merupakan fitur langsung dari API KIRI.
- 4 4. Nama-nama opsi, baik pendek maupun panjang, yang jika mengikuti ketiga aturan pertama
- 5 akan menghasilkan nama yang sama dengan nama opsi lain, akan dinamakan dengan nama
- 6 buatan yang artinya tidak terlalu jauh dengan variabel apa yang direpresentasikan olehnya.

7 Opsi-opsi Perkakas

8 Mengikuti aturan-aturan yang sudah dipaparkan di sub-subbab 3.3.1, berikut merupakan daftar
9 serta penjelasan singkat dari opsi-opsi yang akan disediakan dalam perkakas yang akan dibangun.

- 10 • `-h (--help)`

11 Perlu diingat juga bahwa aplikasi ini merupakan aplikasi *command line* murni, yang berarti
12 bahwa seluruh operasi dari aplikasi ini akan dilakukan tanpa bantuan gambar grafis apapun,
13 sehingga perintah untuk menunjukkan apa perintah-perintah dan opsi-opsi yang tersedia
14 merupakan sebuah keharusan. Hal ini merupakan fungsi satu-satunya dari perintah `-h` ini.

- 15 • `-m <mode> (--mode)`

16 Opsi ini merupakan opsi berparameter yang menentukan fitur (mode operasional perkakas)
17 mana yang ingin digunakan oleh pengguna. Adapun isi dari parameter `<mode>` yang dapat
18 digunakan adalah sebagai berikut:

- 19 – `searchplace`

20 Parameter ini akan menandakan bahwa pengguna ingin menggunakan fitur *search place*
21 dari perkakas. Untuk mode ini, perkakas akan menerima input dari pengguna dalam
22 bentuk parameter-parameter dari opsi-opsi tambahan. Adapun opsi-opsi tersebut adalah
23 sebagai berikut.

- 24 * `-r <region> (--region)`

25 Opsi ini merupakan opsi yang akan menerima parameter berupa kode area dari
26 lokasi yang ingin dicari.

- 27 * `-q <kata kunci> (--query)`

28 Opsi ini merupakan opsi yang akan menerima sebuah *string* sebagai parameternya.
29 *String* ini akan digunakan oleh perkakas sebagai kata kunci untuk pencarian lokasi
30 yang ingin ditemukan oleh pengguna.

- 31 * `-l <kode bahasa> (--locale)`

32 Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh
33 perkakas di keluarannya nanti. Opsi ini merupakan tambahan spesifik dari perkakas
34 ini, karena keluaran dari perkakas mengandung kata-kata yang bukan merupakan
35 nama/koordinat lokasi hasil. Walaupun begitu, perlu diperhatikan bahwa opsi ini
36 tidak akan mempengaruhi nama lokasi dalam keluaran.

- 37 – `findroute`

38 Parameter ini akan menandakan bahwa pengguna ingin menggunakan fitur *find route* dari
39 perkakas. Sama seperti mode `--search`, perkakas akan menerima input dari pengguna
40 dari parameter-parameter milik opsi-opsi tambahan. Adapun opsi-opsi tersebut adalah
41 sebagai berikut.

1 * **-s <lokasi awal> (--start)**

2 Opsi ini merupakan opsi yang akan menerima parameter berupa lokasi awal perjalan pengguna nantinya. Perlu diingat bahwa opsi ini hanya menerima masukan lokasi berupa nilai *latitude* dan *longitude* dari lokasi tersebut.

3 * **-f <lokasi akhir> (--finish)**

4 Opsi ini merupakan opsi yang akan menerima parameter berupa lokasi akhir perjalan pengguna nantinya. Sama seperti opsi sebelumnya, parameter ini juga hanya menerima masukan lokasi berupa nilai *latitude* dan *longitude*.

5 * **-l <kode bahasa> (--locale)**

6 Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh 7 perkakas di keluarannya nanti. Opsi ini merupakan opsi yang sama dengan opsi 8 yang digunakan dalam mode **searchplace**.

9 **- direct**

10 Parameter ini merupakan tambahan fitur spesifik untuk perkakas ini, yang merupakan 11 gabungan dari fitur pencarian lokasi dan pencarian rute. Fitur ini akan menggabungkan 12 kedua fitur tersebut dengan langkah-langkah berikut.

- 13 1. Perkakas akan menerima lokasi awal dan akhir berupa kata kunci dari pengguna
14 yang dimasukkan kedalam parameter dari opsi masing-masing.
- 15 2. Hasil pencarian kedua lokasi akan dikonfirmasi terlebih dahulu, apakah lokasi yang
16 ditemukan benar merupakan lokasi yang ingin digunakan oleh pengguna atau bukan.
- 17 3. Jika hasil tersebut benar, maka koordinat *latitude* dan *longitude* dari kedua lokasi
18 tersebut akan disimpan.
- 19 4. Koordinat *latitude* dan *longitude* dari kedua lokasi tersebut kemudian akan digunakan
20 sebagai masukan untuk langkah kedua dari fitur ini, yaitu pencarian rute.
- 21 5. Hasil dari pencarian rute akan ditampilkan sebagai keluaran akhir dari fitur ini.

22 Adapun parameter yang diperlukan untuk fitur ini merupakan gabungan dari opsi-opsi
23 kedua fitur sebelumnya, yaitu:

24 * **-S <region tempat lokasi awal berada> (--regstart)**

25 Opsi ini akan menerima parameter berupa kode area dari lokasi awal yang akan
26 digunakan.

27 * **-F <region tempat lokasi akhir berada> (--regfinish)**

28 Opsi ini akan menerima parameter berupa kode area dari lokasi akhir yang akan
29 digunakan.

30 * **-s <kata kunci untuk lokasi awal> (--start)**

31 Opsi ini akan menerima parameter berupa kata kunci untuk pencarian lokasi awal
32 yang akan digunakan.

33 * **-f <kata kunci untuk lokasi akhir> (--finish)**

34 Opsi ini akan menerima parameter berupa kata kunci untuk pencarian lokasi akhir
35 yang akan digunakan.

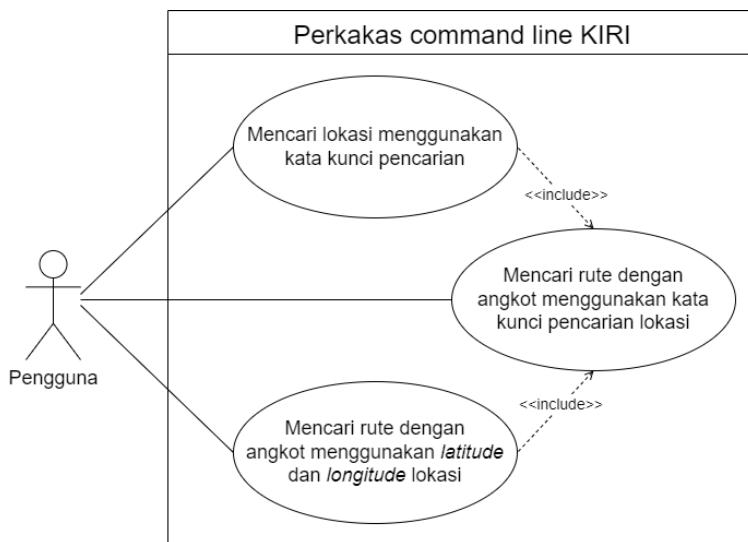
36 * **-l <kode bahasa> (--locale)**

37 Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh
38 perkakas di keluarannya nanti. Opsi ini merupakan opsi yang sama dengan opsi
39 yang digunakan dalam mode **searchplace**.

yang digunakan dalam kedua mode lainnya.

3.3.2 Analisis Use Case

Perkakas *command line* ini akan memiliki tiga fitur, yaitu pencarian lokasi dengan kata kunci, pencarian rute dengan koordinat *latitude* dan *longitude* lokasi, dan pencarian rute dengan kata kunci pencarian lokasi. Oleh karena itu, diagram *use case* dari aplikasi ini akan mengandung dua *use case*, yang dapat dilihat di gambar 3.6. Adapun penjelasan dari tiap-tiap *use case* tersebut dapat dilihat di tabel 3.1, tabel 3.2, dan tabel 3.3.



Gambar 3.6: Diagram *use case* dari perkakas yang akan dibangun.

Tabel 3.1: *Scenario case* untuk fitur pencarian lokasi dengan kata kunci pencarian.

Nama	Mencari lokasi menggunakan kata kunci pencarian
Deskripsi	Fitur untuk mencari lokasi di area tertentu berdasarkan kata kunci pencarian.
Aktor	Pengguna aplikasi
Pre-kondisi	-
Pos-kondisi	Nama dari lokasi serta koordinat <i>latitude</i> dan <i>longitude</i> -nya ditampilkan kepada pengguna.
Skenario utama	<ol style="list-style-type: none"> 1. Perkakas membaca masukan berupa kata kunci pencarian lokasi dari argumen dalam perintah <i>command line</i>. 2. Perkakas memproses masukan tersebut dan mengirimkannya ke API KIRI untuk diproses lebih lanjut. 3. Perkakas menerima keluaran berupa nama serta koordinat <i>latitude</i> dan <i>longitude</i> dari lokasi yang ditemukan. 4. Perkakas menampilkan kembali keluaran tersebut kepada pengguna.
Lanjut di halaman berikutnya...	

Tabel 3.1 – Lanjutan dari halaman sebelumnya

Skenario eksepsi	<ul style="list-style-type: none"> - Pengguna memasukkan masukan yang tidak valid sebagai kata kunci pencarian. - Lokasi tidak berhasil ditemukan.
-------------------------	--

Tabel 3.2: *Scenario case* untuk fitur pencarian rute dengan angkot, dengan nilai *latitude* dan *longitude* kedua lokasi sebagai masukan.

Nama	Mencari rute dengan angkot menggunakan <i>latitude</i> dan <i>longitude</i> lokasi
Deskripsi	Fitur untuk mencari cara pergi ke satu lokasi ke lokasi lainnya dengan menggunakan angkot dengan nilai <i>latitude</i> dan <i>longitude</i> lokasi sebagai masukan.
Aktor	Pengguna aplikasi
Pre-kondisi	-
Pos-kondisi	Langkah-langkah yang harus ditempuh untuk perjalanan tersebut akan ditampilkan kepada pengguna.
Skenario utama	<ol style="list-style-type: none"> 1. Perkakas membaca masukan berupa nilai <i>latitude</i> dan <i>longitude</i> lokasi awal dan akhir dari argumen dalam perintah <i>command line</i>. 2. Perkakas memproses masukan tersebut dan mengirimkannya ke API KIRI untuk diproses lebih lanjut. 3. Perkakas menerima keluaran akhir dari API KIRI berupa langkah-langkah dalam rute yang perlu ditempuh. 4. Perkakas menampilkan kembali keluaran tersebut kepada pengguna.
Skenario eksepsi	<ul style="list-style-type: none"> - Pengguna memasukkan masukan yang tidak valid sebagai kata kunci pencarian. - Rute tidak berhasil ditemukan/tidak tersedia.

Tabel 3.3: *Scenario case* untuk fitur pencarian rute dengan angkot, dengan kata kunci pencarian lokasi awal dan akhir sebagai masukan.

Nama	Mencari rute dengan angkot menggunakan kata kunci pencarian lokasi
Deskripsi	Fitur untuk mencari cara pergi ke satu lokasi ke lokasi lainnya dengan menggunakan angkot dengan dengan kata kunci pencarian kedua lokasi sebagai masukan.
Aktor	Pengguna aplikasi
Pre-kondisi	-
	Lanjut di halaman berikutnya....

Tabel 3.3 – Lanjutan dari halaman sebelumnya

Pos-kondisi	Langkah-langkah yang harus ditempuh untuk perjalanan tersebut akan ditampilkan kepada pengguna.
Skenario utama	<ol style="list-style-type: none"> 1. Perkakas membaca masukan berupa kata kunci pencarian lokasi awal dan akhir dari argumen dalam perintah <i>command line</i>. 2. Perkakas memproses kata kunci pencarian lokasi awal dan mengirimkannya ke API KIRI untuk diproses lebih lanjut. 3. Perkakas menerima keluaran berupa koordinat <i>latitude</i> dan <i>longitude</i> lokasi awal dari API KIRI. 4. Perkakas memproses kata kunci pencarian lokasi akhir dan mengirimkannya ke API KIRI untuk diproses lebih lanjut. 5. Perkakas menerima keluaran berupa koordinat <i>latitude</i> dan <i>longitude</i> lokasi akhir dari API KIRI. 6. Perkakas mengirimkan kedua nilai <i>latitude</i> dan <i>longitude</i> lokasi ke API KIRI sebagai masukan dari proses pencarian rute. 7. Perkakas menerima keluaran akhir dari API KIRI berupa langkah-langkah dalam rute yang perlu ditempuh. 8. Perkakas menampilkan kembali keluaran tersebut kepada pengguna.
Skenario eksepsi	<ul style="list-style-type: none"> - Pengguna memasukkan masukan yang tidak valid di dalam salah satu parameter. - Lokasi yang dicari dalam langkah 2 atau langkah 4 tidak ditemukan. - Rute tidak berhasil ditemukan/tidak tersedia.

1

BAB 4

2

PERANCANGAN

- 3 Pada bab ini akan dipaparkan berbagai macam rancangan dari perkakas *command line* yang akan
4 dibuat, seperti diagram-diagram terkait, masukan serta keluaran perangkat lunak,

5 **4.1 Rancangan Alur Kerja Perkakas**

- 6 Bagian ini akan membahas mengenai alur kerja perkakas yang akan dibuat, dalam bentuk *activity*
7 *diagram* serta *sequence diagram* dari fitur-fitur perkakas. Perlu diingat bahwa perkakas yang akan
8 dibuat memiliki tiga buah fitur, yaitu:

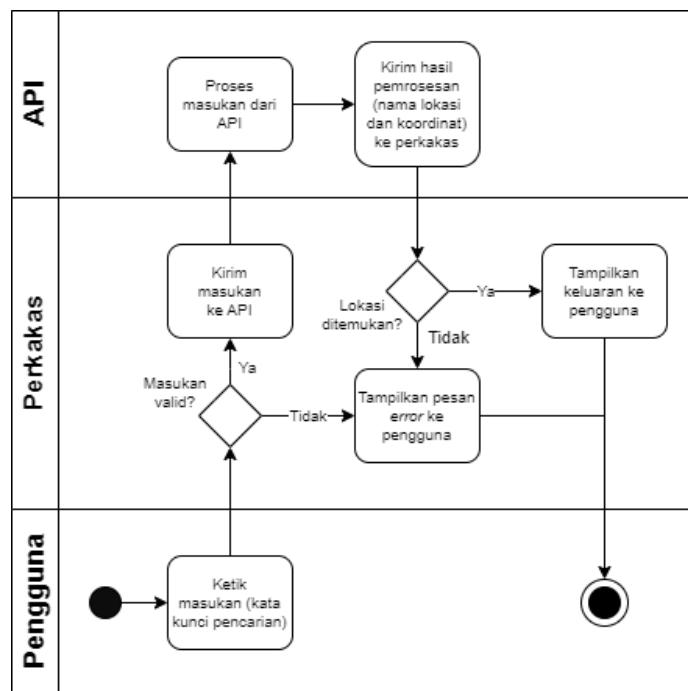
- 9 • mencari lokasi menggunakan kata kunci pencarian (**searchplace**),
- 10 • mencari rute dengan angkot menggunakan *latitude* dan *longitude* lokasi (**findroute**), dan
- 11 • mencari rute dengan angkot menggunakan kata kunci pencarian lokasi (**direct**).

- 12 Oleh karena itu, tiap-tiap fitur akan memiliki satu dari setiap tipe diagram. Adapun penjelasan
13 dari bagaimana fitur-fitur ini akan diimplementasikan adalah sebagai berikut.

14 **4.1.1 Mencari lokasi menggunakan kata kunci pencarian**

- 15 *Activity* dan *sequence diagram* untuk fitur pencarian lokasi dengan kata kunci ini dapat dilihat
16 di gambar 4.1 dan 4.2. Dari diagram-diagram tersebut, maka alur kerja fitur ini adalah sebagai
17 berikut.

- 18 1. Perkakas akan meminta masukan dari pengguna berupa kata kunci dari lokasi yang ingin
19 dicari.
- 20 2. Masukan akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan hanya akan
21 dianggap tidak valid apabila pengguna tidak memasukkan apapun sebagai masukan perkakas
22 (masukan kosong).
- 23 3. Jika kata kunci masukan:
 - 24 • tidak valid, maka perkakas akan mengeluarkan pesan *error* dan keluar.
 - 25 • dianggap valid, kata kunci tersebut akan dikirim ke API KIRI untuk diproses lebih
26 lanjut.
- 27 4. Setelah selesai diproses, keluaran dari API akan dikembalikan ke perkakas.
- 28 5. Perkakas akan mengecek keluaran yang diterimanya. Apabila lokasi:
 - 29 • ditemukan, maka nama dan koordinat *latitude* dan *longitude* lokasi akan ditampilkan ke
30 pengguna sebagai keluaran akhir.
 - 31 • tidak ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.



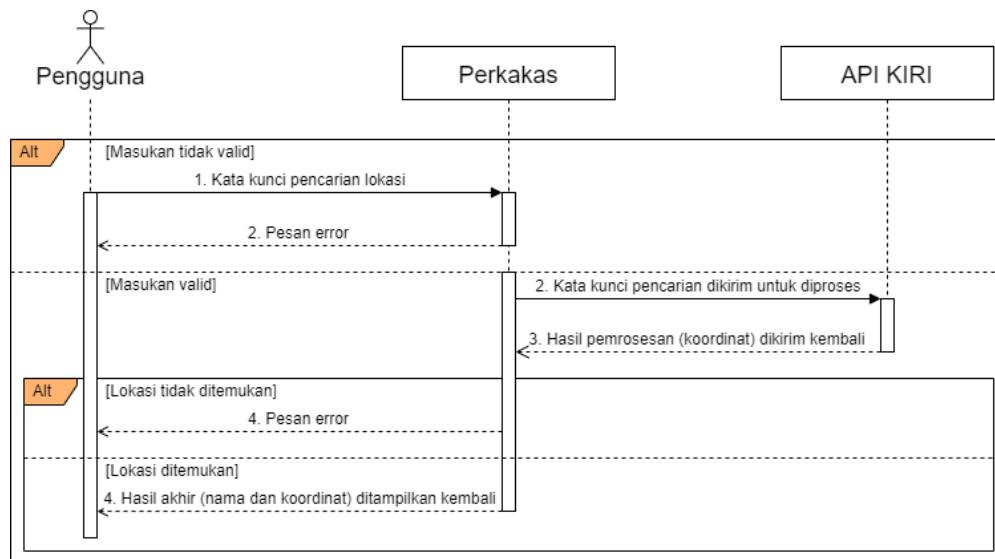
Gambar 4.1: *Activity diagram* dari fitur pencarian lokasi menggunakan kata kunci pencarian.

Perlu diingat bahwa perkakas tidak peduli apakah lokasi yang ditemukan benar (sesuai dengan yang diinginkan pengguna) atau tidak.

4.1.2 Mencari rute dengan angkot menggunakan *latitude* dan *longitude* lokasi

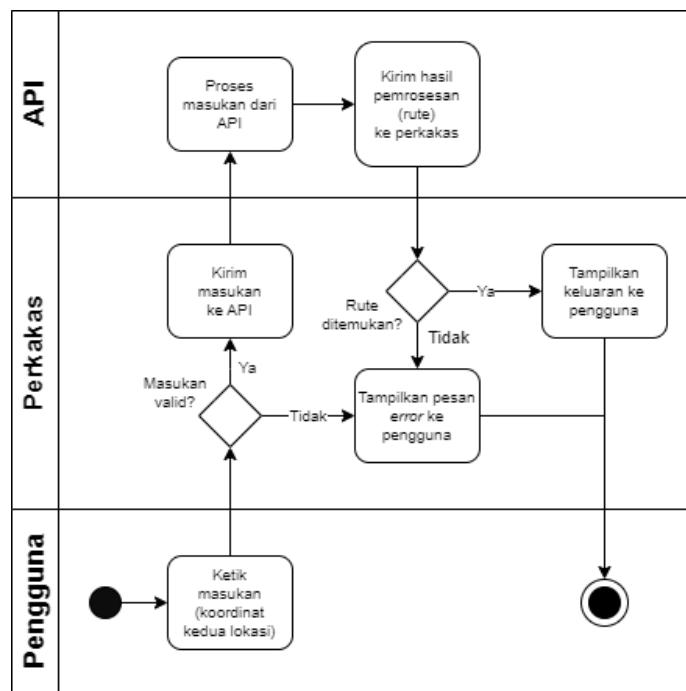
Activity dan *sequence diagram* untuk fitur pencarian rute dengan menggunakan koordinat lokasi awal dan akhir dapat dilihat di gambar 4.3 dan 4.4. Detail dari alur kerja untuk fitur ini adalah sebagai berikut.

