

## SKRIPSI

PERKAKAS ANTARMUKA BARIS PERINTAH UNTUK  
APLIKASI BERBASIS WEB KIRI



Alfred Aprianto Liaunardi

NPM: 6181801014

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2023



**UNDERGRADUATE THESIS**

**COMMAND LINE TOOL FOR KIRI WEB-BASED  
APPLICATION**



**Alfred Aprianto Liaunardi**

**NPM: 6181801014**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2023**



## **LEMBAR PENGESAHAN**

### **PERKAKAS ANTARMUKA BARIS PERINTAH UNTUK APLIKASI BERBASIS WEB KIRI**

**Alfred Aprianto Liaunardi**

**NPM: 6181801014**

**Bandung, 19 Januari 2023**

**Menyetuju,**

**Pembimbing**

**Pascal Alfadian, Nugroho, M.Comp.**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**Vania Natali, M.T.**

**Lionov, Ph.D.**

**Mengetahui,**

**Ketua Program Studi**

**Mariskha Tri Adithia, P.D.Eng**



## **PERNYATAAN**

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

### **PERKAKAS ANTARMUKA BARIS PERINTAH UNTUK APLIKASI BERBASIS WEB KIRI**

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

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

Dinyatakan di Bandung,  
Tanggal 19 Januari 2023

Alfred Aprianto Liaunardi  
NPM: 6181801014



## **ABSTRAK**

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



## ABSTRACT

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



*Skripsi ini dipersembahkan kepada Tuhan Yang Maha Esa, keluarga, para dosen, rekan-rekan sesama mahasiswa, serta teman-teman yang sudah memberi motivasi, baik di dalam maupun di luar UNPAR.*



## KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan karunia-Nya, yang memungkinkan diselesaikannya penulisan skripsi yang berjudul “Perkakas *Command Line KIRI*” ini. Selama penulisan skripsi ini tentunya penulis menemui berbagai macam kesulitan serta menghadapi berbagai macam halangan, yang puji syukur dapat diselesaikan dengan baik dengan bantuan beberapa pihak. Pada kesempatan ini, penulis ingin mengucapkan terima kasih sebesar-besarnya kepada pihak-pihak tersebut, yaitu:

- Bapak Pascal Alfadian Nugroho, M.Comp. selaku dosen pembimbing yang telah membimbing serta membantu memberikan arahan kepada penulis dalam seluruh proses pembuatan skripsi ini.
- Ibu Vania Natali, M.T. serta Bapak Lionov, Ph. D. selaku dosen-dosen penguji yang telah memeriksa hasil skripsi ini serta memberikan kritik dan saran yang membangun.
- Teman-teman penulis—Daniel, Yalvi, Rama, JJ, serta teman-teman lainnya, yang telah memberikan dorongan, semangat, serta saran tambahan selama penulisan skripsi ini.
- Keluarga penulis, yang telah memberikan bantuan dan dukungan dalam seluruh aspek lainnya yang tidak berhubungan langsung dengan penulisan skripsi ini.
- Para staf tata usaha, baik dalam tata usaha FTIS maupun BAA, yang telah membantu penyelesaian semua urusan non-akademik dalam proses pembuatan skripsi ini.

Terakhir, penulis menyadari bahwa penulisan skripsi ini jauh dari sempurna, karena berbagai macam keterbatasan yang ada. Oleh karena itu, pada kesempatan ini penulis juga ingin mengucapkan permohonan maaf atas kekurangan-kekurangan yang ada. Walaupun begitu, penulis berharap skripsi ini dapat bermanfaat bagi seluruh pihak yang membacanya.

Bandung, Januari 2023

Penulis



# DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR KODE PROGRAM</b>	<b>xxi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	3
1.3 Tujuan . . . . .	3
1.4 Batasan Masalah . . . . .	3
1.5 Metodologi . . . . .	4
1.6 Sistematika Pembahasan . . . . .	4
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 KIRI . . . . .	5
2.1.1 Tampilan . . . . .	5
2.1.2 API . . . . .	6
2.2 Pengantar <i>Command Line Interface</i> . . . . .	11
2.2.1 <i>Command Line Interface</i> dan <i>Graphical User Interface</i> . . . . .	12
2.2.2 <i>Command Line</i> di Linux . . . . .	13
2.2.3 <i>Command Line</i> di Windows . . . . .	15
2.3 Fungsi dan <i>Library</i> Bahasa C . . . . .	18
2.3.1 getopt . . . . .	18
2.3.2 libcurl . . . . .	21
2.3.3 cJSON . . . . .	23
2.3.4 CMake . . . . .	27
<b>3 ANALISIS</b>	<b>33</b>
3.1 Analisis Aplikasi Sejenis . . . . .	33
3.1.1 Chrome <i>Web Store Item Property</i> CLI . . . . .	33
3.1.2 iTunes <i>Search API</i> . . . . .	35
3.1.3 Uber CLI . . . . .	36
3.1.4 Google <i>Maps Direction</i> CLI . . . . .	39
3.2 Analisis API KIRI . . . . .	40
3.2.1 <i>Search Place</i> . . . . .	40
3.2.2 <i>Routing</i> . . . . .	40
3.3 Analisis Perkakas yang Dibuat . . . . .	41
3.3.1 Analisis Fitur Perkakas . . . . .	41
3.3.2 Analisis Kebutuhan Perkakas . . . . .	44