1. Perkakas akan meminta dua buah masukan dari pengguna, berupa koordinat *latitude* dan *longitude* dari lokasi awal (mulai) dan lokasi akhir (tujuan).
2. Masukan akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan akan dianggap tidak valid apabila ada masukan yang bukan berupa koordinat *latitude* dan *longitude*.
3. Jika:
 - satu atau lebih masukan tidak valid, maka perkakas akan mengeluarkan pesan *error* dan keluar.
 - kedua masukan dianggap valid, kedua koordinat tersebut akan dikirim ke API KIRI untuk diproses lebih lanjut.
4. Setelah selesai diproses, keluaran dari API akan dikembalikan ke perkakas.
5. Perkakas akan mengecek keluaran yang diterimanya. Apabila rute:
 - berhasil ditemukan (ada setidaknya satu langkah dalam rute), maka rute tersebut akan ditampilkan ke pengguna sebagai keluaran akhir.
 - tidak berhasil ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.



Gambar 4.2: *Sequence diagram* dari fitur pencarian lokasi menggunakan kata kunci pencarian.

- 1 4.1.3 Mencari rute dengan angkot menggunakan kata kunci pencarian lokasi**
- 2 Activity dan sequence diagram untuk fitur pencarian rute dengan pencarian langsung lokasi awal dan akhirnya dapat dilihat di gambar 4.5 dan 4.6. Berikut alur dari diagram-diagram tersebut.*
- 3 1. Perkakas akan meminta masukan dari pengguna berupa kata kunci dari lokasi awal dan lokasi akhir yang ingin dicari.*
- 4 2. Masukan untuk lokasi awal akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan hanya akan dianggap tidak valid apabila pengguna tidak memasukkan apapun (masukan kosong), atau memasukkan koordinat *latitude* dan *longitude* sebagai masukan perkakas.*
- 5 3. Jika kata kunci masukan:*
 - 6 • tidak valid, maka perkakas akan mengeluarkan pesan *error* dan keluar.*
 - 7 • dianggap valid, kata kunci tersebut akan dikirim ke API KIRI untuk diproses lebih lanjut.*
- 8 4. Setelah selesai diproses, keluaran dari API akan dikembalikan ke perkakas.*
- 9 5. Perkakas akan mengecek keluaran yang diterimanya. Apabila lokasi:*
 - 10 • tidak ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.*
 - 11 • ditemukan, maka nama dan koordinat *latitude* dan *longitude* lokasi awal akan disimpan sebagai variabel untuk dipakai di langkah selanjutnya.*
- 12 6. Langkah 2 sampai 5 akan diulang kembali untuk lokasi akhir.*
- 13 7. Jika kata kunci kedua lokasi berhasil diproses, perkakas akan mengirim kembali koordinat *latitude* dan *longitude* dari kedua lokasi tersebut ke API sebagai masukan untuk proses kedua, yaitu mencari rute angkot dengan koordinat *latitude* dan *longitude* lokasi.*
- 14 8. Setelah selesai diproses, keluaran berupa rute dari API akan dikembalikan ke perkakas.*
- 15 9. Perkakas akan mengecek keluaran yang diterimanya. Apabila rute:*
 - 16 • berhasil ditemukan (ada setidaknya satu langkah dalam rute), maka rute tersebut akan ditampilkan ke pengguna sebagai keluaran akhir.*
 - 17 • tidak berhasil ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.*



Gambar 4.3: *Activity diagram* dari fitur pencarian rute angkot menggunakan *latitude* dan *longitude* lokasi.

1 4.2 Rancangan Implementasi Perkakas

2 Bagian ini akan membahas hal-hal seputar rancangan implementasi alur kerja perkakas, variabel
 3 variabel utama yang diperlukan di dalam perkakas, serta hal-hal seputar fungsi-fungsi yang ada di
 4 dalam perkakas, seperti nama fungsinya, apa tujuan dari fungsi tersebut, masukan dan keluarannya,
 5 serta garis besar dari cara kerja fungsi tersebut.

6 4.2.1 Cara Kerja Perkakas

7 Untuk setiap operasinya, perkakas *command line* KIRI ini terdiri atas empat proses utama, dengan
 8 urutan operasi internal sebagai berikut.

9 1. Penerimaan opsi dan argumen

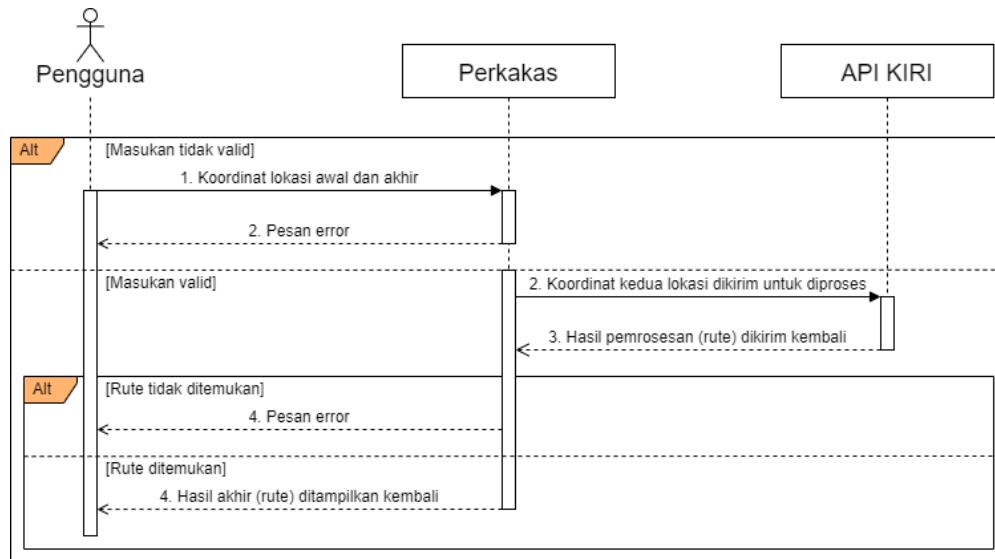
10 Di proses ini, perkakas akan melakukan pemeriksaan paling dasar, yaitu memeriksa apakah
 11 pengguna menggunakan opsi dan memasukkan argumennya dengan tepat atau tidak. *Error*
 12 *checking* yang dilakukan di tahap ini juga hanya sebatas pengecekan dasar, seperti pengecekan
 13 validitas opsi yang digunakan, jumlah argumen yang dimasukkan, serta validitas mode operasi
 14 dari perkakas (opsi `--mode`). Kesalahan apapun yang terdapat di hal-hal selain kedua aspek
 15 tersebut hanya akan ditandai.

16 2. Verifikasi kebenaran opsi dan argumen

17 Di proses ini, perkakas akan melakukan pengecekan lanjutan terhadap argumen-argumen
 18 yang dimasukkan pengguna. Jika ada argumen yang tidak valid, perkakas akan langsung
 19 berhenti dan mengeluarkan pesan *error* yang sesuai.

20 3. Pengiriman permintaan GET ke API serta penerimaan kembali respons

21 Setelah semua opsi dan argumen yang dibutuhkan diverifikasi validitasnya, perkakas akan



Gambar 4.4: *Sequence diagram* dari fitur pencarian rute angkot menggunakan *latitude* dan *longitude* lokasi.

1 membangun URL yang diperlukan untuk permintaan GET ke API, dan melakukan permintaan
 2 tersebut melalui cURL. *Library* cURL ini nantinya juga adalah *library* yang memfasilitasi
 3 penerimaan kembali respons dari API.

4. Verifikasi akhir respons API

5 Terakhir, isi dari respons API akan diperiksa. Segala abnormalitas yang terdapat di dalam
 6 isi respons tersebut akan ditangani langsung oleh perkakas dalam bentuk pesan-pesan *error*
 7 untuk tiap kasusnya.

8 4.2.2 Tipe Data Tambahan

9 Salah satu dari variabel global yang ada di perkakas ini memiliki tipe data **chunk**, yang merupakan
 10 sebuah **struct** tambahan yang dibuat sendiri. Adapun variabel-variabel yang ada di dalam struktur
 11 tersebut adalah sebagai berikut.

- 12 • **data** (`char *`)
- 13 • **size** (`size_t`)

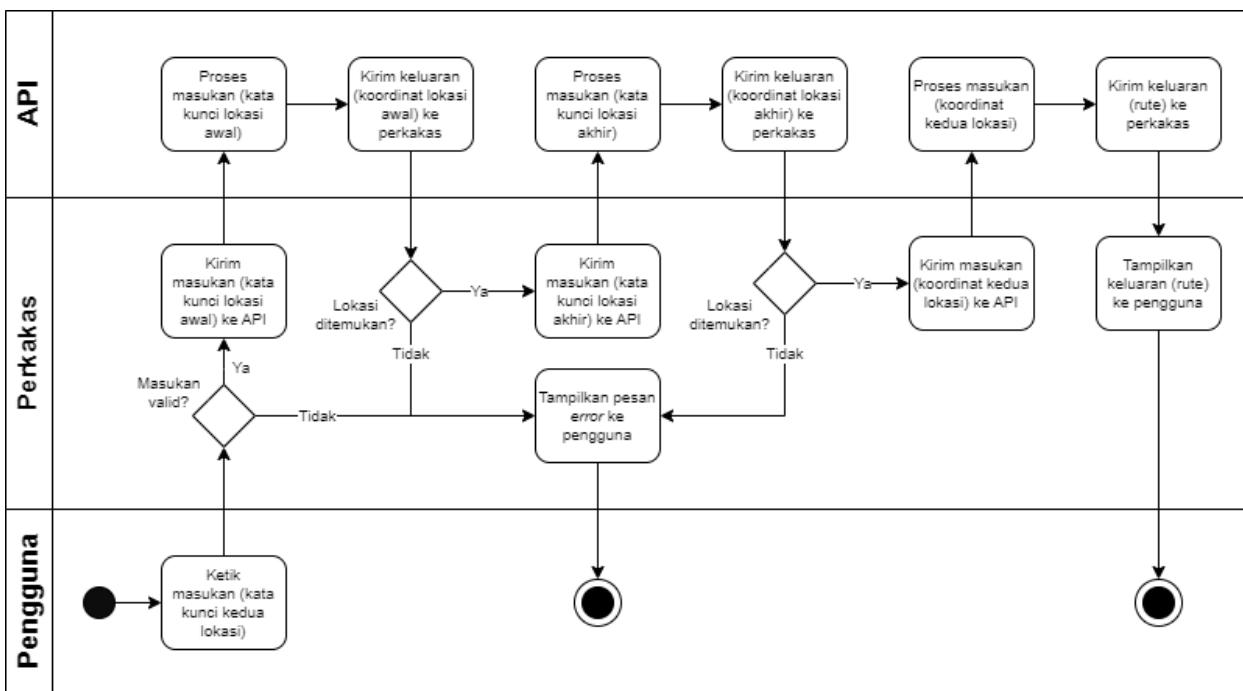
14 Struktur ini akan digunakan untuk variabel **responsedata**, yang akan diisi dengan respons dari
 15 API KIRI setelah eksekusi proses cURL. Variabel **data** di struktur ini merupakan penunjuk ke
 16 *array* karakter yang akan menampung data dari respons API itu sendiri, sedangkan **size**, yang
 17 merupakan sebuah *unsigned integer*, akan diisi dengan ukuran dari data tersebut.

18 4.2.3 Variabel Global

19 Perkakas ini memiliki beberapa variabel global, yang sengaja diletakkan sebagai variabel global
 20 karena variabel-variabel ini digunakan di hampir keseluruhan dari keempat proses di subbab???.
 21 Adapun variabel-variabel ini adalah sebagai berikut:

- 22 • **responsedata** (`chunk`)

23 Variabel ini akan diisi oleh respons dari API KIRI. Seperti yang telah dipaparkan secara



Gambar 4.5: *Activity diagram* dari fitur pencarian rute angkot menggunakan kata kunci pencarian.

singkat di subbab sebelumnya, struktur ini memiliki dua variabel, yaitu **data**, yang akan berisi data responsnya sendiri, dan **size**, yang merupakan ukuran dari data tersebut.

- **responseJSON (cJSON)**

Berisi hasil konversi JSON dari respons API yang dapat dibaca oleh perkakas dan utilitas *library cJSON*.

- **URL (char[])**

Basis dari URL yang akan digunakan sebagai URL permintaan GET ke API. Awalnya variabel ini akan berisi “<https://projectkiri.id/api?version=2>”—bagian awal ini tidak akan berubah bagaimanapun permintaan GET-nya.

- **mode (int)**

Kode operasional perkakas berupa bilangan bulat. Bilangan ini memiliki rentang dari -1 hingga 4, dengan arti dari tiap bilangan sebagai berikut.

- **-1**: Mode belum didefinisikan

- **0**: Mode tidak valid

- **1**: Mode bantuan (*help*)

- **2**: Mode *searchplace*

- **3**: Mode *findroute*

- **4**: Mode *direct*

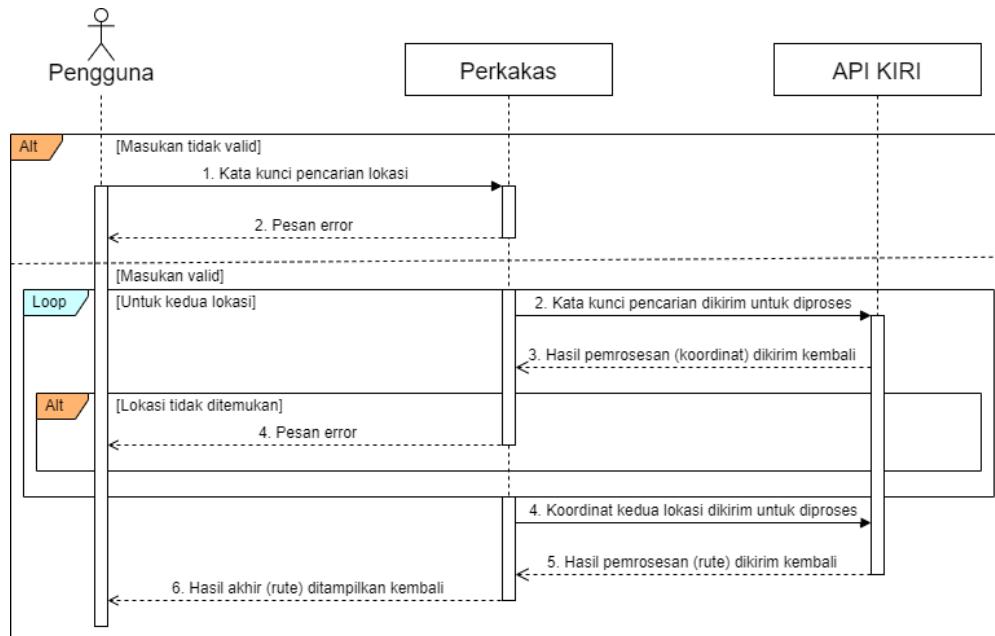
- **region (int)**

Kode region berupa bilangan bulat untuk mode *searchplace*. Bilangan ini memiliki rentang dari -1 hingga 4, di mana arti dari tiap bilangan dapat dilihat di daftar berikut.

- **-1**: Region belum didefinisikan

- **0**: Region tidak valid

- **1**: cgk (Cengkareng/Jakarta)



Gambar 4.6: Sequence diagram dari fitur pencarian rute angkot menggunakan kata kunci pencarian.

- 1 – **2:** bdo (Bandung)
- 2 – **3:** mlg (Malang)
- 3 – **4:** sub (Surabaya)
- 4 • **query (char[])**
 5 *String* (berupa *array* karakter) yang menyimpan kata kunci pencarian lokasi untuk mode
 6 **searchplace**.
- 7 • **start (char[])**
 8 *String* (berupa *array* karakter) yang menyimpan koordinat lokasi awal (**findroute**) atau kata
 9 kunci pencarian lokasi awal (**direct**).
- 10 • **finish (char[])**
 11 *String* (berupa *array* karakter) yang menyimpan koordinat lokasi akhir (**findroute**) atau
 12 kata kunci pencarian lokasi akhir (**direct**).
- 13 • **escape (char[])**
 14 Variabel ini dibutuhkan karena pengkodean URL (untuk permintaan GET) tidak mendukung
 15 karakter spasi (' '), melainkan "%20". *String* yang karakter spasinya sudah diganti dengan
 16 "%20" akan disimpan sementara di variabel ini, sebelum isinya disalin kembali ke variabel
 17 awalnya.
- 18 • **regstart (int)**
 19 Kode region lokasi awal berupa bilangan bulat untuk mode **direct**. Bilangan ini memiliki
 20 rentang dari -1 hingga 4, dengan tiap bilangan bulat merepresentasikan region yang sama
 21 dengan kode bilangan untuk variabel **region**.
- 22 • **regfinish (int)**
 23 Kode region lokasi akhir berupa bilangan bulat untuk mode **direct**. Bilangan ini memiliki
 24 rentang dari -1 hingga 4, dengan tiap bilangan bulat merepresentasikan region yang sama
 25 dengan kode bilangan untuk variabel **region**.

1 • **locale** (int)

2 Kode bahasa berupa bilangan bulat. Bilangan ini memiliki rentang dari 0 hingga 2, dengan
3 tiap-tiap bilangan merepresentasikan arti berikut.

4 – **0**: id (Indonesia)

5 – **1**: en (Inggris)

6 – **2**: Kode bahasa tidak valid

7 Variabel ini juga merupakan satu-satunya variabel kode *integer* yang, jika tidak diubah nilai
8 awalnya (**0**), tidak akan menyebabkan *error*.

9 • **step** (int)

10 Kode khusus untuk mode **direct** yang menandakan proses mana yang sedang berlangsung—pencarian lokasi awal, pencarian lokasi akhir, atau pencarian rute.

11 • **error** (int)

12 Kode yang menandakan apakah sebuah *error* telah terjadi. Nilai awal dari variabel ini adalah
13 **0**, dan jika variabel ini diganti menjadi **1**, di pengecekan kode *error* selanjutnya, perkakas
14 akan dihentikan.

16 **4.2.4 print_help()**

17 Fungsi ini merupakan fungsi yang akan dipanggil ketika pengguna memilih opsi standar **--help**.
18 Adapun tujuan dari fungsi ini hanya satu, yaitu menampilkan sekumpulan *array* karakter yang
19 merupakan bantuan penggunaan perkakas.

20 **4.2.5 replace_space()**

21 Fungsi ini merupakan fungsi yang berguna untuk meng-*escape* semua karakter spasi (' ') dalam
22 variabel-variabel kata kunci pencarian sebelum isi dari variabel-variabel tersebut dimasukkan ke
23 dalam URL untuk permintaan GET. Hal ini perlu dilakukan karena pengkodean HTML yang
24 digunakan untuk format URL tidak medukung karakter spasi, dan melainkan mensubstitusi seluruh
25 kejadian karakter tersebut dalam suatu URL menjadi '%20'. Fungsi ini akan menerima sebuah *array*
26 karakter berisi URL permintaan sebagai masukannya, mengganti semua karakter spasi yang ada di
27 dalamnya menjadi '%20', dan meletakkan hasilnya di variabel global **escape**. Adapun *pseudocode*
28 dari fungsi ini dapat dilihat di algoritma 1.

29 **4.2.6 build_url_searchplace()**

30 Fungsi ini merupakan fungsi yang digunakan untuk pembangunan URL permintaan GET untuk
31 proses pencarian lokasi. Fungsi ini menerima dua variabel, yaitu **region** dan **query**, yang keduanya
32 dapat diatur berdasarkan proses pencarian lokasi untuk fitur mana yang sedang dilakukan. Hasil
33 dari pembangunan URL akan dituliskan langsung di dalam variabel global **url**.

34 Selain itu, fitur ini juga memiliki implementasi tambahan yang tidak ada di API KIRI, yaitu
35 implementasi variabel **locale** untuk mode **searchplace**, yang memungkinkan pemilihan bahasa
36 Indonesia atau Inggris untuk keluaran serta pesan-pesan *error* yang berhubungan dengan mode
37 tersebut. Adapun algoritma dari fungsi ini dapat dilihat di algoritma 2.

Algorithm 1 Algoritma fungsi `replace_space()`

Variabel global: `escape`**Input:** `string`**Output:** -

```

j ← 0
for i ← 0 to size of string do
    if string[i] == '\0' then    ▷ Berhenti jika fungsi sudah mencapai karakter terakhir string
        break
    end if
    if string[i] == ' ' then
        escape[j] ← '%'
        escape[j + 1] ← '2'
        escape[j + 2] ← '0'
        j ← j + 2                ▷ Majukan indeks escape sebanyak 3 (2 + 1 di akhir fungsi)
    else
        escape[j] ← string[i]
    end if
    j ← j + 1
end for


---



```

1 4.2.7 `build_url_findroute()`

- 2 Fungsi ini merupakan fungsi yang digunakan untuk pembangunan URL permintaan GET untuk
 3 proses pencarian rute angkot. Fungsi ini menerima tiga variabel, yaitu `locale`, `start`, dan `finish`.
 4 Hasil dari pembangunan URL permintaan juga akan dituliskan langsung di dalam variabel global
 5 `url`. Adapun rancangan implementasi dari fungsi ini dapat dilihat di algoritma 3.

6 4.2.8 `reset_url()`

- 7 Fungsi ini digunakan untuk mengembalikan isi dari variabel `URL` ke nilai semula, dengan cara menim-
 8 pa isi dari variabel tersebut dengan nilai awalnya, yaitu “<https://projectkiri.id/api?version=2>”.
 9 Fungsi ini akan dipanggil hanya di fitur `direct`, dan hanya di saat pergantian proses dalam fitur
 10 tersebut, dengan asumsi bahwa tidak ada `error` yang terjadi (nilai variabel `error` bukan 1) di
 11 langkah terakhir sebelum fungsi ini dipanggil.

12 4.2.9 `execute_curl()`

- 13 Fungsi ini digunakan untuk proses pengiriman permintaan ke API, serta penerimaan respons dari
 14 API. Semua utilitas dari library cURL yang dipakai dalam perkakas ini, seperti `handler` kelebihan
 15 memori data yang masuk, dan aturan variabel apa yang akan diisi dengan data respons API, akan
 16 diimplementasikan hanya di dalam fungsi ini. Adapun *pseudocode* dari fungsi ini dapat dilihat di
 17 potongan algoritma 4.

18 4.2.10 `print_curl_error()`

- 19 Fungsi ini merupakan fungsi sederhana yang akan mengeluarkan pesan `error` apabila terjadi `error`
 20 koneksi ketika perkakas ingin melakukan komunikasi dengan API KIRI. Fungsi ini akan mengubah

- 1 nilai variabel **error** ke 1, untuk menandakan bahwa sebuah *error* telah terjadi, dan kemudian
- 2 mengeluarkan pesan *error* mengenai terjadinya *error* cURL.

3 **4.2.11 write_malloc()**

- 4 Fungsi ini adalah fungsi tambahan untuk cURL yang bertugas menerima data yang masuk dari API.
- 5 Di fungsi ini juga diimplementasikan sebuah *handler* pengalokasian memori, yang akan menghindari
- 6 terjadinya *error* akibat ukuran data yang diterima melebihi alokasi memori yang diperbolehkan
- 7 untuk variabel yang akan diisi dengan data tersebut. Adapun alur kerja dari fungsi ini dapat dilihat
- 8 di algoritma 5.

9 **4.2.12 write_searchplace()**

- 10 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian
- 11 lokasi dari mode **searchplace**. Fungsi ini akan mengambil data dari variabel **responsedata** dan
- 12 langsung menampilkan keluaran akhir, atau jika terjadi *error*, maka fitur ini akan menampilkan
- 13 pesan *error* yang sesuai. Adapun implementasi dari fungsi ini dapat dilihat di potongan algoritma
- 14 6.

15 **4.2.13 write_findroute()**

- 16 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian rute
- 17 angkot. Fungsi ini akan mengambil data dari variabel **responsedata** dan langsung menampilkan
- 18 keluaran akhir, atau jika terjadi *error*, maka fitur ini akan menampilkan pesan *error* yang sesuai.
- 19 Adapun implementasi dari fungsi ini dapat dilihat di potongan algoritma 7.

20 Perlu diperhatikan bahwa ada sebuah *bug* dari API KIRI di mana durasi rute dalam menit tidak
21 diterjemahkan dari bahasa Inggris, apabila variabel **locale** diatur ke bahasa Indonesia. Potongan
22 kode untuk mengatasi *bug* ini terdapat di fungsi ini.

23 **4.2.14 write_searchplace_noreturns()**

- 24 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian lokasi
- 25 dari mode **direct**. Berbeda dengan fungsi **searchplace** sebelumnya, fungsi ini akan mengambil
- 26 data dari variabel **responsedata**, tetapi hanya akan menampilkan nama lokasi (awal dan/atau
- 27 akhir), atau jika terjadi *error* di salah satu prosesnya, maka fitur ini akan menampilkan pesan *error*
- 28 yang sesuai. Adapun implementasi dari fungsi ini dapat dilihat di potongan algoritma 8.

29 **4.2.15 Fungsi utama (main)**

- 30 Tentunya seluruh perangkat lunak bahasa C akan memiliki satu buah fungsi utama. Bagian ini akan
- 31 membahas garis besar alur kerja dari fungsi utama yang ada di perkakas ini melalui *pseudocode* 9.

Algorithm 2 Algoritma fungsi `build_url_searchplace()`

Variabel global: *locale, error, url*

Input: *region, query*

Output: *url* yang sudah dilengkapi

```
if locale == 2 then
    error ← 1
    print pesan error: locale tidak valid
    hentikan perkakas
end if
switch region do
    case -1
        error ← 1
        print pesan error sesuai locale: region tidak dimasukkan
        hentikan perkakas
    case 1
        tambahkan parameter region "cgk" ke url
        break
    case 2
        tambahkan parameter region "bdo" ke url
        break
    case 3
        tambahkan parameter region "mlg" ke url
        break
    case 4
        tambahkan parameter region "sub" ke url
        break
    default
        error ← 1
        print pesan error sesuai locale: region tidak valid
        hentikan perkakas
end switch
if query == '\0' then
    error ← 1
    print pesan error sesuai locale: query tidak dimasukkan
    hentikan perkakas
else
    tambahkan parameter query ke url
end if
tambahkan parameter kunci API ke url
```

Algorithm 3 Algoritma fungsi `build_url_findroute()`

Variabel global: *url, error*

Input: *locale, start, finish*

Output: *url* yang sudah dilengkapi

```
switch region do
    case 1
        tambahkan parameter locale "en" ke url
        break
    case 2
        error ← 1
        print pesan error: locale tidak valid
        hentikan perkakas
    default
        tambahkan parameter locale "id" ke url
        break
end switch
if start == '\0' then
    error ← 1
    print pesan error sesuai locale: start tidak dimasukkan
    hentikan perkakas
else
    tambahkan parameter start ke url
end if
if finish == '\0' then
    error ← 1
    print pesan error sesuai locale: finish tidak dimasukkan
    hentikan perkakas
else
    tambahkan parameter query ke url
end if
tambahkan parameter presentation "desktop" ke url
tambahkan parameter kunci API ke url
```

Algorithm 4 Algoritma fungsi `execute_curl()`

Variabel global: *responsesdata, mode, step***Input:** -**Output:** *responsesdata* yang sudah diisi dengan data respons cURL

```


curl ← penunjuk ke handle cURL
curlcode ← kode respons cURL
kosongkan responsesdata                                ▷ Untuk fungsi dengan banyak langkah
inisialisasi curl
if curl ≠ false then                                     ▷ Selama proses cURL masih berjalan
    atur opsi-opsi curl
    curlcode ← jalankan proses curl
    if curlcode bukan error then
        panggil fungsi: print_curl_error()
    end if
    switch mode do
        case 2
            panggil fungsi: write_searchplace()
            break
        case 3
            panggil fungsi: write_findroute()
            break
        case 4
            if step == 0 || step == 1 then                  ▷ Pencarian lokasi awal dan akhir
                panggil fungsi: write_searchplace_noreturns()
                break
            else if step == 2 then                         ▷ Pencarian rute
                panggil fungsi: write_findroute()
                break
            end if
        end switch
        lakukan cleanup handle curl
    end if
    lakukan global cleanup cURL



---



```

Algorithm 5 Algoritma fungsi `write_memalloc()`

Variabel global: -**Input:** *incomingdata, size, nmenb, userdata***Output:** *realsize*

```


realsize ← size * nmemb                                ▷ Hitung ukuran data yang diterima
memory ← userdata                                      ▷ Masukkan data yang diterima ke variabel tujuan
ptr ← realokasi ukuran memori dari data dalam memory
if ptr == NULL then                                     ▷ Kalau realokasi gagal...
    print pesan error: Memori habis
    return 0                                              ▷ Menandakan bahwa realokasi memori gagal
end if
Isi data dalam ptr ke chunk memory
Perbarui ukuran data dalam chunk memory
return realsize                                         ▷ Untuk verifikasi keutuhan data asli



---



```

Algorithm 6 Algoritma fungsi `write_searchplace()`

Variabel global: `responsedata`, `responseJSON`, `error`, `locale`

Input: -

Output: -

```
responseJSON ← data dalam responsedata
status ← "status" dalam responseJSON
if status ≠ "ok" then
    error ← 1
    print pesan error sesuai locale: hasil API error
else
    result ← "searchresult" dalam responseJSON
    if size of result == 0 then
        error ← 1
        print pesan output sesuai locale: Tidak ada lokasi yang ditemukan
    else
        indexitem ← 1
        for each resultitem in result do                                ▷ Untuk setiap lokasi dalam reest...
            resultitemname ← "placename" dalam resultitem
            resultitemlocation ← "location" dalam resultitem
            print resultitemname dan resultitemlocation
            indexitem ← indexitem + 1
        end for
    end if
end if
```

Algorithm 7 Algoritma fungsi `write_findroute()`

Variabel global: `responedata`, `responseJSON`, `error`, `locale`

Input: -

Output: -

```

responseJSON ← data dalam responedata
status ← "status" dalam responseJSON
if status ≠ "ok" then
    error ← 1
    print pesan error sesuai locale: hasil API error
else
    result ← "routingresults" dalam responseJSON
    indexroute ← 1
    for each route in result do                                ▷ Untuk setiap rute dalam reesult...
        routesteps ← "steps" dalam resultitem
        routetime ← "traveltime" dalam resultitem
        if routetime == NULL then
            error ← 1
            print pesan output sesuai locale: Tidak ada rute yang ditemukan
        else
            indexstep ← 1
            if locale ≠ 1 then                               ▷ Betulkan bug durasi menit dalam locale id
                routetimestricting ← hasil terjemahan “minute” dalam routetime ke “menit”
            else
                routetimestricting ← routetime
            end if
            print routetimestricting
            for each routestepitem in routesteps do          ▷ Untuk setiap langkah rute...
                routestepdetail ← routestepitem[3]           ▷ Langkah dalam bahasa natural ada di indeks ke-3
                print routestepdetail
            end for
        end if
        indexroute ← indexroute + 1
    end for
end if


---



```

Algorithm 8 Algoritma fungsi `write_searchplace_noreturns()`

Variabel global: `responsesdata`, `responseJSON`, `error`, `step`, `locale`

Input: -

Output: -

```

responseJSON ← data dalam responsesdata
status ← "status" dalam responseJSON
if status ≠ "ok" then
    error ← 1
    print pesan error sesuai locale: hasil API error
else
    result ← "searchresult" dalam responseJSON
    if size of result == 0 then
        error ← 1
        print pesan output sesuai locale: Tidak ada lokasi yang ditemukan
    else
        indexitem ← 1
        resultitem ← result[0]
            ▷ Mode direct hanya mendukung lokasi pertama yang ditemukan
        resultitemname ← "placename" dalam resultitem
        resultitemlocation ← "location" dalam resultitem
        print resultitemname
        if step == 0 then
            start ← resultitemlocation
        else if step == 1 then
            finish ← resultitemlocation
        end if
    end if
end if
```

Algorithm 9 Alur kerja fungsi utama perkakas

Variabel global: semua variabel di subbab 4.2.3

Input: *argc*, *argv* ← akan digunakan untuk *getopt*

Output: *Exit code* perkakas.

```

while TRUE do
    funct ← hasil getopt
    switch funct do
        case 'h'
            if mode ≠ -1 then
                ganti mode ke mode bantuan
            end if
        case 'm'
            if mode ≠ -1 then
                ganti mode ke mode operasional yang sesuai
            end if
        case 'r'
            ganti region ke region yang sesuai
        case 'q'
            if Masukan untuk query tidak kosong then
                escape semua karakter spasi dalam masukan
                query ← hasil escape
            end if
        case 's'
            if Masukan untuk start tidak kosong then
                escape semua karakter spasi dalam masukan
                start ← hasil escape
            end if
        case 'f'
            if Masukan untuk finish tidak kosong then
                escape semua karakter spasi dalam masukan
                finish ← hasil escape
            end if
        case 'S'
            ganti regstart ke region yang sesuai
        case 'F'
            ganti regfinish ke region yang sesuai
        case 'l'
            ganti locale ke bahasa yang sesuai
        case ':'
            error ← 1
            print pesan error sesuai locale: Ada opsi yang kehilangan argumen
            hentikan perkakas
        case '?'
            error ← 1
            print pesan error sesuai locale: Ada opsi yang tidak valid
            hentikan perkakas
    end switch
end while

```

Algorithm 9 - Lanjutan dari halaman sebelumnya

```

if ada kelebihan argumen then
    error  $\leftarrow$  1
    print pesan error sesuai locale: Ada kelebihan argumen
else
    switch funct do
        case -1
            error  $\leftarrow$  1
            print pesan error sesuai locale: Tidak ada masukan ke mode
            hentikan perkakas
        case 1
            panggil fungsi: print_help()
        case 2
            panggil fungsi: build_url_searchplace(region, query)
        case 3
            panggil fungsi: build_url_findroute(locale, start, finish)
        case 4
            step  $\leftarrow$  0
            panggil fungsi: build_url_searchplace(regstart, start)
            panggil fungsi: execute_curl()
            if error == 1 then
                break
            else
                panggil fungsi: reset_url()
            end if
            step  $\leftarrow$  1
            panggil fungsi: build_url_searchplace(regfinish, finish)
            panggil fungsi: execute_curl()
            if error == 1 then
                break
            else
                panggil fungsi: reset_url()
            end if
            step  $\leftarrow$  2
            panggil fungsi: build_url_findroute(locale, start, finish)
            panggil fungsi: execute_curl()
        default
            error  $\leftarrow$  1
            print pesan error sesuai locale: mode tidak valid
            hentikan perkakas
    end switch
end if
return 0

```

1

BAB 5

2

IMPLEMENTASI DAN PENGUJIAN

- 3 Bab ini akan membahas hal-hal mengenai implementasi kode dari tiap-tiap fungsi dalam perkakas,
4 serta skenario-skenario pengujian yang akan digunakan dalam pengujian fungsional perkakas.

5 **5.1 Implementasi Kode**

- 6 Subbab ini akan memaparkan implementasi dari fungsi-fungsi yang ada dalam perkakas secara
7 singkat tetapi menyeluruh, serta membahas secara detail sebagian kecil dari fungsi-fungsi tersebut
8 yang memerlukan penjelasan lebih lanjut.

9 **5.1.1 print_help()**

- 10 Fungsi ini akan menampilkan *string-string* (dalam bentuk *array-array*karakter) yang merupakan
11 versi singkat dari bantuan penggunaan perkakas ke pengguna. Implementasi dari fungsi ini ada di
12 lampiran A.2, mulai dari baris ke-336 sampai dengan baris ke-394.

13 **5.1.2 replace_space()**

- 14 Seperti yang telah dijelaskan di subbab 4.2.5, fungsi ini akan menerima sebuah *array* karakter,
15 mengganti semua karakter spasi yang ada di dalamnya menjadi ‘%20’, dan meletakkan hasilnya di
16 variabel global `escape`. Implementasi dari fungsi ini dapat dilihat di *source code* murni perkakas
17 dalam lampiran A.2, mulai dari baris ke-623 sampai dengan baris ke-643.

18 **5.1.3 build_url_searchplace()**

- 19 Fungsi ini bekerja dengan menambahkan URL ke variabel global `url`, sesuai dengan opsi-opsi yang
20 telah dimasukkan (dan diperlukan oleh API) untuk fitur *searchplace*. Adapun implementasi dari
21 fungsi ini dapat dilihat di lampiran A.2, di baris 455 sampai baris 523.

22 **5.1.4 build_url_findroute()**

- 23 Fungsi ini bekerja dengan menambahkan URL ke variabel global `url`, sesuai dengan opsi-opsi yang
24 telah dimasukkan (dan diperlukan oleh API) untuk fitur *findroute*. Adapun implementasi dari
25 fungsi ini dapat dilihat di lampiran A.2, di baris 547 sampai baris 616.

1 **5.1.5 reset_url()**

2 Fungsi ini hanya terdiri atas satu baris kode yang akan mengembalikan isi dari variabel `url` ke nilai
3 awalnya. Implementasi dari fungsi ini dapat dilihat langsung di lampiran [A.2](#), di baris 618 sampai
4 621.

5 **5.1.6 execute_curl()**

6 Fungsi ini merupakan fungsi yang mengatur seluruh proses yang berhubungan langsung dengan
7 cURL dalam perkakas, mulai dari inisialisasi *easy handle* cURL (serta proses pembersihannya di
8 akhir fungsi), pengaturan penerimaan data keluaran proses cURL (`write_memalloc()`), pengaturan
9 variabel apa dalam perkakas yang akan diisi oleh data keluaran, serta *error handler* apabila proses
10 cURL gagal. Selain itu, fungsi ini juga mengatur proses apa yang harus dilakukan setelah data
11 tersebut berhasil diterima, dengan cara memanggil fungsi yang sesuai dengan mode operasional
12 yang telah ditentukan dalam masukan yang diberikan oleh pengguna. Adapun implementasi dari
13 fungsi ini dapat dilihat di lampiran [A.2](#) dari baris 410 ke baris 453.

14 **5.1.7 print_curl_error()**

15 Fungsi ini akan dipanggil apabila proses curl dari fungsi `execute_curl()` tidak mengembalikan
16 “OK” sebagai kode respons cURLnya. Cara kerja fungsi ini adalah dengan mengecek kode bahasa
17 (variabel `locale`) dan mengeluarkan pesan *error* yang sesuai. Implementasi fungsi ini dapat dilihat
18 di lampiran [A.2](#), di baris 396 sampai dengan 408.

19 **5.1.8 write_memalloc()**

20 Fungsi ini adalah fungsi yang bertugas untuk memastikan bahwa data keluaran dari API yang
21 diterima tidak melebihi ukuran maksimal yang diperbolehkan untuk dimasukkan ke dalam variabel
22 tujuan. Implementasi dari fungsi ini dapat dilihat di lampiran [A.2](#), di baris ke-33 sampai dengan
23 baris ke-47.

24 Penjelasan detail dari proses yang dilakukan dalam fungsi ini adalah sebagai berikut.

- 25 • Fungsi ini akan memiliki *return value* bertipe *unsigned integer size_t*. Hal ini merupakan
26 kewajiban dari *library* cURL sendiri.
- 27 • Seperti yang sudah disebutkan di bagian rancangan implementasi, fungsi ini akan memiliki
28 empat variabel masukan, yaitu:

29 – `incomingdata`

30 Variabel ini merupakan data keluaran dari API yang diterima dalam proses cURL.

31 – `size`

32 Variabel ini merupakan ukuran dari satu buah objek data. Variabel ini selalu bernilai 1,
33 yang juga merupakan kewajiban dari *library* cURL.