<b>4 PERANCANGAN</b>	<b>49</b>
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 . . . . .	50
4.2 Rancangan Implementasi Perkakas . . . . .	51
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> . . . . .	58
4.2.9 <code>execute_curl()</code> . . . . .	58
4.2.10 <code>print_curl_error()</code> . . . . .	58
4.2.11 <code>write_malloc()</code> . . . . .	58
4.2.12 <code>write_searchplace()</code> . . . . .	59
4.2.13 <code>write_findroute()</code> . . . . .	59
4.2.14 <code>write_searchplace_noreturns()</code> . . . . .	60
4.2.15 Fungsi utama ( <code>main</code> ) . . . . .	60
<b>5 IMPLEMENTASI DAN PENGUJIAN</b>	<b>69</b>
5.1 Implementasi Kode . . . . .	69
5.1.1 <code>print_help()</code> . . . . .	69
5.1.2 <code>replace_space()</code> . . . . .	69
5.1.3 <code>build_url_searchplace()</code> . . . . .	69
5.1.4 <code>build_url_findroute()</code> . . . . .	69
5.1.5 <code>reset_url()</code> . . . . .	69
5.1.6 <code>execute_curl()</code> . . . . .	70
5.1.7 <code>print_curl_error()</code> . . . . .	70
5.1.8 <code>write_malloc()</code> . . . . .	70
5.1.9 <code>write_searchplace()</code> . . . . .	70
5.1.10 <code>write_findroute()</code> . . . . .	71
5.1.11 <code>write_searchplace_noreturns()</code> . . . . .	71
5.1.12 Fungsi utama ( <code>main</code> ) . . . . .	71
5.1.13 CMakeLists . . . . .	71
5.1.14 Halaman manual ( <i>man page</i> ) . . . . .	71
5.2 Pengujian . . . . .	72
5.2.1 Lingkungan Perangkat Keras . . . . .	72
5.2.2 Lingkungan Perangkat Lunak . . . . .	72
5.2.3 Pembangunan dan Instalasi . . . . .	72
5.2.4 Pengujian . . . . .	74
<b>6 KESIMPULAN</b>	<b>89</b>
6.1 Kesimpulan . . . . .	89
6.2 Saran . . . . .	89
<b>DAFTAR REFERENSI</b>	<b>91</b>
<b>A KODE PERKAKAS <i>Command Line KIRI</i></b>	<b>93</b>

## DAFTAR GAMBAR

1.1	Unit angkot . . . . .	1
1.2	Tampilan halaman web KIRI . . . . .	2
2.1	Tampilan awal halaman web KIRI . . . . .	5
2.2	Tampilan akhir halaman web KIRI . . . . .	6
2.3	Halaman web <i>API Keys</i> KIRI. . . . .	7
2.4	Penggunaan API KIRI untuk layanan pencarian lokasi . . . . .	8
2.5	Penggunaan API KIRI untuk layanan pencarian rute . . . . .	10
2.6	Penggunaan API KIRI untuk layanan <i>smart direction</i> . . . . .	11
2.7	Contoh CLI . . . . .	12
2.8	Dua jenis antarmuka perangkat lunak . . . . .	12
2.9	Baris <i>shell prompt</i> terminal di sistem operasi Linux. . . . .	13
2.10	Contoh keluaran <i>man page</i> . . . . .	15
2.11	Tampang kedua antarmuka <i>command line</i> bawaan di sistem operasi Windows. . . . .	16
2.12	Tampilan aplikasi cmake-gui . . . . .	29
2.13	Tampilan aplikasi ccmake . . . . .	30
3.1	Contoh penggunaan perkakas Chrome <i>Web Store Item Property</i> CLI . . . . .	34
3.2	Opsi bantuan penggunaan pada perkakas Chrome <i>Web Store Item Property</i> CLI . .	35
3.3	Opsi bantuan penggunaan pada perkakas <i>iTunes Search API</i> . . . . .	36
3.4	Contoh penggunaan perkakas <i>iTunes Search API</i> . . . . .	37
3.5	Contoh penggunaan perkakas Uber CLI( <i>time</i> ) . . . . .	37
3.6	Contoh penggunaan perkakas Uber CLI( <i>price</i> ) . . . . .	38
3.7	Contoh penggunaan perkakas Google <i>Maps Direction</i> CLI . . . . .	39
3.8	Diagram <i>use case</i> perkakas yang dibangun . . . . .	45
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 . . . . .	19
2.2	Contoh sederhana penggunaan getopt_long . . . . .	20
2.3	Loop sederhana dari penggunaan <i>multi handle curl</i> . . . . .	22
2.4	Kumpulan implementasi penggunaan <i>multi socket handle curl</i> . . . . .	23
2.5	Struktur data cJSON . . . . .	25
2.6	Kode utama operasional CMake . . . . .	28
2.7	Contoh kode pembangunan CMake lebih dari satu mode . . . . .	31
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



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Zaman sekarang ada banyak masalah-masalah terkait transportasi yang sudah mulai mempengaruhi hampir semua orang, seperti pemanasan global akibat gas emisi kendaraan, kemacetan di mana-mana, dan bagi pemilik kendaraan bermotor, semakin meningginya harga BBM (bahan bakar minyak). Upaya dari pemerintah untuk menyelesaikan atau meringankan masalah-masalah ini adalah dengan mendorong masyarakat untuk menggunakan sarana transportasi publik. Di Indonesia ada beberapa jenis transportasi umum, sama seperti di negara-negara lain, seperti kereta api, bus, taksi, tetapi satu jenis transportasi umum belum tentu ada di negara-negara lain adalah angkutan kota, atau sering kali disingkat menjadi “angkot”.

Angkot (dapat dilihat di Gambar 1.1) merupakan sebuah unit transportasi umum yang metode operasinya menyerupai bus, hanya saja penumpang angkot dapat meminta pengemudinya untuk turun di mana saja, selama lokasinya masih berada di dalam rute yang sudah ditentukan untuk unit angkot tersebut. Hal ini membuat rute angkot esensial untuk diketahui, karena pengetahuan mengenai lokasi mana saja yang akan dilewati unit-unit angkot yang dinaiki dapat menghemat waktu perjalanan—apabila lokasi yang ingin dituju dilewati oleh unit angkot tersebut, tetapi bukan merupakan pemberhentian akhirnya, penumpangnya dapat langsung meminta pengendara angkot untuk berhenti di lokasi tersebut. Sistem ini menimbulkan dua masalah lain. Pertama, untuk pergi dari suatu lokasi ke lokasi lain, pengguna juga harus mengetahui unit-unit angkot mana saja yang harus dinaiki. Kedua, untuk mengefektifkan waktu perjalanan dengan angkot, penggunanya juga harus mengetahui persis lokasi-lokasi mana saja yang dilewati oleh unit angkot yang dinaiki.