34 – `nmem`

35 Variabel ini merupakan ukuran dari data keluaran tersebut.

36 – `userdata`

37 Variabel ini merupakan penunjuk ke variabel dalam perkakas yang akan diisi oleh data
38 yang diterima proses cURL.

- 1 • Implementasi cara kerja fungsi ini adalah sebagai berikut:
- 2 1. **Baris 34:** Hitung ukuran dari data yang masuk.
- 3 2. **Baris 35–40:** Cek apakah ukuran data melebihi yang diperbolehkan.
- 4 3. **Baris 41–44:** Setor data yang dimasukkan ke dalam variabel tujuan.
- 5 4. **Baris 46:** Kembalikan ukuran data yang masuk untuk verifikasi keutuhan data.

6 **5.1.9 write_searchplace()**

7 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian
8 lokasi dari mode `searchplace`. Penerapan fungsi ini hanya meliputi pengambilan data yang
9 diterima, mengubah format data tersebut dari JSON ke tipe data yang bisa langsung diproses dalam
10 fungsi-fungsi bawaan bahasa C, dan kemudian menampilkan hasilnya ke pengguna. Implementasi
11 dari fungsi ini dapat dilihat di lampiran kode murni perkakas, baris 49 sampai baris 127.

12 **5.1.10 write_findroute()**

13 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian rute
14 angkot. Sama seperti fungsi `write_searchplace()`, penerapan fungsi ini hanya meliputi pengam-
15 bilan data yang diterima, konversi format data tersebut dari JSON, dan kemudian menampilkan
16 hasilnya ke pengguna. Implementasi dari fungsi ini dapat dilihat di lampiran kode murni perkakas,
17 baris 218 sampai baris 334.

18 Perlu ditekankan kembali bahwa fitur ini memiliki pembetulan *bug* tambahan, yang diimplemen-
19 tasikan di baris 273 sampai dengan baris 292.

20 **5.1.11 write_searchplace_noreturns()**

21 Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian
22 lokasi dari mode `direct`. Seperti yang sudah dijelaskan di bagian perancangan alur kerja dari
23 fungsi ini (subbab 4.2.14), fungsi ini mirip dengan fungsi `write_searchplace()`, hanya saja untuk
24 keluarannya, fungsi ini tidak akan menampilkan koordinat lokasi yang ditemukan. Implementasi
25 dari fungsi ini ada di lampiran A.2, di baris 129 sampai dengan 216.

26 **5.1.12 Fungsi utama (`main`)**

27 Fungsi `main` di perkakas ini dapat dibagi menjadi dua buah proses utama, yaitu penerimaan
28 masukan dari pengguna, serta penentuan langkah-langkah yang harus dijalankan untuk tiap-tiap
29 mode operasional. Adapun implementasi dari fungsi utama ini ada di baris 645 sampai dengan 918,
30 dengan proses-proses internalnya meliputi langkah-langkah berikut.

- 31 1. **Baris 646–827:** Implementasi `getopt-long` dan pentransferan argumen ke dalam variabel-
32 variabel internal perkakas.
- 33 2. **Baris 647 & 648:** Inisialisasi `opterr` dengan nilai 0, untuk mencegah `getopt-long` menge-
34 luarkan pesan *error default*-nya untuk opsi-opsi yang tidak diketahui ('?').
- 35 3. **Baris 830–848:** Pengecekan kelebihan argumen dalam perintah masukan.
- 36 4. **Baris 850–915:** Penentuan langkah-langkah yang harus ditempuh untuk setiap kemungkinan
37 mode operasional.

1 5. **Baris 917:** *Return code 0*, menandakan bahwa perkakas berhasil berjalan tanpa ada masalah.

2 5.1.13 CMakeLists

3 Cara kerja file CMakeLists untuk perkakas ini dapat dibagi menjadi beberapa bagian sebagai
4 berikut:

5. **Baris 1 & 2:** Pengaturan nama proyek serta file-file utama dari proyeknya.
6. **Baris 4:** Pengaturan versi minimum CMake. Versi CMake yang dibutuhkan minimal adalah
7 3.18, karena fitur *file compression* baru didukung di versi tersebut.
8. **Baris 5:** Pendefinisian nama proyek, versi, serta bahasa pemrograman dari proyek tersebut.
9. **Baris 7:** Penghubungan file-file utama proyek ke proyek yang sudah didefinisikan di baris 5.
10. **Baris 9–14:** Pencarian dan pengintegrasian *library-library* yang diperlukan perkakas.
11. **Baris 16–32:** Perintah-perintah spesifik untuk sistem operasi berbasis Linux, yang berhubu-
12 ungan dengan instalasi halaman manual (*man page*). File yang berisi kode murni *man page*
13 akan di-*compress*, dan kemudian diatur untuk dapat diinstal langsung ke direktori yang benar
14 di Linux.

15 Implementasi lengkap dari file CMakeLists.txt ini dapat dilihat di lampiran [A.1](#).

16 5.1.14 Halaman manual (*man page*)

17 Kode ini diperlukan untuk fitur halaman manual (*man page*) di sistem operasi berbasis Linux,
18 yang pada dasarnya merupakan bantuan penggunaan perkakas yang lebih panjang dan lebih detail.
19 Implementasi dari perkakas ini dapat dibagi menjadi bagian-bagian sebagai berikut, berdasarkan
20 sintaks yang mengawali bagian-bagian dalam kode tersebut, yaitu:

- 21. • **.TH**
22 Judul dari halaman manual. Tanggal dibuat, versi, serta judul dari halaman manualnya
23 sendiri diatur di bagian ini.
- 24. • **.SH**
25 Setiap baris yang diawali dengan sintaks ini akan di-format menjadi judul dari tiap-tiap bab
26 di dalam halaman manual.
- 27. • Sisa dari file yang tidak diawali dengan kedua perintah di atas merupakan deskripsi dari
28 tiap-tiap bagian yang ada di dalam halaman manual nantinya.

29 Adapun implementasi lengkap dari kode halaman manual ini ada di lampiran [A.3](#).

30 5.2 Pengujian

31 Bagian ini akan menjelaskan hal-hal yang seputar pengujian perkakas yang telah dibuat—lingkungan
32 pengujian, cara instalasi dan penggunaan perkakas, serta pengujian fungsional melalui berbagai
33 macam kasus tes (*test case*).

34 5.2.1 Lingkungan Perangkat Keras

35 Berikut merupakan spesifikasi perangkat keras yang digunakan dalam pengujian perkakas ini:

- 36. • *Processor*: Intel® Core™ i5-10300H @ 2.50 GHz

- 1 • RAM: 8 GB
- 2 • *Hard disk*: SSD 512 GB (NVMeTM M.2)
- 3 • Perangkat keras pendukung: Keyboard

4 5.2.2 Lingkungan Perangkat Lunak

5 Berikut merupakan spesifikasi perangkat lunak yang digunakan dalam pengujian perkakas ini:

- 6 • Windows:
 - 7 – OS: Windows 10 Home Single Language (64-bit)
 - 8 – *Compiler*: MinGW (GNU GCC—versi 12.1.0)
 - 9 – *Library*:
 - 10 * curl (versi 7.83.1)
 - 11 * cmake (versi 3.24.1)
- 12 • Linux:
 - 13 – OS: Ubuntu Jammy (22.04)
 - 14 – *Compiler*: GNU GCC—versi 11.3.0
 - 15 – *Library*:
 - 16 * curl (versi 7.81.0)
 - 17 * cmake (versi 3.22.1)

18 5.2.3 Pembangunan dan Instalasi

19 Syarat Instalasi

20 Instalasi perkakas ini tentunya mengharuskan *library-library* yang telah dibahas untuk diinstal
21 terlebih dahulu. Karena banyaknya perbedaan dari detil-detil yang ada di dalam persyaratan
22 instalasi untuk kedua sistem operasi yang didukung, maka bagian ini akan dibagi dua, menjadi satu
23 bagian per sistem operasi.

- 24 • Windows

25 Untuk sistem operasi Windows, pengguna perlu menginstal hal-hal berikut:

- 26 – vcpkg
- 27 – cURL
- 28 – CMake

29 Perlu diperhatikan bahwa cURL (di Windows) harus diinstal melalui vcpkg—cURL bawaan
30 dari Windows tidak mengandung *library-library development* sekunder yang dibutuhkan oleh
31 perkakas ini. Di lain hal, instalasi cURL secara manual hanya memungkinkan perkakas
32 cURL-nya sendiri untuk diakses dari mana saja (melalui variabel *environment*), tetapi hal ini
33 tidak berlaku untuk *library development* sekundernya.

- 34 • Linux

35 Untuk sistem operasi berbasis Linux, pengguna perlu menginstal hal-hal berikut:

- 36 – cURL
- 37 – CMake
- 38 – libcurl4-openssl-dev
- 39 – GNU Make (opsional)

- 1 Ingat bahwa perkakas ini juga menggunakan *library* cJSON, tetapi untuk alasan kompatibilitas
- 2 antar Windows dan Linux, *source code* dari *library* ini langsung diikutkan di dalam perkakasnya
- 3 sendiri, sehingga tidak perlu diinstal oleh pengguna lagi.

4 Cara Instalasi

- 5 Untuk memakai perkakasnya sendiri, perkakas ini perlu dibangun dan diinstal terlebih dahulu.
- 6 Berikut merupakan langkah-langkah yang perlu diambil untuk seluruh proses tersebut.

- 7 1. Buka *folder* “build” di dalam *folder* perkakas.
- 8 2. Buka *terminal/command prompt* di dalam *folder* tersebut.
- 9 3. Sesuai dengan sistem operasi tempat perkakas akan digunakan, ketik dan jalankan perintah
berikut di *terminal*:

- 10 • Windows:

```
12 cmake -DCMAKE_BUILD_TYPE:STRING=Release -DCMAKE_TOOLCHAIN_FILE=<direktori  
13 file toolchain vcpkg>" -G "<compiler>" ../
```

- 14 • Linux:

```
15 cmake ../
```

16 Untuk apa yang harus menggantikan variabel <compiler> dapat dilihat dengan perintah `cmake --help`. Daftar *compiler* yang didukung oleh *cmake* dapat dilihat di bagian “Generators”, dan pengguna tinggal menyesuaikan dengan *compiler* yang telah diinstal sebelumnya. Ada beberapa hal yang perlu dijelaskan/diperhatikan untuk langkah ini.

- 20 • Windows

- 21 – Opsi `-DCMAKE_TOOLCHAIN_FILE` merupakan metode pengintegrasian *vcpkg* untuk
22 proyek CMake. Untuk direktori persisnya (dan sintaks lengkap dari opsi ini) dapat
23 dilihat setelah langkah “Using *vcpkg* with MSBuild/Visual Studio” di halaman
24 panduan instalasi *vcpkg*.¹
- 25 – Direkomendasikan untuk menginstal *compiler* **MinGW**, karena *compiler* ini sudah
26 mengikutkan salah satu file *header* yang dibutuhkan oleh perkakas ini. Apabila
27 pengguna menggunakan *compiler* ini, variabel <compiler> harus diisi dengan
28 “**MinGW Makefiles**”.
- 29 – **Jangan menggunakan *compiler* Visual Studio**, karena *compiler* ini tidak
30 mengandung file *header* C yang dibutuhkan di perkakas ini. Perlu diperhatikan juga
31 bahwa *compiler* Visual Studio ini merupakan nilai *default* dari <compiler> untuk
32 sistem operasi Windows, jadi pengguna juga tidak boleh menghilangkan opsi `-G`
33 tersebut begitu saja.

- 34 • Linux

35 Untuk sistem operasi berbasis Linux, tidak perlu mengatur *compiler*, karena nilai *default*
36 dari variabel <compiler> di sistem operasi berbasis Linux (**Unix Makefiles**) sudah
37 ideal.

- 38 4. Lanjutkan dengan instalasi perkakas.

¹<https://vcpkg.io/en/getting-started.html>

- 1 • Windows:

2 Jalankan perintah berikut.

3 `cmake --build .`

- 4 • Linux:

5 Jalankan kedua perintah berikut.

6 `cmake --build .`

7 `cmake --install .`

8 Jika **GNU Make** terinstal di perangkat pengguna, maka kedua perintah ini dapat
9 digantikan dengan perintah berikut.

10 `make install`

11 Jika terjadi *error permission*, cukup tambahkan perintah `sudo` di depan perintah yang
12 ingin dijalankan.

- 13 5. File *executable* akan terletak di dalam *folder* “build”, dan siap dijalankan.

14 5.2.4 Pengujian

15 Pengujian akan dilakukan untuk setiap fitur untuk memeriksa apakah semua fitur perkakas sudah
16 berfungsi sebagaimanamestinya, serta semua kemungkinan *error* yang ada sudah diatasi dengan
17 benar. Perlu ditekankan bahwa pengujian berikut juga akan dilakukan dengan versi panjang dari
18 opsi-opsi yang ada di dalam perintah (misal `-h` diganti menjadi `--help`). Akan tetapi, untuk alasan
19 keringkasan dokumen, kecuali terjadi kegagalan, tes-tes tersebut tidak akan dicatat.

20 Tabel 5.1 memaparkan jumlah tes yang akan dilakukan. Adapun penjelasan dari apa persisnya
21 yang akan dites (*scope*) untuk setiap objek tes akan dibahas langsung di tiap-tiap baginya.

No.	Objek tes	Jumlah tes
1	Sintaks dasar	7
2	Mode bantuan (<code>--help</code>)	3
3	Mode <i>searchplace</i>	9
4	Mode <i>findroute</i>	8
5	Mode <i>direct</i>	10
6	Integrasi perkakas <i>command line</i> lain	6

Tabel 5.1: Jumlah kategori dan tes yang dilakukan.

22 Sintaks dasar

23 Pengujian ini akan dilakukan untuk mengecek apakah perkakas akan merespon terhadap masukan
24 yang sama sekali tidak sesuai dengan apa yang diharapkan oleh perkakas. Beberapa dari kasus-
25 kasus berikut sudah meliputi kemungkinan-kemungkinan kesalahan masukan untuk fitur-fitur yang
26 disediakan perkakas, jadi tes spesifik per fitur nantinya tidak akan mengikutkan pengujian sintaks
27 lagi.

- 28 1. Perintah tanpa opsi

- Perintah masukan:

30 1 `kiritoool`

- 1 • Keluaran yang diharapkan:
2 Perkakas akan mengeluarkan pesan *error* yang mengingatkan pengguna untuk mema-
3 sukkan mode operasional perkakas.

- 4 • Keluaran perkakas:

5 1 Error:
6 2 Mohon masukkan mode pengunaan perkakas.

- 9 • Status tes: **Sukses**

10 2. Perintah dengan satu atau lebih opsi tidak valid

- 11 • Perintah masukan:

12 1 kiritool --mode searchplace --region bdo --query unpar --language id

- 15 • Keluaran yang diharapkan:

16 Perkakas akan mengeluarkan pesan *error* yang memberi tahu pengguna bahwa ada opsi
17 yang tidak valid di dalam perintah masukan.

- 18 • Keluaran perkakas:

19 1 Error:
20 2 Anda telah memasukkan opsi yang tidak valid.
21 3 Mohon periksa kembali penulisan perintah yang anda masukkan.

- 24 • Status tes: **Sukses**

25 3. Perintah tanpa argumen di akhir perintah

- 26 • Perintah masukan:

27 1 kiritool --mode searchplace --region bdo --query unpar --locale

- 30 • Keluaran yang diharapkan:

31 Perkakas akan mengeluarkan pesan *error* mengenai adanya opsi yang kehilangan argu-
32 mennya di dalam perintah masukan.

- 33 • Keluaran perkakas:

34 1 Error:
35 2 Salah satu dari opsi yang anda masukkan kehilangan argumen yang dibutuhkan.
36 3 Mohon periksa kembali penulisan perintah yang anda masukkan.

- 39 • Status tes: **Sukses**

40 4. Perintah tanpa argumen di tengah perintah

- 41 • Perintah masukan:

42 1 kiritool --mode searchplace --region bdo --query --locale id

- 45 • Keluaran yang diharapkan:

46 Perkakas akan mengeluarkan pesan *error* mengenai kelebihan argumen yang dimasukkan
47 pengguna, serta mendaftarkan argumen apa saja yang berlebih.

- 48 • Keluaran perkakas:

49 1 Error:
50 2 Anda telah memasukkan kelebihan argumen: id
51 3 Mohon periksa kembali penulisan perintah yang anda masukkan.

- 54 • Status tes: **Sukses**

- 55 • Catatan tambahan:

56 Kasus ini idealnya berakhir dengan pesan *error* yang menyatakan bahwa ada opsi yang

kehilangan argumennya. Akan tetapi, akibat batasan teknis, *library* getopt sendiri tidak bisa menangani kasus di mana opsi yang kehilangan argumennya berada di tengah perintah, karena getopt akan menginterpretasikan opsi selanjutnya sebagai argumen dari opsi yang kehilangan argumennya. Satu-satunya solusi yang mungkin adalah mengecek apakah argumen dimulai dengan karakter tanda hubung ('-'), tetapi solusi ini tidak dapat diimplementasikan, karena argumen dari beberapa opsi berpotensi untuk diawali dengan karakter tersebut.

5. Perintah dengan terlalu banyak argumen

- Perintah masukan:

```
1 kiritoor --mode searchplace --region bdo --query unpar --locale id en
```

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan pesan *error* mengenai kelebihan argumen yang dimasukkan pengguna, serta mendaftarkan argumen apa saja yang berlebih.

- Keluaran perkakas:

```
1 Error:  
2 Anda telah memasukkan kelebihan argumen: en  
3 Mohon periksa kembali penulisan perintah yang anda masukkan.
```

- Status tes: **Sukses**

6. Perintah dengan mode yang tidak valid

- Perintah masukan:

```
1 kiritoor -m help
```

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa mode yang dimasukkan tidak valid.

- Keluaran perkakas:

```
1 Error:  
2 Anda telah memasukkan mode yang tidak valid.  
3 Mohon periksa kembali apakah mode yang anda masukkan sudah diketik dengan benar.
```

- Status tes: **Sukses**

7. Pengunaan banyak mode sekaligus

- Perintah masukan:

```
1 kiritoor --mode searchplace --region bdo --query unpar --help --locale en
```

- Keluaran yang diharapkan:

Perkakas hanya akan merespon terhadap mode operasional pertama yang dimasukkan (beserta opsi-opsinya).

- Keluaran perkakas:

```
1 Location:  
2 -----  
3 Name: Universitas Katolik Parahyangan  
4 Coordinates: -6.87520,107.60492
```

- Status tes: **Sukses**

1 Mode bantuan

2 Pengujian ini akan dilakukan untuk mengecek apakah fungsi-fungsi perkakas yang berhubungan
3 dengan fitur bantuan penggunaan perkakas sudah berfungsi dengan baik.

4 1. Panggilan bantuan normal

- 5 • Perintah masukan:

```
6   1 kiritool --help
```

- 9 • Keluaran yang diharapkan:

10 Perkakas akan mengeluarkan bantuan penggunaan perkakas.

- 11 • Keluaran perkakas:

12 Perkakas akan mengeluarkan bantuan penggunaan perkakas dengan benar. Keluaran
13 dari kasus tes ini dapat dilihat di lampiran [A.4](#).

- 14 • Status tes: **Sukses**

15 2. Panggilan bantuan dengan tambahan opsi valid yang tidak relevan

- 16 • Perintah masukan:

```
17   1 kiritool --help --start unpar
```

- 20 • Keluaran yang diharapkan:

21 Perkakas akan mengeluarkan bantuan penggunaan perkakas, tanpa memerlukan opsi
22 serta argumen tambahan di dalam perintah.