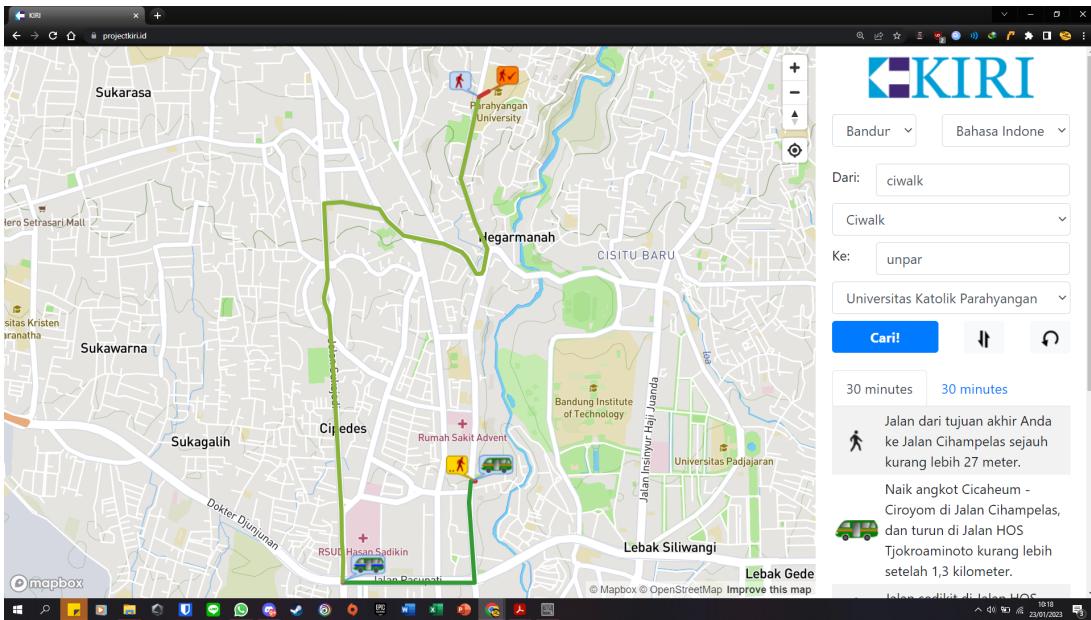
Gambar 1.1: Unit mobil angkutan kota (angkot).

Masalah inilah yang merupakan salah satu tujuan dari Project KIRI. Project KIRI<sup>1</sup> (akan disingkat sebagai KIRI dalam dokumen ini) adalah sebuah perangkat lunak berbasis *web* yang dibuat untuk membantu penggunanya, baik masyarakat maupun turis, dalam menggunakan angkot. Cara

---

<sup>1</sup><https://projectkiri.id>

KIRI mempermudah penggunaan angkot adalah dengan menunjukkan rute yang akan ditempuh, beserta langkah-langkah yang harus dilakukan oleh pengguna yang ingin berpergian dari satu titik ke titik lain, mulai dari seberapa jauh pengguna harus berjalan untuk menaiki angkot yang bersangkutan, di mana pengguna harus naik atau turun, seberapa jauh lagi pengguna harus berjalan sampai ke titik tujuan, dan seberapa lama estimasi waktu perjalanan yang akan ditempuh—semua di halaman *web* dari KIRI, yang dapat dilihat di Gambar 1.2. Walaupun begitu, dalam kasus-kasus tertentu KIRI memiliki berbagai keterbatasan, misalnya pencarian lokasi tidak akurat, atau rute angkot tidak berhasil ditemukan (walaupun bisa jadi ada rute angkot yang cukup dekat dengan lokasi awal dan tujuan).



Gambar 1.2: Tampilan halaman web KIRI, yang menunjukkan rute dari Cihampelas Walk ke Universitas Katolik Parahyangan.

Pada saat penulisan skripsi ini, KIRI hanya dapat diakses langsung melalui halaman *web*nya, di mana aplikasi ini memiliki banyak bantuan elemen-elemen grafis, yang menjadikannya sebuah aplikasi berbasis antarmuka pengguna grafis (*graphical user interface/GUI*). GUI ini hanya merupakan satu dari dua jenis antarmuka perangkat lunak, dengan jenis antarmuka satu lagi berupa antarmuka baris perintah (*command line interface/CLI*). *Command line interface* adalah sebuah antarmuka yang berupa sebuah *window* yang memuat teks berupa perintah-perintah,<sup>2</sup> yang menerima masukan dari pengguna dan menjalankannya [1]. Perintah-perintah ini hanya berupa gabungan dari teks dan simbol-simbol berupa karakter. Singkatnya, tipe perangkat lunak ini bukan merupakan tipe yang paling indah untuk dilihat oleh para pengguna, tetapi jika digunakan dengan tepat, maka jenis perangkat lunak ini bisa menyuruh komputer untuk melakukan banyak sekali perintah-perintah dengan sangat cepat dan sangat efektif [2].

Walaupun tipe antarmuka CLI muncul lebih awal dari GUI, sampai sekarang juga masih banyak perangkat-perangkat lunak yang memiliki versi CLI, atau bahkan hanya berbentuk CLI. Ada beberapa alasan CLI masih dipakai di perangkat-perangkat lunak modern, seperti [3]:

- penggunaannya lebih cepat dan sederhana,
- selalu kompatibel di berbagai sistem operasi, bahkan di instalasi-instalasi yang paling mendasar, dan
- kesederhanaannya membuat CLI lebih ideal digunakan untuk tugas-tugas di mana kemudahan konfigurasi lebih penting dari efisiensi dibandingkan GUI.

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

Selain aplikasi *web*-nya sendiri, KIRI juga memiliki antarmuka pemrograman aplikasi (*application programming interface/API*) yang dapat digunakan untuk tujuan pengembangan perangkat lunak<sup>3</sup>. API ini merupakan sebuah antarmuka logikal ke perangkat lunak serta menyembunyikan detail-detail internal implementasinya. Dalam kata lain, API menyediakan sebuah abstraksi untuk sebuah masalah serta mendiktekan bagaimana klien/pengguna harus berinteraksi dengan komponen perangkat lunak yang menyelesaikan masalah tersebut. Secara esensi, API akan mendefinisikan bagian-bagian yang dapat digunakan ulang, yang memungkinkan potongan-potongan fungsi modular yang dapat langsung diimplementasikan ke perangkat-perangkat lunak lainnya [4]. Khusus untuk kasus ini, API KIRI dapat digunakan dengan mengirimkan permintaan GET, dan nantinya akan mengeluarkan keluaran berupa objek JSON<sup>4</sup>.