- 23 • Keluaran perkakas:

24 Keluaran dari kasus tes ini sama seperti kasus sebelumnya, yang juga dapat dilihat di
25 lampiran [A.4](#).

- 26 • Status tes: **Sukses**

27 3. Pemanggilan *manual page* (khusus Linux)

- 28 • Perintah masukan:

```
29   1 man kiritool
```

- 32 • Keluaran yang diharapkan:

33 Terminal akan menampilkan manual page dari perkakas yang sesuai.

- 34 • Keluaran perkakas:

35 Terminal menampilkan manual page yang sesuai. Hasil dari kasus tes ini dapat dilihat di
36 lampiran [A.5](#). Perlu diingat bahwa *man page* memiliki *formatting* otomatisnya sendiri,
37 jadi ketika dikonversi ke file .txt (agar bisa dimasukkan sebagai lampiran), beberapa
38 aspek *formatting*-nya menjadi tidak sempurna.

- 39 • Status tes: **Sukses**

40 Mode *searchplace*

41 Pengujian ini akan dilakukan untuk mengecek apakah fungsi-fungsi perkakas yang berhubungan
42 dengan fitur pencarian lokasi sudah berfungsi dengan baik.

43 1. Pencarian lokasi sukses dengan satu hasil

- 44 • Perintah masukan:

```
45   1 kiritool --mode searchplace --region bdo --query unpar --locale id
```

- 1 • Keluaran yang diharapkan:

2 Perkakas akan menampilkan nama dan koordinat *latitude* dan *longitude* lokasi, sesuai
3 dengan bahasa yang diminta oleh pengguna.

- 4 • Keluaran perkakas:

```
5     1 Lokasi:  
6     2 -----  
7     3 Nama: Universitas Katolik Parahyangan  
8     4 Koordinat: -6.87520,107.60492
```

- 11 • Status tes: **Sukses**

12 2. Pencarian lokasi sukses dengan lebih dari satu hasil

- 13 • Perintah masukan:

```
14     1 kiritool --mode searchplace --region bdo --query ab --locale id
```

- 17 • Keluaran yang diharapkan:

18 Perkakas akan menampilkan nama dan koordinat *latitude* dan *longitude* semua kemung-
19 kinan lokasi, sesuai dengan bahasa yang diminta oleh pengguna.

- 20 • Keluaran perkakas:

```
21     1 Lokasi 1:  
2     2 -----  
24     3 Nama: Blk. A-B  
25     4 Koordinat: -6.88612,107.65028  
26     5  
27     6 Lokasi 2:  
28     7 -----  
29     8 Nama: Blk. A-B  
30     9 Koordinat: -6.90845,107.67641  
31     10  
32     11 Lokasi 3:  
33     12 -----  
34     13 Nama: Blk. AB  
35     14 Koordinat: -6.89748,107.70782
```

- 37 • Status tes: **Sukses**

38 3. Pencarian lokasi gagal

- 39 • Perintah masukan:

```
40     1 kiritool --mode searchplace --region bdo --query abasdasd --locale id
```

- 43 • Keluaran yang diharapkan:

44 Perkakas akan mengeluarkan pesan keluaran yang memberitahu pengguna bahwa lokasi
45 tidak berhasil ditemukan.

- 46 • Keluaran perkakas:

```
47     1 Lokasi tidak berhasil ditemukan.  
48     2 Silakan cek ulang apakah kata kunci pencarian sudah dimasukkan dengan benar.
```

- 51 • Status tes: **Sukses**

52 4. Pencarian lokasi tanpa opsi region

- 53 • Perintah masukan:

```
54     1 kiritool --mode searchplace --query unpar --locale id
```

- 57 • Keluaran yang diharapkan:

58 Perkakas akan menampilkan pesan *error* yang mengingatkan pengguna untuk mema-
59 sukkan kode region.

- 60 • Keluaran perkakas:

```
1 Error:  
2 Fitur pencarian lokasi memerlukan pengaturan region lokasi yang ingin dicari.  
3 Mohon pastikan anda sudah memasukkan salah satu dari empat kode region yang tersedia.  
4 Pilihan region: cgk, bdo, mlg, sub
```

- Status tes: **Sukses**

5. Pencarian lokasi dengan region yang tidak valid

- Perintah masukan:

```
1 kiritool --mode searchplace --region bdg --query unpar --locale id
```

- Keluaran yang diharapkan:

Perkakas akan menampilkan pesan *error* bahwa region yang dimasukkan tidak valid.

- Keluaran perkakas:

```
1 Error:  
2 Anda telah memasukkan region yang tidak valid.  
3 Mohon periksa kembali apakah kode region yang anda masukkan merupakan salah satu dari empat kode region yang tersedia.  
4 Pilihan region: cgk, bdo, mlg, sub
```

- Status tes: **Sukses**

6. Pencarian lokasi tanpa kata kunci pencarian

- Perintah masukan:

```
1 kiritool --mode searchplace --region bdo --locale id
```

- Keluaran yang diharapkan:

Perkakas akan menampilkan pesan *error* yang mengingatkan pengguna untuk memasukkan kata kunci pencarian.

- Keluaran perkakas:

```
1 Error:  
2 Fitur pencarian lokasi memerlukan sebuah kata kunci pencarian.  
3 Mohon pastikan anda sudah memasukkan kata kunci untuk melakukan pencarian lokasi.
```

- Status tes: **Sukses**

7. Pencarian lokasi dengan kata kunci pencarian yang tidak valid

- Perintah masukan:

```
1 kiritool --mode searchplace --region bdo --query -6.87520,107.60492 --locale id
```

- Keluaran yang diharapkan:

Perkakas akan menampilkan pesan bahwa keluaran API adalah sebuah *error*.

- Keluaran perkakas:

```
1 Error:  
2 API mengembalikan error sebagai keluarannya.  
3 Silakan cek ulang apakah kata kunci pencarian sudah diformat dengan benar.
```

- Status tes: **Sukses**

- Catatan tambahan:

Perkakas tetap menerima kata kunci pencarian yang diawali dengan tanda hubung ('-'), jadi perkakas tidak bisa langsung mengembalikan *error* ke pengguna apabila pengguna (misal dalam kasus ini) memasukkan koordinat *latitude* dan *longitude* sebagai argumen opsi *-q*.

8. Pencarian lokasi tanpa pengaturan bahasa

- 1 • Perintah masukan:

2 1 kiritool --mode searchplace --region bdo --query unpar

- 5 • Keluaran yang diharapkan:

6 Perkakas tetap berfungsi seperti biasa, dengan mengeluarkan keluaran dalam bahasa
7 Indonesia.

- 8 • Keluaran perkakas:

9 1 Lokasi:
10 2 -----
11 3 Nama: Universitas Katolik Parahyangan
12 4 Koordinat: -6.87520,107.60492

- 15 • Status tes: **Sukses**

16 9. Pencarian lokasi dengan pengaturan bahasa yang tidak valid

- 17 • Perintah masukan:

18 1 kiritool --mode searchplace --region bdo --query unpar --locale ch

21 Perkakas akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa bahasa
22 yang dimasukkan tidak valid.

- 23 • Keluaran perkakas:

24 1 Error:
25 2 Anda telah memasukkan pilihan bahasa (locale) yang tidak valid.
26 3 Mohon periksa kembali apakah pilihan bahasa yang anda masukkan valid atau tidak.
27 4 Pilihan locale: id, en
28 5 -----
29 6 You have inputted an invalid language (locale) option.
30 7 Please recheck whether the language code you inserted was supported or not.
31 8 Locale available: id, en

- 34 • Status tes: **Sukses**

35 **Mode *findroute***

36 Pengujian ini akan dilakukan untuk mengecek apakah fungsi-fungsi perkakas yang berhubungan
37 dengan fitur pencarian rute angkot sudah berfungsi dengan baik.

38 1. Pencarian rute sukses dengan satu kemungkinan rute

- 39 • Perintah masukan:

40 1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.87520,107.60492 --locale id

- 43 • Keluaran yang diharapkan:

44 Perkakas akan menampilkan estimasi durasi rute, serta langkah-langkah yang perlu
45 ditempuh di dalam rutennya, sesuai dengan bahasa yang diminta oleh pengguna.

- 46 • Keluaran perkakas:

47 1 Rute:
48 2 -----
49 3 Estimasi waktu: 25 menit
50 4 -----
51 5 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 203 meter.
52 6 Langkah 2: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
53 3,9 kilometer.
54 7 Langkah 3: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.

- 57 • Status tes: **Sukses**

58 2. Pencarian rute sukses dengan satu kemungkinan rute yang sangat jauh

- 1 • Perintah masukan:

2 1 kiritool --mode findroute --start -6.16935,106.78899 --finish -6.87520,107.60492 --locale id

- 5 • Keluaran yang diharapkan:

6 Jauhnya rute tidak akan mempengaruhi kinerja dari perkakas.

- 7 • Keluaran perkakas:

8 1 Rute:
 9 2 -----
 10 3 Estimasi waktu: 3 jam
 11 4 -----
 12 5 Langkah 1: Jalan dari tujuan akhir Anda ke Jelambar 1 sejauh kurang lebih 365 meter.
 13 6 Langkah 2: Naik Daytrans Grogol - Cihampelas di Jelambar 1, dan turun di SMA Pasundan 2 Bandung kurang lebih setelah 155,8 kilometer.
 14 7 Langkah 3: Jalan dari SMA Pasundan 2 Bandung ke Jalan Cihampelas sejauh kurang lebih 35 meter.
 15 8 Langkah 4: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah 3,9 kilometer.
 16 9 Langkah 5: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.

- 21 • Status tes: **Sukses**

22 3. Pencarian rute sukses dengan lebih dari satu kemungkinan rute

- 23 • Perintah masukan:

24 1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.91527,107.59454 --locale id

- 27 • Keluaran yang diharapkan:

28 Perkakas akan menampilkan setiap rutenya, diikuti dengan estimasi waktunya serta
 29 langkah-langkah yang perlu ditempuh di dalam rutenya, sesuai dengan bahasa yang
 30 diminta oleh pengguna.

- 31 • Keluaran perkakas:

32 1 Rute 1:
 33 2 -----
 34 3 Estimasi waktu: 30 menit
 35 4 -----
 36 5 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 56 meter.
 37 6 Langkah 2: Naik angkot Cicaheum - Ciroyom di Jalan Cihampelas, dan turun di Jalan Arjuna kurang lebih setelah 4,2
 38 kilometer.
 39 7 Langkah 3: Jalan sedikit di Jalan Arjuna.
 40 8 Langkah 4: Naik angkot Dago - Caringin di Jalan Arjuna, dan turun di Jalan Kebon Jati kurang lebih setelah 688 meter
 41 .
 42 9 Langkah 5: Jalan dari Jalan Kebon Jati ke lokasi mulai Anda sejauh kurang lebih 138 meter.
 43
 44 11 Rute 2:
 45 12 -----
 46 13 Estimasi waktu: 25 menit
 47 14 -----
 48 15 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 26 meter.
 49 16 Langkah 2: Naik angkot Cisitu - Tegallega di Jalan Cihampelas, dan turun di Jalan Pasir Kaliki kurang lebih setelah
 50 3,7 kilometer.
 51 17 Langkah 3: Jalan dari Jalan Pasir Kaliki ke lokasi mulai Anda sejauh kurang lebih 396 meter.

- 54 • Status tes: **Sukses**

55 4. Pencarian rute gagal

- 56 • Perintah masukan:

57 1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.88307,107.65529 --locale id

- 60 • Keluaran yang diharapkan:

61 Perkakas akan mengeluarkan pesan kepada pengguna bahwa rute perjalanan dengan
 62 angkot tidak berhasil ditemukan.

- 63 • Keluaran perkakas:

64 1 Maaf, kami tidak dapat menemukan rute transportasi publik untuk perjalanan anda.

- 1 • Status tes: **Sukses**

2 5. Pencarian rute tanpa masukan lokasi (2 kasus)

- 3 • Perintah masukan:

```
4         1 kiritool --mode findroute --finish -6.87520,107.60492 --locale id  
5         2 kiritool --mode findroute --start -6.89350,107.60430 --locale id
```

- 6 • Keluaran yang diharapkan:

7 Perkakas akan mengeluarkan pesan *error* yang mengingatkan pengguna untuk memasukkan koordinat *latitude* dan *longitude* lokasi awal atau akhir, tergantung dari lokasi mana yang tidak dimasukkan.

- 8 • Keluaran perkakas:

9 **Kasus 1:** Lokasi awal tidak dimasukkan

```
10          1 Error:  
11          2 Anda belum memasukkan sebuah koordinat untuk lokasi awal.  
12          3 Mohon masukkan koordinat untuk lokasi awal pencarian rute melalui opsi yang sesuai.
```

13 **Kasus 2:** Lokasi akhir tidak dimasukkan

```
14          1 Error:  
15          2 Anda belum memasukkan sebuah koordinat untuk lokasi akhir.  
16          3 Mohon masukkan koordinat untuk lokasi akhir pencarian rute melalui opsi yang sesuai.
```

- 17 • Status tes: **Sukses**

18 6. Pencarian rute dengan koordinat lokasi yang tidak valid

- 19 • Perintah masukan:

```
20         1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.87520,107.6049a --locale id
```

- 21 • Keluaran yang diharapkan:

22 Perkakas akan mengeluarkan pesan bahwa keluaran API adalah sebuah *error*.

- 23 • Keluaran perkakas:

```
24          1 Error:  
25          2 API mengembalikan error sebagai keluarannya.  
26          3 Silakan cek ulang apakah koordinat kedua lokasi sudah dimasukkan dengan benar.
```

- 27 • Status tes: **Sukses**

- 28 • Catatan tambahan:

29 Akibat penggunaan variabel yang sama dengan variabel dalam fitur **direct**, di mana di dalam fitur tersebut opsi **-s** dan opsi **-f** memiliki aturan masukan yang berbeda, maka fitur ini tidak akan langsung mendeteksi *error* apabila pengguna memasukkan koordinat yang tidak valid.

30 7. Pencarian lokasi tanpa pengaturan bahasa

- 31 • Perintah masukan:

```
32         1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.87520,107.60492
```

- 33 • Keluaran yang diharapkan:

34 Perkakas tetap berfungsi seperti biasa, dengan mengeluarkan keluaran dalam bahasa Indonesia.

- 35 • Keluaran perkakas:

```
36          1 Rute:  
37          2 -----  
38          3 Estimasi waktu: 25 menit
```

```

1   -----
2   5 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 203 meter.
3   6 Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
4   3,9 kilometer.
5   7 Langkah 3: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.

```

- Status tes: **Sukses**

8. Pencarian lokasi dengan pengaturan bahasa yang tidak valid

- Perintah masukan:

```
1 kiritool --mode findroute --start -6.89350,107.60430 --finish -6.87520,107.60492 --locale ch
```

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa bahasa yang dimasukkan tidak valid.

- Keluaran perkakas:

```

1 Error:
2 Anda telah memasukkan pilihan bahasa (locale) yang tidak valid.
3 Mohon periksa kembali apakah pilihan bahasa yang anda masukkan valid atau tidak.
4 Pilihan locale: id, en
5 -----
6 You have inputted an invalid language (locale) option.
7 Please recheck whether the language code you inserted was supported or not.
8 Locale available: id, en

```

- Status tes: **Sukses**

28 Mode *direct*

Pengujian ini akan dilakukan untuk mengecek apakah fungsi-fungsi perkakas yang berhubungan dengan fitur pencarian rute angkot langsung dari kata kunci lokasi awal dan akhir sudah berfungsi dengan baik.

1. Pencarian rute langsung sukses dengan satu kemungkinan rute

- Perintah masukan:

```
1 kiritool --mode direct --restart bdo --start ciwalk --regfinish bdo --finish unpar --locale id
```

- Keluaran yang diharapkan:

Perkakas akan menampilkan nama lokasi awal dan akhir, estimasi durasi rute, serta langkah-langkah yang perlu ditempuh di dalam rutenya, sesuai dengan bahasa yang diminta oleh pengguna.

- Keluaran perkakas:

```

1 Lokasi awal: Cihampelas Walk Extention
2 Lokasi akhir: Universitas Katolik Parahyangan
3
4 Rute:
5 -----
6 Estimasi waktu: 25 menit
7 -----
8 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 203 meter.
9 Langkah 2: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
3,9 kilometer.
10 Langkah 3: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.

```

- Status tes: **Sukses**

2. Pencarian rute langsung sukses dengan satu kemungkinan rute yang sangat jauh

- Perintah masukan:

```
1 kiritool --mode direct --restart cgk --start untar --regfinish bdo --finish unpar --locale id
```

- 1 • Keluaran yang diharapkan:
2 Jauhnya rute tidak akan berpengaruh ke keluaran perkakas.
- 3 • Keluaran perkakas:

```

4   1 Lokasi awal: Universitas Tarumanagara
5   2 Lokasi akhir: Universitas Katolik Parahyangan
6
7
8   4 Rute:
9   5 -----
10  6 Estimasi waktu: 3 jam
11  7 -----
12  8 Langkah 1: Jalan dari tujuan akhir Anda ke Jelambar 1 sejauh kurang lebih 365 meter.
13  9 Langkah 2: Naik Daytrans Grogol - Cihampelas di Jelambar 1, dan turun di SMA Pasundan 2 Bandung kurang lebih setelah
14    155,8 kilometer.
15  10 Langkah 3: Jalan dari SMA Pasundan 2 Bandung ke Jalan Cihampelas sejauh kurang lebih 35 meter.
16  11 Langkah 4: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
17    3,9 kilometer.
18  12 Langkah 5: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.

```

- 20 • Status tes: **Sukses**

21 3. Pencarian rute langsung sukses dengan lebih dari satu kemungkinan rute

- 22 • Perintah masukan:

```
23   1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish paskal --locale id
```

- 26 • Keluaran yang diharapkan:

27 Perkakas akan menampilkan nama lokasi awal dan akhir, serta tiap-tiap kemungkinan
28 rutanya, dimulai dari estimasi durasi, diikuti dengan langkah-langkah yang perlu ditempuh
29 di dalamnya, sesuai dengan bahasa yang diminta oleh pengguna.

- 30 • Keluaran perkakas:

```

31   1 Lokasi awal: Cihampelas Walk Extention
32   2 Lokasi akhir: 23 Paskal Shopping Center
33
34
35   4 Rute 1:
36   5 -----
37   6 Estimasi waktu: 30 menit
38   7 -----
39   8 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 56 meter.
40   9 Langkah 2: Naik angkot Cicahem - Ciroyom di Jalan Cihampelas, dan turun di Jalan Arjuna kurang lebih setelah 4,2
41    kilometer.
42  10 Langkah 3: Jalan sedikit di Jalan Arjuna.
43  11 Langkah 4: Naik angkot Dago - Caringin di Jalan Arjuna, dan turun di Jalan Kebon Jati kurang lebih setelah 688 meter
44    .
45  12 Langkah 5: Jalan dari Jalan Kebon Jati ke lokasi mulai Anda sejauh kurang lebih 138 meter.
46
47
48   14 Rute 2:
49   15 -----
50   16 Estimasi waktu: 25 menit
51   17 -----
52   18 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 26 meter.
53   19 Langkah 2: Naik angkot Cisitu - Tegallega di Jalan Cihampelas, dan turun di Jalan Pasir Kaliki kurang lebih setelah
54    3,7 kilometer.
55  20 Langkah 3: Jalan dari Jalan Pasir Kaliki ke lokasi mulai Anda sejauh kurang lebih 396 meter.

```

- 56 • Status tes: **Sukses**

57 4. Pencarian rute langsung gagal

- 58 • Perintah masukan:

```
59   1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish paku --locale id
```

- 62 • Keluaran yang diharapkan:

63 Perkakas akan menampilkan nama lokasi awal dan akhir, tetapi akan mengeluarkan
64 pesan bahwa rute tidak berhasil ditemukan.

- 65 • Keluaran perkakas:

```

1 Lokasi awal: Cihampelas Walk Extention
2 Lokasi akhir: Curug Paku
3
4
5 Maaf, kami tidak dapat menemukan rute transportasi publik untuk perjalanan anda.