Pada skripsi ini akan dibuat sebuah perangkat lunak berupa perkakas *command line* (*command line tool*) yang dapat menjalankan fungsi-fungsi API dari KIRI. Perangkat lunak ini, seperti jenisnya, akan dibuat murni sebagai perkakas yang dijalankan dari *command line* (Terminal, cmd, PowerShell, dll.). Keseluruhan dari perangkat lunak ini akan dibangun dalam bahasa C—bahasa C dipakai karena penggunaan bahasa ini memungkinkan pengaturan manual besar memori sistem yang dipakai oleh perkakas [5] (karena perkakas termasuk perangkat lunak yang ringan, maka memori maksimum yang boleh digunakan dapat diatur sekecil mungkin.) Selain itu, perkakas ini juga akan menggunakan bantuan fungsi bahasa C, yaitu getopt untuk penerimaan opsi-opsi masukan, serta *library-library* tambahan, seperti libcurl untuk proses transfer data, cJSON untuk mem-parse keluaran API, dan CMake untuk kompatibilitas antar sistem operasi. Bagaimana dan kapan persisnya modul-modul ini akan digunakan di dalam perkakas yang akan dibuat akan dibahas di bagian perancangan. Terakhir, perkakas akan memiliki fitur-fitur perkakas *command line* pada umumnya, seperti halaman/mode bantuan, serta akan memiliki kemampuan integrasi dengan perkakas-perkakas *command line* lainnya, dalam arti bahwa keluaran dari perkakas ini akan bisa digunakan sebagai masukan untuk perkakas-perkakas *command line* lainnya yang sudah ada, seperti *pipeline* (>) atau grep.

## 1.2 Rumusan Masalah

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

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

## 1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

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

## 1.4 Batasan Masalah

Perkakas yang dibuat hanya akan menyelesaikan batasan yang berhubungan langsung dengan format keluaran (kesalahan terjemahan) yang sudah sejak awal terdapat dalam KIRI. Batasan-batasan lain yang tidak berhubungan dengan format keluaran (lokasi tidak terdeteksi, rute tidak berhasil ditemukan, dsb.) akan dituliskan di keluaran perkakas apa adanya.

---

<sup>3</sup><https://projectkiri.id>

<sup>4</sup><https://github.com/projectkiri/Tirtayasa/wiki/KIRI-API-v2>

## 1.5 Metodologi

Metodologi yang akan diikuti dalam skripsi ini adalah sebagai berikut:

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

## 1.6 Sistematika Pembahasan

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

1. **Bab 1: Pendahuluan**  
Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
2. **Bab 2: Dasar Teori**  
Bab ini berisi pembahasan-pembahasan teoretis mengenai aspek-aspek yang akan dirujuk di dalam skripsi ini, seperti *command line*, bahasa C, dan juga KIRI.
3. **Bab 3: Analisis**  
Bab ini berisi analisis perkakas-perkakas sejenis, analisis API KIRI, serta analisis fitur-fitur perkakas yang akan dibuat.
4. **Bab 4: Perancangan**  
Bab ini berisi pembahasan mengenai rancangan cara kerja tiap-tiap fitur serta fungsi-fungsi dalam kode perkakas yang akan dibuat.
5. **Bab 5: Implementasi dan Pengujian**  
Bab ini berisi dua bagian utama, yaitu:
  - **Implementasi**  
Bagian ini meliputi struktur kelas dan penjelasan tiap-tiap fungsi di dalamnya.
  - **Pengujian**  
Bagian ini meliputi cara instalasi, cara menggunakan perkakas, serta pengujian fungsional terhadap fitur-fitur dari perkakas yang telah dibuat.
6. **Bab 6: Kesimpulan dan Saran**  
Bab ini berisi kesimpulan hasil pembuatan perangkat lunak dan saran-saran terhadap hasil perangkat lunak yang diberikan selama penggerjaan skripsi.

## BAB 2

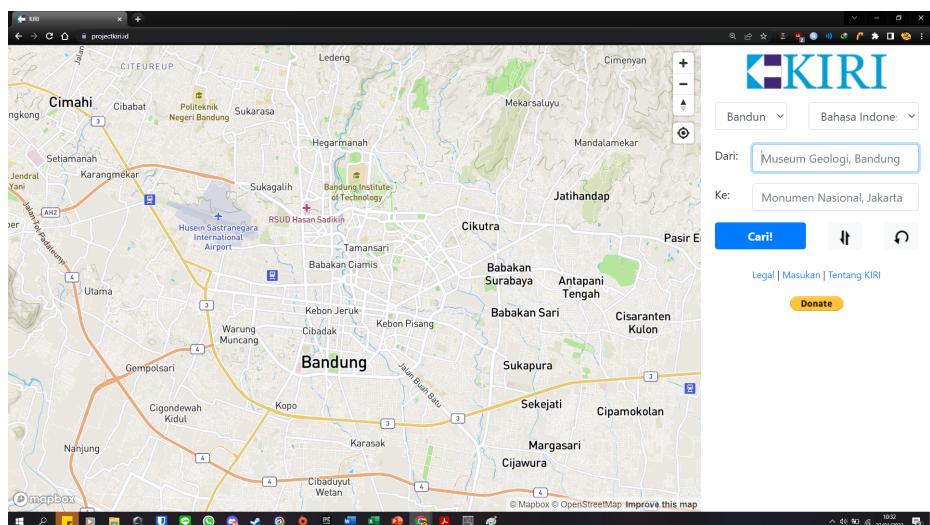
# LANDASAN TEORI

### 2.1 KIRI

KIRI merupakan sebuah perangkat lunak berbasis *web* yang berfungsi untuk menyelesaikan (atau setidaknya mengurangi) dampak dari masalah-masalah yang dapat diselesaikan oleh transportasi umum/publik di Indonesia, seperti pemanasan global, kemacetan, atau peningkatan harga bensin. Selain itu, turis mancanegara juga memilih untuk menaiki transportasi umum, karena jenis sarana transportasi tersebut tidak hanya jauh lebih murah, tetapi juga memberikan kesempatan yang mudah kepada mereka untuk melihat seluk-beluk dari kota-kota yang mereka kunjungi.<sup>1</sup>