```

- Status tes: **Sukses**

5. Pencarian rute langsung tanpa region lokasi (2 kasus)

- Perintah masukan:

```

1 kiritool --mode direct --start ciwalk --regfinish bdo --finish unpar --locale id
2 kiritool --mode direct --restart bdo --start ciwalk --finish unpar --locale id

```

- Keluaran yang diharapkan:

Perkakas akan menampilkan nama lokasi hingga lokasi yang tidak diberikan regionnya, dan kemudian akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa ada kata kunci pencarian lokasi yang hilang.

- Keluaran perkakas:

Kasus 1: Region awal tidak dimasukkan

```

1 Error:
2 Fitur pencarian lokasi memerlukan pengaturan region lokasi yang ingin dicari.
3 Mohon pastikan anda sudah memasukkan salah satu dari empat kode region yang tersedia.
4 Pilihan region: cgk, bdo, mlg, sub

```

Kasus 2: Region akhir tidak dimasukkan

```

1 Lokasi awal: Cihampelas Walk Extention
2
3 Error:
4 Fitur pencarian lokasi memerlukan pengaturan region lokasi yang ingin dicari.
5 Mohon pastikan anda sudah memasukkan salah satu dari empat kode region yang tersedia.
6 Pilihan region: cgk, bdo, mlg, sub

```

- Status tes: **Sukses**

6. Pencarian rute langsung dengan region lokasi yang tidak valid (2 kasus)

- Perintah masukan:

```

1 kiritool --mode direct --restart bdg --start ciwalk --regfinish bdo --finish unpar --locale id
2 kiritool --mode direct --restart bdo --start ciwalk --regfinish bdg --finish unpar --locale id

```

- Keluaran yang diharapkan:

Perkakas akan menampilkan nama lokasi hingga lokasi yang tidak diberikan regionnya, dan kemudian akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa ada kata kunci pencarian lokasi yang hilang.

- Keluaran perkakas:

Kasus 1: Region awal tidak valid

```

1 Error:
2 Anda telah memasukkan region yang tidak valid.
3 Mohon periksa kembali apakah kode region yang anda masukkan merupakan salah satu dari empat kode region yang tersedia.
4 Pilihan region: cgk, bdo, mlg, sub

```

Kasus 2: Region akhir tidak valid

```

1 Lokasi awal: Cihampelas Walk Extention
2
3 Error:
4 Anda telah memasukkan region yang tidak valid.
5 Mohon periksa kembali apakah kode region yang anda masukkan merupakan salah satu dari empat kode region yang tersedia.
6 Pilihan region: cgk, bdo, mlg, sub

```

- 1 • Status tes: **Sukses**

2 7. Pencarian rute langsung tanpa kata kunci pencarian lokasi (2 kasus)

- 3 • Perintah masukan:

```
4         1 kiritool --mode direct --regstart bdo --regfinish bdo --finish unpar --locale id  
5         2 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --locale id
```

- 6 • Keluaran yang diharapkan:

7 Perkakas akan menampilkan nama lokasi hingga lokasi yang tidak diberikan kata kunci pencariannya, dan kemudian akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa ada kata kunci pencarian lokasi yang hilang.

- 8 • Keluaran perkakas:

9 **Kasus 1:** Lokasi awal tidak dimasukkan

```
10          1 Error:  
11          2 Fitur pencarian lokasi memerlukan sebuah kata kunci pencarian.  
12          3 Mohon pastikan anda sudah memasukkan kata kunci untuk melakukan pencarian lokasi.
```

13 **Kasus 2:** Lokasi akhir tidak dimasukkan

```
14          1 Lokasi awal: Cihampelas Walk Extention  
15          2 Error:  
16          3 Fitur pencarian lokasi memerlukan sebuah kata kunci pencarian.  
17          4 Mohon pastikan anda sudah memasukkan kata kunci untuk melakukan pencarian lokasi.
```

- 18 • Status tes: **Sukses**

19 8. Pencarian rute langsung dengan kata kunci pencarian lokasi yang tidak valid (2 kasus)

- 20 • Perintah masukan:

```
21         1 kiritool --mode direct --regstart bdo --start -6.89350,107.60430 --regfinish bdo --finish unpar --locale id  
22         2 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish -6.87520,107.60492 --locale id
```

- 23 • Keluaran yang diharapkan:

24 Perkakas akan menampilkan nama lokasi hingga lokasi yang tidak diberikan kata kunci pencariannya, dan kemudian akan mengeluarkan pesan yang memberitahu pengguna bahwa API mengembalikan sebuah *error*.

- 25 • Keluaran perkakas:

26 **Kasus 1:** Lokasi awal tidak valid

```
27          1 Error:  
28          2 API mengembalikan error sebagai koordinat lokasi awal.  
29          3 Silakan cek ulang apakah koordinat lokasi awal sudah dimasukkan dengan benar.
```

30 **Kasus 2:** Lokasi akhir tidak valid

```
31          1 Lokasi awal: Cihampelas Walk Extention  
32          2 Error:  
33          3 API mengembalikan error sebagai koordinat lokasi akhir.  
34          4 Silakan cek ulang apakah koordinat lokasi akhir sudah dimasukkan dengan benar.
```

- 35 • Status tes: **Sukses**

36 9. Pencarian rute langsung sukses tanpa pengaturan bahasa

- 37 • Perintah masukan:

```
38         1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar
```

- 39 • Keluaran yang diharapkan:

40 Perkakas tetap berfungsi seperti biasa, dengan mengeluarkan keluaran dalam bahasa Indonesia.

- 1 • Keluaran perkakas:

```

2     1 Lokasi awal: Cihampelas Walk Extention
3     2 Lokasi akhir: Universitas Katolik Parahyangan
4     3
5     4 Rute:
6     5 -----
7     6 Estimasi waktu: 25 menit
8     7 -----
9     8 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 203 meter.
10    9 Langkah 2: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
11    3,9 kilometer.
12    10 Langkah 3: Jalan dari 40141 ke lokasi mulai Anda sejauh kurang lebih 64 meter.
13

```

- 15 • Status tes: **Sukses**

16 10. Pencarian rute langsung sukses dengan pengaturan bahasa yang tidak valid

- 17 • Perintah masukan:

```
18    1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale ch
```

- 21 • Keluaran yang diharapkan:

22 Perkakas akan mengeluarkan pesan *error* yang memberitahu pengguna bahwa bahasa
23 yang dimasukkan tidak valid.

- 24 • Keluaran perkakas:

```

25    1 Error:
26    2 Anda telah memasukkan pilihan bahasa (locale) yang tidak valid.
27    3 Mohon periksa kembali apakah pilihan bahasa yang anda masukkan valid atau tidak.
28    4 Pilihan locale: id, en
29    5 -----
30    6 You have inputted an invalid language (locale) option.
31    7 Please recheck whether the language code you inserted was supported or not.
32    8 Locale available: id, en
33

```

- 35 • Status tes: **Sukses**

36 Integrasi perkakas *command line* lain

37 Pengujian ini dilakukan untuk menguji kompatibilitas keluaran perkakas dengan fungsi dari perkakas-
38 perkakas *command line* bawaan yang ada. Sebelum dilakukan, perlu ditetapkan beberapa aturan
39 dasar untuk pengujian ini, yaitu sebagai berikut:

- 40 • Tes kasus yang akan digunakan sebagai masukan adalah pencarian rute langsung dengan satu
41 kemungkinan rute, dari “ciwalk” (Bandung) ke “unpar” (Bandung), dalam bahasa Inggris.
42 Masukan ini akan disamakan untuk seluruh bagian ini untuk alasan konsistensi.
- 43 • Pengujian akan dilakukan sebanyak dua kali untuk tiap kasus, untuk Windows dan Linux.
- 44 • Pengujian hanya akan dilakukan sekali untuk kasus-kasus tertentu apabila fungsi yang diuji
45 memiliki perintah yang sama, atau fungsi tersebut tidak didukung oleh perkakas *command*
46 *line* bawaan dalam sistem operasi tersebut.

47 Berikut merupakan hasil pengujian untuk bagian ini.

- 48 1. Memasukkan keluaran perkakas ke dalam file output

- 49 • Perintah masukan:

```
50    1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en > out.txt
```

- 53 • Keluaran yang diharapkan:

54 Perkakas akan menuliskan keluaran dari perintah tersebut langsung di dalam file *out.txt*.

- 55 • Keluaran perkakas: Akan dihasilkan sebuah file *out.txt* yang isinya adalah sebagai
56 berikut.

```

1 Starting location: Cihampelas Walk Extention
2 Finish location: Universitas Katolik Parahyangan
3
4 Route:
5 -----
6 Estimated duration: 25 minutes
7 -----
8 Step 1: Walk about 203 meter from your starting point to Jalan Cihampelas.
9 Step 2: Take angkot Ciumbuleuit - St. Hall (belok) at Jalan Cihampelas, and alight at 40141 about 3.9 kilometer
10 later.
11 Step 3: Walk about 64 meter from 40141 to your destination.
12
13

```

- Status tes: **Sukses**

2. Mengeluarkan hanya langkah-langkah yang perlu ditempuh (Windows)

- Perintah masukan:

```

1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en | findstr /i
2 "step"

```

- Keluaran yang diharapkan:

Perkakas akan menampilkan hanya baris yang memaparkan durasi rute yang ingin ditempuh. Jika tes berhasil, keluaran untuk Windows maupun Linux (tes selanjutnya) akan sama.

- Keluaran perkakas:

```

1 Step 1: Walk about 203 meter from your starting point to Jalan Cihampelas.
2 Step 2: Take angkot Ciumbuleuit - St. Hall (belok) at Jalan Cihampelas, and alight at 40141 about 3.9 kilometer
3 later.
3 Step 3: Walk about 64 meter from 40141 to your destination.

```

- Status tes: **Sukses**

3. Mengeluarkan hanya langkah-langkah yang perlu ditempuh (Linux)

- Perintah masukan:

```

1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en | grep -i
2 "step"

```

- Keluaran yang diharapkan:

Perkakas akan menampilkan hanya baris yang memaparkan durasi rute yang ingin ditempuh. Jika tes berhasil, keluaran untuk Windows (tes sebelumnya) maupun Linux akan sama.

- Keluaran perkakas:

```

1 Step 1: Walk about 203 meter from your starting point to Jalan Cihampelas.
2 Step 2: Take angkot Ciumbuleuit - St. Hall (belok) at Jalan Cihampelas, and alight at 40141 about 3.9 kilometer
3 later.
3 Step 3: Walk about 64 meter from 40141 to your destination.

```

- Status tes: **Sukses**

4. Mengeluarkan hanya durasi dari rute (Windows)

- Perintah masukan:

```

1 kiritool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en | findstr /i
2 "duration"

```

- Keluaran yang diharapkan:

Perkakas akan menampilkan hanya baris yang memaparkan durasi rute yang ingin ditempuh. Jika tes berhasil, keluaran untuk Windows maupun Linux (tes selanjutnya) akan sama.

- 1 • Keluaran perkakas:

2 1 Estimated duration: 25 minutes

- 3 • Status tes: **Sukses**

4 5. Mengeluarkan hanya durasi dari rute (Linux)

- 5 • Perintah masukan:

6 1 kiritoool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en | grep -i
7 2 "duration"

- 8 • Keluaran yang diharapkan:

9 Perkakas akan menampilkan hanya baris yang memaparkan durasi rute yang ingin
10 ditempuh. Jika tes berhasil, keluaran untuk Windows (tes sebelumnya) maupun Linux
11 akan sama.

- 12 • Keluaran perkakas:

13 1 Estimated duration: 25 minutes

- 14 • Status tes: **Sukses**

15 6. Menampilkan jumlah langkah yang perlu ditempuh (Linux)

- 16 • Perintah masukan:

17 1 kiritoool --mode direct --regstart bdo --start ciwalk --regfinish bdo --finish unpar --locale en | grep -i "step"
18 2 | wc -l

- 19 • Keluaran yang diharapkan:

20 Ada tiga langkah dalam rute yang diuji, jadi perkakas akan mengeluarkan ‘3’.

- 21 • Keluaran perkakas:

22 1 3

- 23 • Status tes: **Sukses**

- 24 • Catatan tambahan:

25 Tes ini hanya berlaku untuk sistem operasi berbasis Linux, karena tidak ada perintah
26 dasar bawaan dalam *command line* Windows yang dapat menghitung jumlah baris dalam
27 suatu keluaran. Tes ini juga tidak dapat dilakukan ke pencarian rute yang menghasilkan
28 lebih dari satu hasil.

1

BAB 6

2

KESIMPULAN

3 Bab ini akan membahas kesimpulan yang telah ditarik selama seluruh proses pembuatan skripsi
4 ini, mulai dari studi literatur, studi kasus (analisis perkakas sejenis), perancangan dan pembuatan
5 perkakas, sampai dengan pengujian perkakas, serta saran-saran yang dapat diperhatikan untuk
6 perkembangan perkakas atau riset lanjutan yang dapat dilakukan.

7 6.1 Kesimpulan

8 Berikut merupakan kesimpulan yang diambil dari penulisan skripsi ini:

- 9 • Telah berhasil dibuat sebuah perkakas bernama “*KIRI Tool*”, yang merupakan perkakas
10 *command line* yang mengimplementasikan fitur-fitur API KIRI.
- 11 • Integrasi perkakas *KIRI Tool* dengan perkakas-perkakas *command line* lainnya dapat dilakukan
12 dengan beberapa metode, dengan metode yang paling umum adalah menggunakan `grep` untuk
13 mengekstraksi aspek tertentu dari keluaran perkakas.

14 6.2 Saran

15 Berikut adalah saran yang dapat dipertimbangkan untuk penelitian lanjutan dari skripsi ini:

- 16 • Mengembangkan perangkat agar bisa digunakan di sistem operasi lain selain Windows dan
17 Linux, misal macOS.
- 18 • Mencari tahu dan mengimplementasikan optimisasi apa saja yang dapat diberlakukan terhadap
19 perkakas, seperti fungsi-fungsi apa dari *library-library* yang ada yang dapat mengefektifkan
20 penggunaan memori atau kecepatan pemrosesan perkakas, atau bahkan menggunakan *library*
21 yang lebih tepat.
- 22 • Menambahkan fitur baru kepada perkakas yang mungkin dapat memberitahu persis lokasi-
23 lokasi yang dilewati dalam rute yang didaftarkan.

DAFTAR REFERENSI

- [1] Loosemore, S. dkk. (2022) *The GNU C Library Reference Manual*, 2.35 edition. Free Software Foundation, Inc., Massachusetts.
- [2] Stenberg, D. (2022) *Everything curl*. GitBook, Rhone-Alpes.
- [3] Martin, K. dan Hoffman, B. (2013) *Mastering CMake*, 6th edition. Kitware, Inc., New York.
- [4] Marsh, N. (2010) *Introduction to the Command Line: The Fat-Free Guide to Unix and Linux Commands*, 2nd edition. CreateSpace, South Carolina.
- [5] Shotts Jr., W. E. (2019) *The Linux Command Line*, 5th internet edition. <https://www.linuxcommand.org/tlcl.php>.
- [6] 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.
- [7] Matthew, N. dan Stones, R. (2007) *Beginning Linux® Programming*, 4th edition. Wiley Publishing, Inc., Indiana.
- [8] Microsoft Docs (2021) Windows commands. <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands>. versi 01 Mei 2022.

LAMPIRAN A

KODE PERKAKAS *COMMAND LINE KIRI*

Kode A.1: CMakeLists.txt

```

1 set(PROJECT_NAME kiritool)
2 set(SOURCES main.c includes/cJSON/cJSON.c includes/cJSON/cJSON.h)
3
4 cmake_minimum_required(VERSION 3.18.0)
5 project(${PROJECT_NAME} VERSION 1.2.13 LANGUAGES C)
6
7 add_executable(${PROJECT_NAME} ${SOURCES})
8
9 find_package(CURL REQUIRED)
10
11 # libcurl
12 target_link_libraries(${PROJECT_NAME} PRIVATE CURL::libcurl)
13 # cJSON
14 target_include_directories(${PROJECT_NAME} PRIVATE includes)
15
16 # UNIX specific commands
17 if (UNIX)
18   include(GNUInstallDirs)
19
20   file(ARCHIVE_CREATE OUTPUT ${CMAKE_CURRENT_SOURCE_DIR}/build/kiritool.1.gz
21     PATHS ${CMAKE_CURRENT_SOURCE_DIR}/additionals/linux/kiritool.1
22     FORMAT raw
23     COMPRESSION GZip
24   )
25   set(MANFILES build/kiritool.1.gz)
26
27   if (DEFINED CMAKE_INSTALL_MANDIR)
28     install(FILES ${MANFILES} DESTINATION ${CMAKE_INSTALL_MANDIR}/man1)
29   else()
30     message(FATAL_ERROR "CMAKE_INSTALL_MANDIR is not defined. \n ")
31   endif()
32 endif()

```

Kode A.2: main.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <getopt.h>
5
6 #include <cJSON/cJSON.h>
7 #include "curl/curl.h"
8
9 struct chunk {
10   char *data;
11   size_t size;
12 };
13 struct chunk respondedata; // Incoming data chunks
14
15 cJSON *responseJSON;
16 char URL[100] = "https://projectkiri.id/api?version=2";
17 int mode = -1; // -1 = undefined, 0 = unknown, 1 = help, 2 = searchplace, 3 = findroute, 4 = direct
18 int region = -1; // -1 = undefined, 0 = unknown, 1 = cgk, 2 = bdo, 3 = mlg, 4 = sub
19 char query[100];
20 char start[100];
21 char finish[100];
22 char escape[100]; // Temporary variable for escaping string inputs
23 int regstart = -1; // -1 = undefined, 0 = unknown, 1 = cgk, 2 = bdo, 3 = mlg, 4 = sub
24 int regfinish = -1; // -1 = undefined, 0 = unknown, 1 = cgk, 2 = bdo, 3 = mlg, 4 = sub
25 int locale = 0; // 0 = id, 1 = en, 2 = unknown
26 int step; // 0 = search starting location, 1 = search finish location, 2 = find route
27 // Used only in multistep modes
28 // 1 = an error has occurred, otherwise 0
29 int error = 0;
30
31 // Allocate the memory of incoming data
32 // If data size + original allocated size exceeds the memory capability, print an error.
33 size_t write_memalloc(void *incomingdata, size_t size, size_t nmemb, void *userdata) {
34   size_t realsize = size * nmemb;
35   struct chunk *memory = (struct chunk *)userdata;
36   char *ptr = realloc(memory->data, memory->size + realsize + 1);
37   if(ptr == NULL) {

```

```

38     fprintf(stderr, "Out_of_memory!\n");
39     return 0;
40 }
41 memory->data = ptr;
42 memcpy(&(memory->data[memory->size]), incomingdata, realsize);
43 memory->size += realsize;
44 memory->data[memory->size] = 0;
45
46 return realsize;
47 }
48
49 void write_searchplace() {
50     cJSON *status;
51     cJSON *result;
52     cJSON *resultitem;
53     cJSON *resultitemname;
54     cJSON *resultitemlocation;
55     int indexitem;
56
57     responseJSON = cJSON_Parse(responsedata.data);
58     status = cJSON_GetObjectItem(responseJSON, "status");
59
60     // Check whether API returned an error
61     if (strcmp(status->valuestring, "ok") != 0) {
62         error = 1;
63         fputs("\nError:\n", stderr);
64
65         if (locale == 1) {
66             fputs("API_returned_an_error_as_its_output.\n", stderr);
67             fputs("Please_recheck_whether_the_query_was_formatted_correctly.\n", stderr);
68         }
69         else {
70             fputs("API_mengembalikan_error_sebagai_keluarannya.\n", stderr);
71             fputs("Silakan_cek_ulang_apakah_kata_kunci_pencarian_sudah_diformat_dengan_benar.\n", stderr);
72         }
73     }
74     else {
75         result = cJSON_GetObjectItem(responseJSON, "searchresult");
76
77         // Check whether API managed to find a single result
78         if (cJSON_GetArraySize(result) == 0) {
79             error = 1;
80             fputs("\n", stderr);
81
82             if (locale == 1) {
83                 fputs("Location_not_found.\n", stderr);
84                 fputs("Please_recheck_whether_the_search_keyword_was_inputted_correctly.\n", stderr);
85             }
86             else {
87                 fputs("Lokasi_tidak_berhasil_ditemukan.\n", stderr);
88                 fputs("Silakan_cek_ulang_apakah_kata_kunci_pencarian_sudah_dimasukkan_dengan_benar.\n", stderr);
89             }
90         }
91         else {
92             indexitem = 1;
93
94             cJSON_ArrayForEach(resultitem, result) {
95                 resultitemname = cJSON_GetObjectItem(resultitem, "placename");
96                 resultitemlocation = cJSON_GetObjectItem(resultitem, "location");
97
98                 // Print location index only if there were more than one found
99                 if (cJSON_GetArraySize(result) > 1) {
100                     if (locale == 1) {
101                         printf("Location_%d:\n", indexitem);
102                     }
103                     else printf("Lokasi_%d:\n", indexitem);
104                 }
105                 else {
106                     if (locale == 1) {
107                         puts("Location:");
108                     }
109                     else puts("Lokasi:");
110                 }
111                 puts("-----");
112
113                 if (locale == 1) {
114                     printf("Name:_%s\n", resultitemname->valuestring);
115                     printf("Coordinates:_%s\n", resultitemlocation->valuestring);
116                 }
117                 else {
118                     printf("Nama:_%s\n", resultitemname->valuestring);
119                     printf("Koordinat:_%s\n", resultitemlocation->valuestring);
120                 }
121                 putchar('\n');
122
123                 indexitem++;
124             }
125         }
126     }
127 }
128
129 void write_searchplace_noreturns() {
130     cJSON *status;
131     cJSON *result;
132     cJSON *resultitem;
133     cJSON *resultitemname;
134     cJSON *resultitemlocation;
135
136     responseJSON = cJSON_Parse(responsedata.data);

```

```

137|     status = cJSON_GetObjectItem(responseJSON, "status");
138|
139|     // Check whether API returned an error
140|     if (strcmp(status->valuestring, "ok") != 0) {
141|         error = 1;
142|         fputs("\nError:\n", stderr);
143|
144|         if (step == 0) {
145|             if (locale == 1) {
146|                 fputs("API_returned_an_error_as_its_start_coordinates.\n", stderr);
147|                 fputs("Please_recheck_whether_the_starting_location_coordinates_were_inputted_correctly.\n", stderr);
148|             }
149|             else {
150|                 fputs("API_mengembalikan_error_sebagai_koordinat_lokasi_awal.\n", stderr);
151|                 fputs("Silakan_cek_ulang_apakah_koordinat_lokasi_awal_sudah_dimasukkan_dengan_benar.\n", stderr);
152|             }
153|         }
154|         else if (step == 1) {
155|             if (locale == 1) {
156|                 fputs("API_returned_an_error_as_its_end_coordinates.\n", stderr);
157|                 fputs("Please_recheck_whether_the_end_location_coordinates_were_inputted_correctly.\n", stderr);
158|             }
159|             else {
160|                 fputs("API_mengembalikan_error_sebagai_koordinat_lokasi_akhir.\n", stderr);
161|                 fputs("Silakan_cek_ulang_apakah_koordinat_lokasi_akhir_sudah_dimasukkan_dengan_benar.\n", stderr);
162|             }
163|         }
164|     }
165|     else {
166|         result = cJSON_GetObjectItem(responseJSON, "searchresult");
167|
168|         // Check whether API managed to find a single result
169|         if (cJSON_GetArraySize(result) == 0) {
170|             error = 1;
171|             fputs("\n", stderr);
172|
173|             if (step == 0) {
174|                 if (locale == 1) {
175|                     fputs("Starting_location_not_found.\n", stderr);
176|                     fputs("Please_recheck_whether_the_search_keyword_was_inputted_correctly.\n", stderr);
177|                 }
178|                 else {
179|                     fputs("Lokasi_awal_tidak_berhasil_ditemukan.\n", stderr);
180|                     fputs("Silakan_cek_ulang_apakah_kata_kunci_pencarian_sudah_dimasukkan_dengan_benar.\n", stderr);
181|                 }
182|             }
183|             else if (step == 1) {
184|                 if (locale == 1) {
185|                     fputs("Finish_location_not_found.\n", stderr);
186|                     fputs("Please_recheck_whether_the_search_keyword_was_inputted_correctly.\n", stderr);
187|                 }
188|                 else {
189|                     fputs("Lokasi_akhir_tidak_berhasil_ditemukan.\n", stderr);
190|                     fputs("Silakan_cek_ulang_apakah_kata_kunci_pencarian_sudah_dimasukkan_dengan_benar.\n", stderr);
191|                 }
192|             }
193|         }
194|     }
195|     else {
196|         // Direct route mode only supports first location found
197|         resultitem = cJSON_GetArrayItem(result, 0);
198|         resultitemname = cJSON_GetObjectItem(resultitem, "placename");
199|         resultitemlocation = cJSON_GetObjectItem(resultitem, "location");
200|
201|         if (step == 0) {
202|             if (locale == 1) {
203|                 printf("Starting_location:_%s\n", resultitemname->valuestring);
204|             }
205|             else printf("Lokasi_awal:_%s\n", resultitemname->valuestring);
206|             strcpy(start, resultitemlocation->valuestring);
207|
208|         else if (step == 1) {
209|             if (locale == 1) {
210|                 printf("Finish_location:_%s\n", resultitemname->valuestring);
211|             }
212|             else printf("Lokasi_akhir:_%s\n", resultitemname->valuestring);
213|             strcpy(finish, resultitemlocation->valuestring);
214|         }
215|     }
216| }
217|
218| void write_findroute() {
219|     cJSON *status;
220|     cJSON *result;
221|     cJSON *route;
222|     cJSON *routesteps;
223|     cJSON *routetime;
224|     cJSON *routestepitem;
225|     cJSON *routestepdetail;
226|     char routetimestring[20]; // Store translated duration string
227|     char *routetimetemp;
228|     char *timearraytoken;
229|     int indexroute;
230|     int indexstep;
231|
232|     responseJSON = cJSON_Parse(responsedata.data);
233|     status = cJSON_GetObjectItem(responseJSON, "status");
234|
235|     // Check whether API returned an error

```

```

236| if (strcmp(status->valuestring, "ok") != 0) {
237|   error = 1;
238|   fputs("\nError:\n", stderr);
239|
240|   if (locale == 1) {
241|     fputs("API returned an error as its output.\n", stderr);
242|     fputs("Please recheck whether both location's coordinates were inputted correctly.\n", stderr);
243|   }
244|   else {
245|     fputs("API mengembalikan error sebagai keluarannya.\n", stderr);
246|     fputs("Silakan cek ulang apakah koordinat kedua lokasi sudah dimasukkan dengan benar.\n", stderr);
247|   }
248| }
249| else {
250|   result = cJSON_GetObjectItem(responseJSON, "routingresults");
251|   indexroute = 1;
252|
253|   // For each possible routes...
254|   cJSON_ArrayForEach(route, result) {
255|     routesteps = cJSON_GetObjectItem(route, "steps");
256|     routetime = cJSON_GetObjectItem(route, "traveltime");
257|
258|     if (mode == 4 && indexroute == 1) {
259|       putchar('\n');
260|     }
261|
262|     // Check whether API managed to find a single result
263|     if (cJSON_IsNull(routetime)) {
264|       error = 1;
265|       fputs("\n", stderr);
266|
267|       if (locale == 1) {
268|         fputs("Sorry, we couldn't find a public transport route for your trip.\n", stderr);
269|       }
270|       else fputs("Maaf, kami tidak dapat menemukan rute transportasi publik untuk perjalanan anda.\n", stderr);
271|     }
272|     else {
273|       routetimetemp = routetime->valuestring;
274|       memset(routimestring, 0, sizeof(routimestring));
275|       indexstep = 1;
276|
277|       // Fix bug from KIRI API in which "minute" time unit is not translated in id
278|       // Else skip whole process
279|       if (locale != 1) {
280|         timearraytoken = strtok(routetimetemp, " ");
281|
282|         while (timearraytoken != NULL) {
283|           if (strcmp(timearraytoken, "minutes") == 0) {
284|             strcpy(timearraytoken, "menit");
285|           }
286|           strcat(routimestring, timearraytoken);
287|           strcat(routimestring, " ");
288|
289|           timearraytoken = strtok(NULL, " ");
290|         }
291|       }
292|       else strcpy(routimestring, routetimetemp);
293|
294|       // Print route index only if there were more than one found
295|       // Otherwise simply print "Route:"
296|       if (cJSON_GetArraySize(result) > 1) {
297|         if (locale == 1) {
298|           printf("Route %d:\n", indexroute);
299|         }
300|         else printf("Rute %d:\n", indexroute);
301|       }
302|       else {
303|         if (locale == 1) {
304|           puts("Route:");
305|         }
306|         else puts("Rute:");
307|       }
308|       puts("-----");
309|
310|       if (locale == 1) {
311|         printf("Estimated duration: %s\n", routimestring);
312|       }
313|       else printf("Estimasi waktu: %s\n", routimestring);
314|       puts("-----");
315|
316|       cJSON_ArrayForEach(routestepitem, routesteps) {
317|         routestepdetail = cJSON_GetArrayItem(routestepitem, 3);
318|
319|         if (locale == 1) {
320|           printf("Step %d: ", indexstep);
321|         }
322|         else printf("Langkah %d: ", indexstep);
323|         printf("%s\n", routestepdetail->valuestring);
324|         indexstep++;
325|       }
326|
327|       if (error != 1) {
328|         putchar('\n');
329|     };
330|
331|     indexroute++;
332|   }
333}
334}

```

```

335 void print_help() {
336     puts("KIRI_Command_Line_Tool,_version_1.2.13");
337     puts("Use_the_KIRI_tool_through_the_command_line.");
338     putchar('\n');
339     puts("USAGE:");
340     puts("kiritool_<COMMAND>_[OPTIONS...][<ARGUMENTS>]");
341     putchar('\n');
342     puts("COMMAND:");
343     puts(" -h,--help           Display_usage_tutorial.");
344     puts(" -m,--mode <MODE> Set_tool_operation_mode.\n");
345     puts(" You can only choose one mode per operation.");
346     putchar('\n');
347     puts("OPTIONS:");
348     puts(" -F,--regfinish <REGION> Set_region_to_search_for_finish_location_in.");
349     puts(" -f,--finish      Set_finish_location.");
350     puts(" -l,--locale <LANG> Set_tool_output_language.");
351     puts(" -q,--query <KEYWORD> Set_keyword_for_searching.");
352     puts(" -r,--region <REGION> Set_region_to_search_for_location_in.");
353     puts(" -S,--restart <REGION> Set_region_to_search_for_starting_location_in.");
354     puts(" -s,--start <START> Set_starting_location.");
355     putchar('\n');
356     puts("ARGUMENTS:");
357     puts(" <FINISH>");
358     puts(" Route_finish_location.");
359     puts(" For 'findroute' mode, input the location's latitude_and_longitude_coordinates.");
360     puts(" For 'direct' mode, input the location's search_keyword(<KEYWORD> argument).");
361     puts(" <KEYWORD>");
362     puts(" Keywords_used_by_the_tool_to_search_for_the_desired_location.");
363     puts(" For_multiple_words_queries, encase the keywords in quotation_marks_(\" \").");
364     puts(" <LANG>");
365     puts(" Tool_output_language.");
366     puts(" Available_languages: id_en");
367     puts(" -id_(Indonesian,_Default)");
368     puts(" -en_(English)");
369     puts(" <MODE>");
370     puts(" Tool_operation_mode .Note that not_all_options_are_needed_for_all_modes.");
371     puts(" Should_the_user_input_extra_options,_they_will_simply_not_be_used.");
372     puts(" Available_modes:");
373     puts(" -searchplace");
374     puts(" Required_options: --query,--region");
375     puts(" Optional_options: --locale");
376     puts(" -findroute");
377     puts(" Required_options: --start,--finish");
378     puts(" Optional_options: --locale");
379     puts(" -direct");
380     puts(" Required_options: --restart,--start,--regfinish,--finish");
381     puts(" Optional_options: --locale");
382     puts(" <REGION>");
383     puts(" Region_to_search_for_locations_in.");
384     puts(" Available_regions:");
385     puts(" -cgk_(Jakarta)");
386     puts(" -bdo_(Bandung)");
387     puts(" -mlg_(Malang)");
388     puts(" -sub_(Surabaya)");
389     puts(" <START>");
390     puts(" Route_starting_location.");
391     puts(" For 'findroute' mode, input the location's latitude_and_longitude_coordinates.");
392     puts(" For 'direct' mode, input the location's search_keyword(<KEYWORD> argument).");
393 }
394
395 void print_curl_error() {
396     error = 1;
397     fputs("\nError:\n", stderr);
398
399     if (locale == 1) {
400         fputs("A connection_error_has_occurred.\n", stderr);
401         fputs("Please_verify_whether_the_internet_connection_is_up_and_running.\n", stderr);
402     } else {
403         fputs("Telah_terjadi_error_koneksi.\n", stderr);
404         fputs("Mohon_cek_apakah_koneksi_internet_aktif_dan_terhubung_dengan_baik.\n", stderr);
405     }
406 }
407
408
409 void execute_curl() {
410     CURL *curl;
411     CURLcode response;
412     memset(&responsedata, 0, sizeof(responsedata)); // Empty response data chunk for multiprocess modes
413
414     curl_global_init(CURL_GLOBAL_DEFAULT);
415     curl = curl_easy_init();
416
417     if (curl) {
418         curl_easy_setopt(curl, CURLOPT_URL, URL);
419         curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_memalloc);
420         curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&responsedata);
421         response = curl_easy_perform(curl);
422         if (response != CURLE_OK) {
423             print_curl_error();
424         }
425     }
426
427     switch (mode) {
428         case 2: // searchplace
429             write_searchplace();
430             break;
431
432         case 3: // findroute
433             write_findroute();

```

```

434         break;
435
436     case 4: // directroute
437         if (step == 0 || step == 1) { // search start and finish
438             write_searchplace_noreturns();
439         }
440         else if (step == 2) { // findroute
441             write_findroute();
442         }
443         break;
444
445     default:
446         break;
447     }
448
449     curl_easy_cleanup(curl);
450 }
451
452 curl_global_cleanup();
453 }
454
455 void build_url_searchplace(int region, char* query) {
456     strcat(URL, "&mode=searchplace");
457
458 // Locale check - tool addition
459     if (locale == 2) {
460         error = 1;
461         fputs("\nError:\n", stderr);
462         fputs("Anda telah memasukkan pilihan bahasa_(locale)_yang_tidak_valid.\n", stderr);
463         fputs("Mohon periksa kembali apakah pilihan bahasa_yang_anda_masukkan_valid_atau_tidak.\n", stderr);
464         fputs("Pilihan_locale:_id,_en\n", stderr);
465         fputs("-----\n", stderr);
466         fputs("You_have_inputted_an_invalid_language_(locale)_option.\n", stderr);
467         fputs("Please_recheck_whether_the_language_code_you_inserted_was_supported_or_not.\n", stderr);
468         fputs("Locale_available:_id,_en\n", stderr);
469         exit(1);
470     }
471
472 // Region check
473     switch (region) {
474         case -1:
475             error = 1;
476             fputs("\nError:\n", stderr);
477
478             if (locale == 1) {
479                 fputs("Location_searching_requires_a_region_to_be_set.\n", stderr);
480                 fputs("Please_make_sure_you_have_chosen_between_one_of_the_four_available_regions.\n", stderr);
481                 fputs("Regions_available:_cgk,_bdo,_mlg,_sub\n", stderr);
482             }
483             else {
484                 fputs("Fitur_pencarian_lokasi_memerlukan_pengaturan_region_lokasi_yang_ingin_dicari.\n", stderr);
485                 fputs("Mohon_pastikan_anda_sudah_memasukkan salah_satu_dari_empat_kode_region_yang_tersedia.\n", stderr);
486                 fputs("Pilihan_region:_cgk,_bdo,_mlg,_sub\n", stderr);
487             }
488             exit(1);
489             break;
490
491         case 1:
492             strcat(URL, "&region=cgk");
493             break;
494
495         case 2:
496             strcat(URL, "&region=bdo");
497             break;
498
499         case 3:
500             strcat(URL, "&region=mlg");
501             break;
502
503         case 4:
504             strcat(URL, "&region=sub");
505             break;
506
507     default:
508         error = 1;
509         fputs("\nError:\n", stderr);
510
511         if (locale == 1) {
512             fputs("You_have_inputted_an_invalid_region.\n", stderr);
513             fputs("Please_recheck_whether_the_region_code_chosen_was_one_of_the_four_region_codes_supported.\n", stderr);
514             fputs("Regions_available:_cgk,_bdo,_mlg,_sub\n", stderr);
515         }
516         else {
517             fputs("Anda_telah_memasukkan_region_yang_tidak_valid.\n", stderr);
518             fputs("Mohon_periksa_kembali_apakah_kode_region_yang_anda_masukkan_merupakan salah_satu_dari_empat_kode_region_yang_tersedia.\n", stderr);
519             fputs("Pilihan_region:_cgk,_bdo,_mlg,_sub\n", stderr);
520         }
521         exit(1);
522         break;
523     }
524
525 // Query check
526     if (strcmp(query, "\0") == 0) {
527         error = 1;
528         fputs("\nError:\n", stderr);
529
530         if (locale == 1) {
531             fputs("Location_searching_requires_a_search_query.\n", stderr);

```

```

532         fputs("Please_make_sure_you_have_inputted_a_query_to_be_used_in_the_search.\n", stderr);
533     } else {
534         fputs("Fitur_pencarian_lokasi_memerlukan_sebuah_kata_kunci_pencarian.\n", stderr);
535         fputs("Mohon_pastikan_anda_sudah_memasukkan_kata_kunci_untuk_melakukan_pencarian_lokasi.\n", stderr);
536     }
537     exit(1);
538 }
539 else {
540     strcat(URL, "&querystring=");
541     strcat(URL, query);
542 }
543 strcat(URL, "&apikey=68CD281C8A8EE97C");
544 }
545 }
546 void build_url_findroute(int locale, char* start, char* finish) {
547     strcat(URL, "&mode=findroute");
548
549     // Locale check
550     switch (locale) {
551         case 1:
552             strcat(URL, "&locale=en");
553             break;
554
555         case 2:
556             error = 1;
557             fputs("\nError:\n", stderr);
558             fputs("Anda_telah_memasukkan_pilihan_bahasa_(locale)_yang_tidak_valid.\n", stderr);
559             fputs("Mohon_periksa_kembali_apakah_pilihan_bahasa_yang_anda_masukkan_valid_atau_tidak.\n", stderr);
560             fputs("Pilihan_locale:_id,_en\n", stderr);
561             fputs("-----\n", stderr);
562             fputs("You_have_inputted_an_invalid_language_(locale)_option.\n", stderr);
563             fputs("Please_recheck_whether_the_language_code_you_inserted_was_supported_or_not.\n", stderr);
564             fputs("Locale_available:_id,_en\n", stderr);
565             exit(1);
566             break;
567
568         default:
569             strcat(URL, "&locale=id");
570             break;
571     }
572
573     // Starting location check
574     if (strcmp(start, "\0") == 0) {
575         error = 1;
576         fputs("\nError:\n", stderr);
577
578         if (locale == 1) {
579             fputs("You_did_not_input_the_coordinates_of_the_starting_location.\n", stderr);
580             fputs("Please_input_the_coordinates_of_the_starting_location_through_the_corresponding_option.\n", stderr);
581         }
582         else {
583             fputs("Anda_belum_memasukkan_sebuah_koordinat_untuk_lokasi_awal.\n", stderr);
584             fputs("Mohon_masukkan_koordinat_untuk_lokasi_awal_pencarian_rute_melalui_opsi_yang_sesuai.\n", stderr);
585         }
586         exit(1);
587     }
588     else {
589         strcat(URL, "&start=");
590         strcat(URL, start);
591     }
592
593     // End location check
594     if (strcmp(finish, "\0") == 0) {
595         error = 1;
596         fputs("\nError:\n", stderr);
597
598         if (locale == 1) {
599             fputs("You_did_not_input_the_coordinates_of_the_end_location.\n", stderr);
600             fputs("Please_input_the_coordinates_of_the_end_location_through_the_corresponding_option.\n", stderr);
601         }
602         else {
603             fputs("Anda_belum_memasukkan_sebuah_koordinat_untuk_lokasi_akhir.\n", stderr);
604             fputs("Mohon_masukkan_koordinat_untuk_lokasi_akhir_pencarian_rute_melalui_opsi_yang_sesuai.\n", stderr);
605         }
606         exit(1);
607     }
608     else {
609         strcat(URL, "&finish=");
610         strcat(URL, finish);
611     }
612
613     strcat(URL, "&presentation=desktop");
614     strcat(URL, "&apikey=68CD281C8A8EE97C");
615 }
616 }
617 void reset_url() {
618     // Reset URL for multiple API processes at once
619     strcpy(URL, "https://projectkiri.id/api?version=2");
620 }
621
622 void replace_space(char* string) {
623     // Escapes all whitespace characters in string
624     int i;
625     int j = 0;
626
627     for (i = 0; i < 100; i++) {
628         if (string[i] == '\0') {
629             break;
630         }
631     }
632 }
```