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

#### 2.1.1 Tampilan



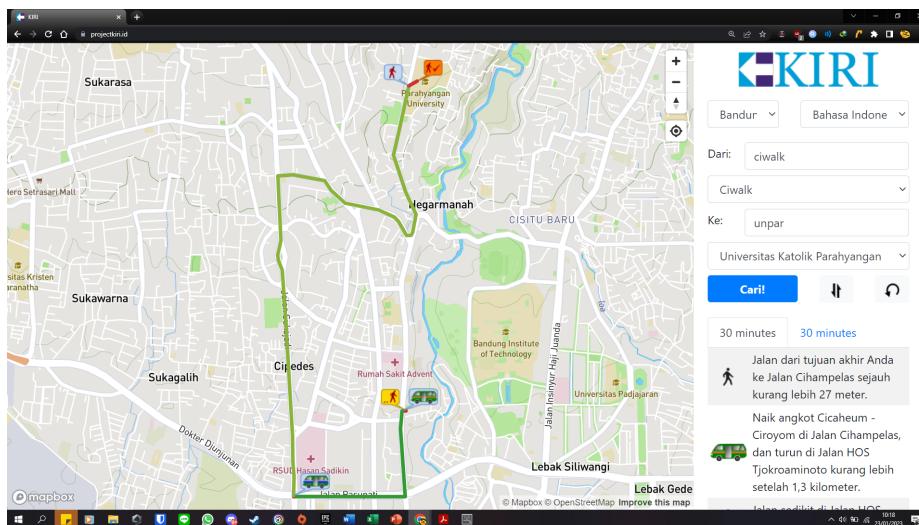
Gambar 2.1: Tampilan awal halaman web KIRI.

Pada saat pertama kali dibuka, hal pertama yang paling dominan di halaman awal *web* KIRI adalah sebuah peta besar di sebelah kiri yang dapat diperbesar ataupun diperkecil. Sedangkan, bagian kanan dari halamannya terdiri atas beberapa bagian. Di bagian paling atas terdapat logo KIRI, beserta sepasang menu *dropdown*—yang pertama merupakan pilihan kota tempat pengguna berada (untuk sekarang hanya tersedia pilihan kota Jakarta dan Bandung), dan yang kedua

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

merupakan pilihan bahasa, entah bahasa Indonesia atau Inggris. Di bawahnya merupakan sepasang menu *dropdown* yang merupakan tempat di mana pengguna memasukkan lokasi awal dan tujuan yang akan diproses oleh KIRI. Terakhir, di bawahnya ada sebuah bagian kosong, yang nantinya akan menjadi tempat di mana KIRI akan meletakkan keluaran dari prosesnya. Adapun tampilan awal dari halaman web ini dapat dilihat di Gambar 2.1.

Ada dua area yang memiliki perbedaan yang signifikan ketika pengguna sudah memasukkan masukan dan menyuruh KIRI untuk memprosesnya. Bagian yang pertama adalah bagian peta, yang setelah pemrosesan masukan, akan memiliki garis-garis berwarna yang menandakan rute angkot maupun perjalanan kaki yang harus ditempuh oleh pengguna. Bagian kedua adalah bagian keluaran, yang sekarang akan berisi langkah-langkah yang harus ditempuh oleh pengguna untuk pergi dari lokasi awal ke lokasi akhir. Spesifiknya, perbedaan-perbedaan ini dapat dilihat di Gambar 2.2.



Gambar 2.2: Tampilan halaman *web* KIRI setelah pemrosesan masukan dari pengguna selesai.

### 2.1.2 API<sup>2</sup>

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

#### *Search Place*

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

- **version**

**Kemungkinan nilai:** 2

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

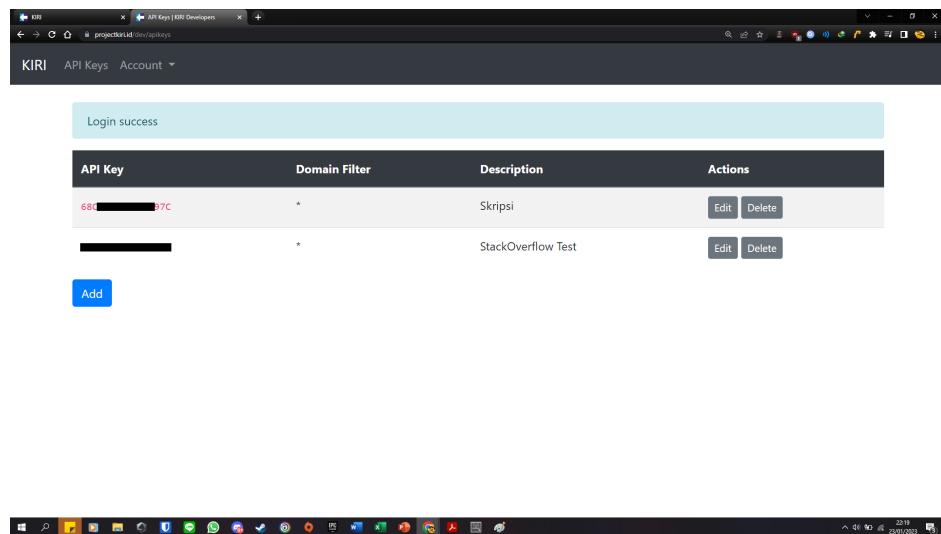
- **mode**

**Kemungkinan nilai:** **searchplace**

Parameter ini merupakan mode dari servis/jasa API yang akan digunakan oleh pengguna.

Untuk penggunaan layanan pencarian lokasi, variabel ini harus diisi dengan **searchplace**.

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



Gambar 2.3: Halaman web *API Keys* KIRI<sup>4</sup>.

- **region**

**Kemungkinan nilai:** `cgk`, `bdo`, `m1g`, atau `sub`

Parameter ini merupakan kode bandara IATA tiga huruf yang merepresentasikan daerah mana tempat lokasi yang ingin dicari berada. Kode yang dapat diproses oleh API ini meliputi `cgk` (Cengkareng/Jakarta), `bdo` (Bandoeng/Bandung), `m1g` (Malang), dan `sub` (Surabaya).

- **querystring**

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

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

- **apikey**

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

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

Salah satu dari parameter yang harus diikutkan dalam pesan tersebut merupakan parameter yang meminta kunci API. Kunci tersebut harus digenerasikan terlebih dahulu sebelum API KIRI dapat digunakan, melalui halaman *API Keys* KIRI,<sup>3</sup> yang dapat dilihat di gambar 2.3.