```

730         memset(escape, 0, sizeof(escape));
731     }
732     break;
733
734 case 'f':
735     // Finish location input
736     if (optarg[0] != '\0') {
737         replace_space(optarg);
738         strcpy(finish, escape);
739         memset(escape, 0, sizeof(escape));
740     }
741     break;
742
743 case 'S':
744     // Starting location regions
745     if (strcmp(optarg, "cgk") == 0) {
746         restart = 1;
747     }
748     else if (strcmp(optarg, "bdo") == 0) {
749         restart = 2;
750     }
751     else if (strcmp(optarg, "mlg") == 0) {
752         restart = 3;
753     }
754     else if (strcmp(optarg, "sub") == 0) {
755         restart = 4;
756     }
757     else {
758         restart = 0;
759     }
760     break;
761
762 case 'F':
763     // Finish location regions
764     if (strcmp(optarg, "cgk") == 0) {
765         regfinish = 1;
766     }
767     else if (strcmp(optarg, "bdo") == 0) {
768         regfinish = 2;
769     }
770     else if (strcmp(optarg, "mlg") == 0) {
771         regfinish = 3;
772     }
773     else if (strcmp(optarg, "sub") == 0) {
774         regfinish = 4;
775     }
776     else {
777         regfinish = 0;
778     }
779     break;
780
781 case 'l':
782     // Locale
783     if (strcmp(optarg, "id") == 0) {
784         locale = 0;
785     }
786     else if (strcmp(optarg, "en") == 0) {
787         locale = 1;
788     }
789     else {
790         locale = 2;
791     }
792     break;
793
794 case ':':
795     // Error: missing arguments
796     error = 1;
797     fputs("\nError:\n", stderr);
798
799     if (locale == 1) {
800         fputs("One_of_the_options_inputted_was_missing_its_required_argument.\n", stderr);
801         fputs("Please_recheck_the_input_command's_syntax.\n", stderr);
802     }
803     else {
804         fputs("Salah_satu_dari_opsi_yang_anda_masukkan_kehilangan_argumen.yang_dibutuhkan.\n", stderr);
805         fputs("Mohon_periksa_kembali_penulisan_perintah_yang_anda_masukkan.\n", stderr);
806     }
807     exit(1);
808
809 case '?':
810     // Error: unknown options
811     error = 1;
812     fputs("\nError:\n", stderr);
813
814     if (locale == 1) {
815         fputs("You_have_inputted_an_invalid_option.\n", stderr);
816         fputs("Please_recheck_the_input_command's_syntax.\n", stderr);
817     }
818     else {
819         fputs("Anda_telah_memasukkan_opsi_yang_tidak_valid.\n", stderr);
820         fputs("Mohon_periksa_kembali_penulisan_perintah_yang_anda_masukkan.\n", stderr);
821     }
822     exit(1);
823
824 default:
825     exit(1);
826 }
827
828 }
```

```

829 // Error: extra arguments
830 if (optind < argc) {
831     error = 1;
832     fputs("\nError:\n", stderr);
833
834     if (locale == 1) {
835         fprintf(stderr, "You_have_inserted_some_extra_arguments:");
836         while (optind < argc) {
837             fprintf(stderr, "%s ", argv[optind++]);
838         }
839         fputs("\nPlease_recheck_the_input_command's_syntax.\n", stderr);
840     }
841     else {
842         fprintf(stderr, "Anda_telah_memasukkan_kelebihan_argumen:");
843         while (optind < argc) {
844             fprintf(stderr, "%s ", argv[optind++]);
845         }
846         fputs("\nMohon_periksa_kembali_penulisan_perintah_yang_anda_masukkan.\n", stderr);
847     }
848 }
849 else {
850     switch (mode) {
851         case -1:
852             error = 1;
853             fputs("\nError:\n", stderr);
854
855             if (locale == 1) {
856                 fputs("Please_enter_tool_operation_mode.\n", stderr);
857             }
858             else {
859                 fputs("Mohon_masukkan_mode_penggunaan_perkakas.\n", stderr);
860             }
861             exit(1);
862             break;
863
864         case 1:
865             print_help();
866             break;
867
868         case 2:
869             build_url_searchplace(region, query);
870             execute_curl();
871             break;
872
873         case 3:
874             build_url_findroute(locale, start, finish);
875             execute_curl();
876             break;
877
878         case 4:
879             step = 0;
880             build_url_searchplace(regstart, start);
881             execute_curl();
882             if (error == 1) {
883                 break;
884             }
885             else reset_url();
886
887             step = 1;
888             build_url_searchplace(regfinish, finish);
889             execute_curl();
890             if (error == 1) {
891                 break;
892             }
893             else reset_url();
894
895             step = 2;
896             build_url_findroute(locale, start, finish);
897             execute_curl();
898             break;
899
900     default:
901         error = 1;
902         fputs("\nError:\n", stderr);
903
904         if (locale == 1) {
905             fputs("You_have_entered_an_invalid_operation_mode.\n", stderr);
906             fputs("Please_recheck_whether_the_operation_mode_had_been_typed_correctly.\n", stderr);
907         }
908         else {
909             fputs("Anda_telah_memasukkan_mode_yang_tidak_valid.\n", stderr);
910             fputs("Mohon_periksa_kembali_apakah_mode_yang_anda_masukkan_sudah_diketik_dengan_benar.\n", stderr);
911         }
912         exit(1);
913         break;
914     }
915 }
916
917 exit(0);
918 }
```

Kode A.3: kiritool.1 (*Source Code man page*)

1	.TH Kiritool 1 "November 2022" "1.2.13" "kiritool Manual"
2	.SH NAME
3	kiritool \- search routes using angkot
4	.SH SYNOPSIS

```

5 \fBkiritool\fR \fIOPTIONS\fR... \fIARGUMENTS\fR
6 .SH DESCRIPTION
7 \fBkiritool\fR searches for available routes and shows steps needed to go from one location to another using angkot (angkutan kota
   ). This tool will be utilizing the KIRI API in its processes.
8 .TP
9 This tool has three features:
10 - Search location names from query
11 .br
12 - Find routes using angkot from starting and end locations' latitude and longitude coordinates
13 .br
14 - Find routes using angkot from starting and end locations' search query keywords
15 .br
16 .SH OPTIONS
17 .SS "Basic Program Options"
18 The user must select either one of these options per operation.
19 .br
20 .TP
21 .B -h, --help
22 Display concise help for tool usage.
23 .TP
24 .B -m, --mode
25 Set tool operation mode.
26 .br
27 .SS "Tool Operation Modes"
28 The user must select only one of these modes per operation.
29 .TP
30 .B searchplace
31 Search location names from user-provided query (\fB-q\fR).
32 Additionally, users must also provide the region (\fB-r\fR) to search the desired location in.
33 If at least one location was found, this mode will output their names, along with the latitude and longitude coordinates.
34 .TP
35 .B findroute
36 Find routes utilizing angkot and display the steps required for the route. User must provide the latitude and longitude
   coordinates of both the starting (\fB-s\fR) and end (\fB-f\fR) locations.
37 .br
38 If a route was found, this mode will output the estimated time required to take the route, along with the steps to be taken in
   said route.
39 .TP
40 .B direct
41 .br
42 Find routes utilizing angkot and display the steps required for the route. This mode combines both \fBsearchplace\fR and \
   \fBfindroute\fR modes, so user will only need to provide the queries for both the starting (\fB-s\fR) and end (\fB-f\fR)
   locations, along with the region (\fB-S\fR and \fB-F\fR) to search for each in.
43 .br
44 If a route was found, this mode will output the name of both START and FINISH, along with the estimated time required to take the
   route, and the steps to be taken in said route.
45 .IP
46 Note that this mode does not support vague queries - should either the search for starting or end locations return more than one
   results, this mode will only take the first ones.
47 .SS "Required Options"
48 Note that not all of these options are required for each of the operation modes. Should the user provide additional options than
   what the mode requires, they will simply be unused.
49 .br
50 For information on which options are required for certain modes, refer to the "\fBTool Operation Modes\fR" section.
51 .TP
52 \fB-F \fIREGION\fR, \fB--regfinish \fIREGION\fR
53 Set the \fIREGION\fR to search for the finish location (\fB-f\fR) in. Only four options are available as regions: \fBcgk\fR (
   Jakarta), \fBbdo\fR (Bandung), \fBmlg\fR (Malang), or \fBsub\fR (Surabaya).
54 .br
55 This option is specifically used for the \fBdirect\fR mode. For the \fBsearchplace\fR mode alternative, see option \fB-r\fR.
56 .TP
57 \fB-f \fIFINISH\fR, \fB--finish \fIFINISH\fR
58 Set where the desired \fIFINISH\fR should be.
59 .br
60 When this option is used in the \fBfindroute\fR mode, it will only accept latitude and longitude coordinates, formatted as \
   \fLATITUDE\fR,\fLONGITUDE\fR.
61 .br
62 Alternatively, when this option is used in the \fBdirect\fR mode, it will only accept keywords as its argument.
63 .TP
64 \fB-l \fILANG\fR, \fB--locale \fILANG\fR
65 Set the output language to \fILANG\fR. This tool only supports two languages: \fBid\fR (Indonesian) and \fBen\fR (English).
66 .br
67 If the user does not use this option, the tool will default to outputting in \fBid\fR.
68 .TP
69 \fB-q \fIKEYWORD\fR, \fB--query \fIKEYWORD\fR
70 Search for the desired location using \fIKEYWORD\fR. If \fIKEYWORD\fR contained more than one words, it must be encased in
   quotation marks ("").
71 .br
72 This option is only used in the \fBsearchplace\fR mode. For the \fBdirect\fR mode alternatives, see option \fB-s\fR and \fB-f\fR.
73 .TP
74 \fB-r \fIREGION\fR, \fB--region \fIREGION\fR
75 Set the \fIREGION\fR to search for the desired location (\fB-s\fR) in. Only four options are available as regions: \fBcgk\fR (
   Jakarta), \fBbdo\fR (Bandung), \fBmlg\fR (Malang), or \fBsub\fR (Surabaya).
76 .br
77 This option is specifically used for the \fBsearchplace\fR mode. For the \fBdirect\fR mode alternative, see options \fB-s\fR and \
   \fB-f\fR.
78 .TP
79 \fB-S \fIREGION\fR, \fB--restart \fIREGION\fR
80 Set the \fIREGION\fR to search for the start location (\fB-s\fR) in. Only four options are available as regions: \fBcgk\fR (
   Jakarta), \fBbdo\fR (Bandung), \fBmlg\fR (Malang), or \fBsub\fR (Surabaya).
81 .br
82 This option is specifically used for the \fBdirect\fR mode. For the \fBsearchplace\fR mode alternative, see option \fB-r\fR.
83 .TP
84 \fB-s \fISTART\fR, \fB--start \fISTART\fR
85 Set where the desired \fISTART\fR should be.
86 .br
87 When this option is used in the \fBfindroute\fR mode, it will only accept latitude and longitude coordinates, formatted as \
   \fLATITUDE\fR,\fLONGITUDE\fR.
88 .br

```

```

89| Alternatively, when this option is used in the \fBdirect\fR mode, it will only accept keywords as its argument.
90|.SH OUTPUT
91|.TP
92|.B searchplace
93 In this mode, the output will be the location name, along with its latitude and longitude coordinates for the user to use as the
   input for \fBfindroute\fR mode.
94|.TP
95|.B findroute
96 In this mode, the output will be the steps needed to be taken within the determined route.
97|.br
98 Should there be more than one possible routes, the tool will list all of them.
99|.TP
100|.B direct
101|.br
102 In this mode, the output will be the starting and end location names, and the steps needed to be taken within the determined route
   .
103|.br
104 Should there be more than one possible routes, the tool will list all of them.
105|.SH ERROR HANDLING
106 Should there be an error occurring somewhere (most likely due to faulty inputs), the tool will output an error message.
107|.br
108 This message will be in either Indonesian or English language, depending on the \fB-l\fR option's input.
109|.SH BUGS
110 No known bugs.
111|.SH AUTHOR
112 Alfred Aprianto Liaunardi (aaliaunardi@gmail.com)

```

Kode A.4: Bantuan penggunaan perkakas

```

1 KIRI Command Line Tool, version 1.2.13
2 Use the KIRI tool through the command line.
3
4 USAGE:
5   kiritoorl <COMMAND> [OPTIONS...] [<ARGUMENTS>]
6
7 COMMAND:
8   -h, --help           Display usage tutorial.
9   -m, --mode <MODE>    Set the tool operation mode.
10
11 You can only choose one mode per operation.
12
13 OPTIONS:
14   -F, --regfinish <REGION>      Set region to search for finish location in.
15   -f, --finish <FINISH>          Set finish location.
16   -l, --locale <LANG>            Set tool output language.
17   -q, --query <KEYWORD>         Set keyword for searching.
18   -r, --region <REGION>         Set region to search for location in.
19   -S, --restart <REGION>        Set region to search for starting location in.
20   -s, --start <START>           Set starting location.
21
22 ARGUMENTS:
23   <FINISH>
24     Route finish location.
25     For 'findroute' mode, input the location's latitude and longitude coordinates.
26     For 'direct' mode, input the location's search keyword (<KEYWORD> argument).
27   <KEYWORD>
28     Keywords used by the tool to search for the desired location.
29     For multiple words queries, encase the keywords in quotation marks (" ").
30   <LANG>
31     Tool output language.
32     Available languages: id, en
33       - id (Indonesian - Default)
34       - en (English)
35   <MODE>
36     Tool operation mode. Note that not all options are needed for all modes.
37     Should the user input extra options, they will simply not be used.
38     Available modes:
39       - searchplace
40         Required options: --query, --region
41         Optional options: --locale
42       - findroute
43         Required options: --start, --finish
44         Optional options: --locale
45       - direct
46         Required options: --restart, --start, --regfinish, --finish
47         Optional options: --locale
48   <REGION>
49     Region to search for locations in.
50     Available regions:
51       - cgk (Jakarta)
52       - bdo (Bandung)
53       - mlg (Malang)
54       - sub (Surabaya)
55   <START>
56     Route starting location.
57     For 'findroute' mode, input the location's latitude and longitude coordinates.
58     For 'direct' mode, input the location's search keyword (<KEYWORD> argument).

```

Kode A.5: man page Perkakas *Command Line KIRI*

<pre> 1 kiritoorl(1) 2 NAME 3 kiritoorl - search routes using angkot </pre>	<p style="text-align: right;">kiritoorl Manual kiritoorl(1)</p>
---	---

```

4
5 SYNOPSIS
6     kiritool OPTIONS... ARGUMENTS
7
8 DESCRIPTION
9     kiritool searches for available routes and shows steps needed to go from one location to another using angkot (angkutan
10    kota). This tool will be utilizing the KIRI API in its processes.
11    This tool has three features:
12        - Search location names from query
13        - Find routes using angkot from starting and end locations' latitude and longitude coordinates
14        - Find routes using angkot from starting and end locations' search query keywords
15
16 OPTIONS
17     Basic Program Options
18         The user must select either one of these options per operation.
19             -h, --help
20                 Display concise help for tool usage.
21             -m, --mode
22                 Set tool operation mode.
23     Tool Operation Modes
24         The user must select only one of these modes per operation.
25             searchplace
26                 Search location names from user-provided query (-q). Additionally, users must also provide the region (-r) to
27                 search the desired location in. If at least one location was found, this mode will output their names,
28                 along with the latitude and longitude coordinates.
29             findroute
30                 Find routes utilizing angkot and display the steps required for the route. User must provide the latitude and
31                 longitude coordinates of both the starting (-s) and end (-f) locations.
32                 If a route was found, this mode will output the estimated time required to take the route, along with the steps to
33                 be taken in said route.
34             direct
35                 Find routes utilizing angkot and display the steps required for the route. This mode combines both searchplace and
36                 findroute modes, so user will only need to provide the queries for both the starting (-s) and end (-f)
37                 locations, along with the region (-S and -F) to search for each in.
38                 If a route was found, this mode will output the name of both START and FINISH, along with the estimated time
39                 required to take the route, and the steps to be taken in said route.
40                 Note that this mode does not support vague queries - should either the search for starting or end locations return
41                 more than one results, this mode will only take the first ones.
42     Required Options
43         Note that not all of these options are required for each of the operation modes. Should the user provide additional options
44         than what the mode requires, they will simply be unused.
45         For information on which options are required for certain modes, refer to the "Tool Operation Modes" section.
46             -F REGION, --regfinish REGION
47                 Set the REGION to search for the finish location (-f) in. Only four options are available as regions: cgk (Jakarta),
48                 bdo (Bandung), mlg (Malang), or sub (Surabaya).
49                 This option is specifically used for the direct mode. For the searchplace mode alternative, see option -r.
50             -f FINISH, --finish FINISH
51                 Set where the desired FINISH should be.
52                 When this option is used in the findroute mode, it will only accept latitude and longitude coordinates, formatted as
53                     LATITUDE,LONGITUDE.
54                 Alternatively, when this option is used in the direct mode, it will only accept keywords as its argument.
55             -l LANG, --locale LANG
56                 Set the output language to LANG. This tool only supports two languages: id (Indonesian) and en (English).
57                 If the user does not use this option, the tool will default to outputting in id.
58             -q KEYWORD, --query KEYWORD
59                 Search for the desired location using KEYWORD. If KEYWORD contained more than one words, it must be encased in
60                     quotation marks ("").
61                 This option is only used in the searchplace mode. For the direct mode alternatives, see option -s and -f.
62             -r REGION, --region REGION
63                 Set the REGION to search for the desired location (-s) in. Only four options are available as regions: cgk (Jakarta),
64                 bdo (Bandung), mlg (Malang), or sub (Surabaya).
65                 This option is specifically used for the searchplace mode. For the direct mode alternative, see options -s and -f.
66             -S REGION, --regstart REGION
67                 Set the REGION to search for the start location (-s) in. Only four options are available as regions: cgk (Jakarta),
68                 bdo (Bandung), mlg (Malang), or sub (Surabaya).
69                 This option is specifically used for the direct mode. For the searchplace mode alternative, see option -r.
70             -s START, --start START
71                 Set where the desired START should be.
72                 When this option is used in the findroute mode, it will only accept latitude and longitude coordinates, formatted as
73                     LATITUDE,LONGITUDE.
74                 Alternatively, when this option is used in the direct mode, it will only accept keywords as its argument.
75
76 OUTPUT
77     searchplace
78         In this mode, the output will be the location name, along with its latitude and longitude coordinates for the user
79             to use as the input for findroute mode.
80             findroute
81                 In this mode, the output will be the steps needed to be taken within the determined route.
82                 Should there be more than one possible routes, the tool will list all of them.
83             direct
84                 In this mode, the output will be the starting and end location names, and the steps needed to be taken within the
85                     determined route.
86                 Should there be more than one possible routes, the tool will list all of them.
87
88 ERROR HANDLING
89     Should there be an error occurring somewhere (most likely due to faulty inputs), the tool will output an error message.
90     This message will be in either Indonesian or English language, depending on the -l option's input.
91
92 BUGS
93     No known bugs.
94
95 AUTHOR
96     Alfred Aprianto Liaunardi (aaliaunardi@gmail.com)
97
98 1.2.13

```