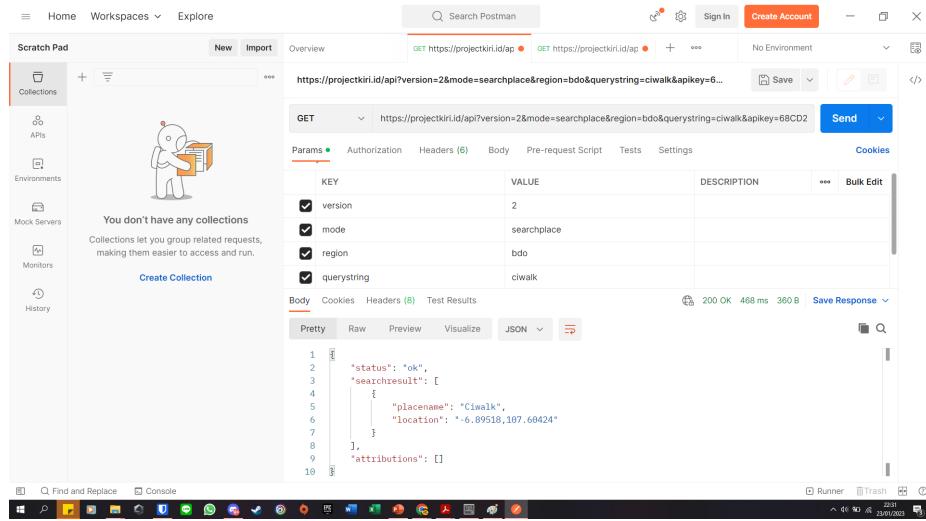
Untuk mengakses halaman tersebut, pengguna harus membuat sebuah akun terlebih dahulu. Ketika akun sudah dibuat, maka pengguna baru akan dapat membuat kunci API yang dibutuhkan, sekaligus membuat filter *domain*, yang membatasi di *domain* mana saja kunci tersebut dapat digunakan, serta memberikan deskripsi untuk kunci API tersebut. Kunci ini kemudian dapat 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.<sup>5</sup> Kedua nilai ini merupakan salah satu dari dua variabel yang dikembalikan dalam respons API untuk layanan

<sup>3</sup><https://projectkiri.id/dev/apikeys>

<sup>4</sup>Bagian tengah kunci API ditutup untuk alasan keamanan. Kunci-kunci API yang tidak berhubungan dengan perkakas di skripsi ini ditutup total.

<sup>5</sup>[https://gsp.humboldt.edu/olm/Lessons/GIS/01%20SphericalCoordinates/Latitude\\_and\\_Longitude.html](https://gsp.humboldt.edu/olm/Lessons/GIS/01%20SphericalCoordinates/Latitude_and_Longitude.html)



Gambar 2.4: Penggunaan API KIRI untuk layanan pencarian lokasi menggunakan Postman. Gambar ini menunjukkan hasil pencarian lokasi “ciwalk” di daerah Bandung.

ini, dengan variabel lainnya berupa nama dari lokasi yang ditemukan itu sendiri. Respons yang diberikan oleh API berupa sebuah objek JSON yang memiliki setidaknya dua variabel, yaitu:

- **status**

**Kemungkinan nilai:** ok atau error

Variabel ini menandakan 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 respons 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.

- **location**

Variabel ini berisi nilai *latitude* dan *longitude* dari lokasi yang ditemukan dalam pencarian.

Contoh dari penggunaan API KIRI untuk layanan ini dapat dilihat di Gambar 2.4.

### **Routing**

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

- **version**

**Kemungkinan nilai:** 2

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

- **mode**

**Kemungkinan nilai:** `findroute`

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

- **locale**

**Kemungkinan nilai:** `en` atau `id`

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

- **start**

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

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

- **finish**

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

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

- **presentation** (opsional)

**Kemungkinan nilai:** `desktop`

Parameter ini hanya digunakan untuk fitur *backwards compatibility*.

- **apikey**

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

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

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

- **status**

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

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

- **message**

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

- **steps**

**Tipe:** `array`

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

- \* Tipe transportasi

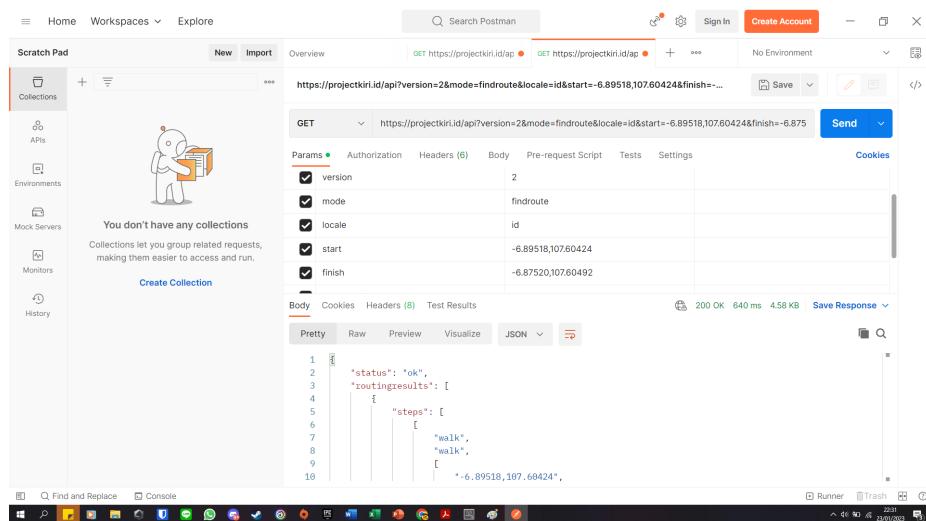
Tipe sarana transportasi yang harus dipakai oleh pengguna. Jika pengguna harus berjalan kaki, variabel ini akan berisi `walk`. Jika pengguna harus menaiki angkot, variabel ini akan berisi `angkot`.

- \* Kode angkot

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

- \* *Array latitude* dan *longitude* lokasi

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



Gambar 2.5: Penggunaan API KIRI untuk layanan pencarian rute menggunakan Postman. Gambar ini menunjukkan hasil pencarian rute dari Cihampelas Walk ke UNPAR.

- \* Deskripsi langkah

Deskripsi langkah yang harus ditempuh, dalam bahasa natural. Bahasa yang digunakan tergantung parameter `locale` yang diatur dalam masukan.

- \* URL untuk mendapatkan tiket kendaraan

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

- \* URL editor rute

Tautan untuk 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 2.5 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 apa pun—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 akan langsung menunjukkan rute yang perlu ditempuh untuk pergi dari lokasi awal ke lokasi akhir. Untuk melakukan 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:

- `start`

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

- **finish**

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

- **locale** (opsional)

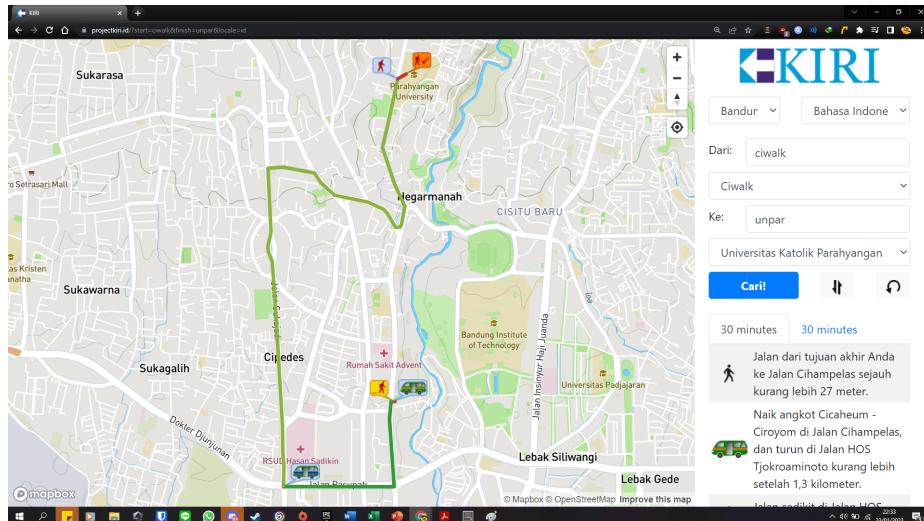
**Kemungkinan nilai:** id atau en

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

Misalkan pengguna ingin mencari rute dari “ciwalk” (Cihampelas Walk) ke “unpar” (Universitas Katolik Parahyangan), dan menampilkan langkah-langkah yang harus ditempuh dalam rutenya dalam bahasa Indonesia. Pengguna dapat memasukkan URL berikut ke peramban mereka.

```
https://projectkiri.id?start=ciwalk&finish=unpar&locale=id
```

Jika URL tersebut sudah dimasukkan ke kotak *link* pada peramban, halaman yang akan ditampilkan akan terlihat seperti pada gambar 2.6.



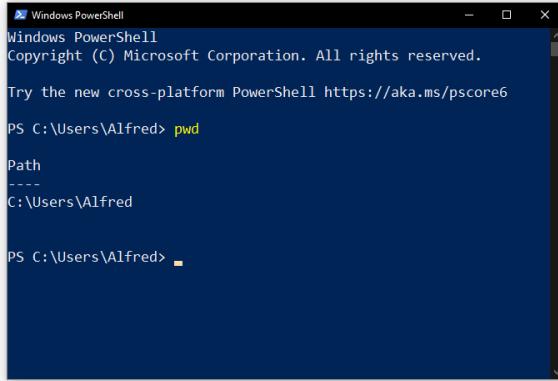
Gambar 2.6: Penggunaan API KIRI untuk layanan *smart direction*, dari Cihampelas Walk ke UNPAR.

## 2.2 Pengantar Command Line Interface

*Command line* (atau *command line interface*, disingkat CLI) dapat diartikan sebagai tampilan antarmuka yang memproses perintah dari pengguna dan meneruskannya langsung ke sistem operasi untuk dijalankan [2]. Seluruh sistem operasi komputer yang ada memiliki sebuah CLI dalam bentuk *shell*, yang dapat digunakan oleh penggunanya untuk langsung mengakses fungsi atau servis yang disediakan oleh sistem operasi [6]. Karakteristik-karakteristik utama dari CLI adalah sebagai berikut [1] [6].

- Pengguna menjalankan pekerjaan-pekerjaan yang ingin dilakukan dengan memasukkan (mengetikkan) perintah tertentu.
- Pekerjaan akan dijalankan (dieksekusi) oleh sistem sampai selesai, sampai terjadi *error*, atau sampai pekerjaannya dihentikan oleh pengguna.
- Ketika pekerjaan telah selesai, keluaran (jika ada) akan ditampilkan, dan kemudian pengguna akan diminta memasukkan perintah berikutnya.

Salah satu dari banyak contoh CLI adalah Windows PowerShell, yang dapat dilihat di Gambar 2.7.

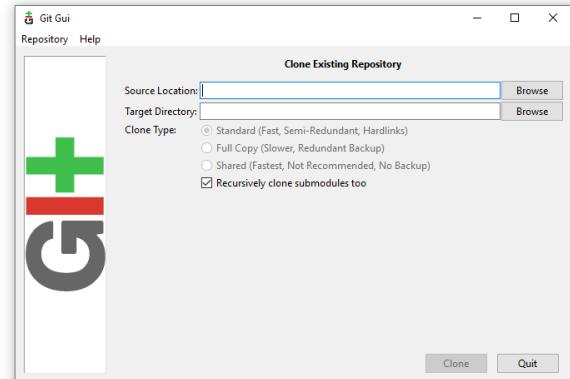


Gambar 2.7: Salah satu contoh CLI, yaitu Windows PowerShell.

### 2.2.1 *Command Line Interface* dan *Graphical User Interface*

Dua tipe antarmuka utama untuk penggunaan perangkat lunak adalah *command line interface* (CLI) dan *graphical user interface* (GUI). CLI merupakan tipe antarmuka yang lebih awal muncul, dalam bentuk sebuah terminal teks. Masukan untuk terminal tersebut akan dimasukkan oleh pengguna (umumnya menggunakan sebuah *keyboard*), dan keluarannya juga akan berupa teks yang ditampilkan langsung di layar perangkat, seperti dapat dilihat di Gambar 2.8a. Seiring berjalannya waktu serta kemajuan teknologi, muncul tipe antarmuka lain, yaitu GUI, yang merupakan antarmuka dengan tambahan berbagai macam elemen-elemen grafis, seperti tertera di Gambar 2.8b [7].

```
MINGW64/c/Users/Alfred/Desktop
$ pwd
/c/Users/Alfred/Desktop
$
```



(a) Antarmuka Git berbasis *command line interface*.

(b) Antarmuka Git berbasis *graphical user interface*.

Gambar 2.8: Kedua jenis antarmuka perangkat lunak untuk perangkat lunak Git.

Selain dari tampilannya sendiri, ada beberapa perbedaan lain antara antarmuka CLI dengan antarmuka 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 dengan antarmuka berbasis CLI lebih rendah dibandingkan dengan perangkat lunak dengan antarmuka berbasis GUI.
- Bagi pengguna pemula (atau pengguna awam pada umumnya), perangkat lunak CLI akan lebih sulit digunakan karena tidak adanya bantuan apa pun 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 dengan antarmuka berbasis CLI lebih sulit untuk dibiasakan penggunaannya.

- Automasi perintah yang bersifat berulang-ulang jauh lebih mudah dilakukan pada perangkat lunak CLI. Hal ini dikarenakan perangkat lunak CLI 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, karena penggunaannya cukup dengan mengetikkan perintah dan menyuruh sistem untuk menjalankannya.

### 2.2.2 Command Line di Linux

Linux merupakan sebuah sistem operasi yang sangat modular, jadi ada banyak sekali tipe *shell* yang dapat dijalankan dan digunakan di dalamnya, seperti Bourne Shell (sh), C shell (csh), Z shell (zsh), dan sebagainya. Walaupun begitu, ada satu *shell* yang selalu datang ter-*install* di dalam semua sistem operasi Linux, yaitu “*bash*” (GNU Bourne Again Shell) [3]. *Shell* jenis ini yang akan dipakai sebagai basis untuk pembahasan ini.

#### 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. 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* [2]. Perbedaan tampilan ini dapat dilihat dengan lebih jelas di Gambar 2.9.

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

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

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

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

Gambar 2.9: Baris *shell prompt* terminal di sistem operasi Linux.

#### Navigasi [2]

Sistem-sistem operasi berbasis Linux menyimpan file-filenya di sebuah struktur direktori yang bersifat hierarkial. Hal ini berarti bahwa file-file tersebut disimpan dalam direktori-direktori (atau *folder-folder*) yang tersusun seperti sebuah pohon. dalam arti bahwa satu *folder* bisa jadi berada di dalam 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 mana pun di direktori mana pun, 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* (satu tingkat di atas) dari *working directory*.

### **man [1] [2]**

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 halaman 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>
```

*Man page* tidak memiliki format umum, tapi pada umumnya memiliki bagian-bagian berikut:

- judul—perangkat lunak mana yang sedang ditampilkan halaman *man*-nya,
- perintah penggunaan perangkat lunak,
- deskripsi singkat dari fungsi perangkat lunak, serta
- daftar fitur-fitur perangkat lunak serta cara penggunaannya.

Walaupun begitu, perlu diperhatikan bahwa halaman `man` umumnya tidak mengikutkan contoh penggunaan, dalam arti bahwa contoh yang diberikan hanya sebatas format penggunaan perintah yang ditampilkan halaman manualnya. Selain itu, halaman-halaman manual ini selalu diletakkan ke satu (atau lebih) dari delapan kategori, di mana kategori-kategori tersebut dapat dilihat di Tabel 2.1.

Tabel 2.1: Jumlah kategori dan tes yang dilakukan.

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

Adapun contoh penggunaan perintah `man` ini ada di Gambar 2.10.

```

ls(1)                               User Commands                               LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options
    too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
        Manual page ls(1) line 1 (press h for help or q to quit)

```

Gambar 2.10: Contoh keluaran `man` untuk perintah `ls`.

Selain `man`, ada juga perintah yaitu `whatis`, yang akan menampilkan deskripsi satu baris untuk perangkat lunak yang dipilih. Misal, jika pengguna memasukkan perintah berikut:

```
whatis ls
```

Hasilnya akan menyerupai isi dari bagian `NAME` untuk perintah `man ls` dengan tambahan kategori mana perintah tersebut berada, yaitu sebagai berikut.

```
ls (1)          - list directory contents
```

### 2.2.3 *Command Line* di Windows

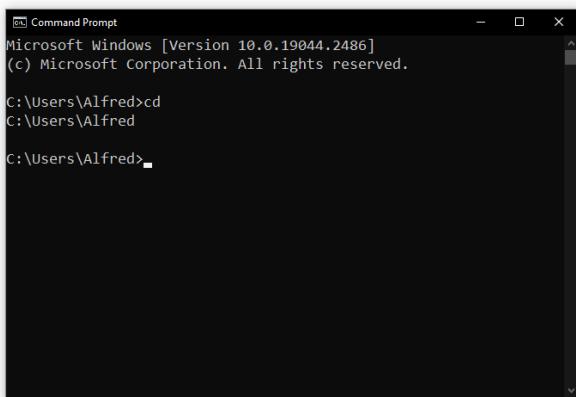
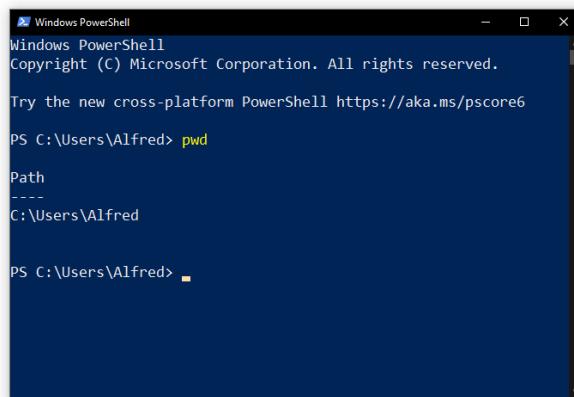
Cara kerja CLI di Windows serupa dengan cara kerja CLI di Linux, dalam arti bahwa untuk bekerja dengan *command line* di Windows, penggunanya juga akan langsung berinteraksi dengan utilitas yang disediakan oleh sistem operasi. *Command line* di Windows juga dapat digunakan untuk hal-hal yang serupa dengan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

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

#### Jenis [6]

Di sistem operasi Windows, ada dua jenis CLI, yaitu *Command Prompt* (cmd) dan Windows PowerShell, yang keduanya dapat dilihat di Gambar 2.11. Menurut tampilannya, keduanya sama saja—tampilan awalnya berbeda, tetapi keduanya sama-sama bisa dimodifikasi sesuai keinginan pengguna.

Akan tetapi, bukan berarti keduanya merupakan versi yang berbeda dari perangkat lunak yang sama. Perbedaan utama antara cmd dan Windows PowerShell, adalah bahwa beberapa perintah untuk fungsi yang sama bisa memiliki nama yang berbeda, atau memiliki perintah sekunder dengan nama yang berbeda. Misal perintah untuk mengganti *working directory*, `cd`. Di PowerShell ada perintah yaitu `set-location` yang memiliki fungsi yang sama, tetapi `set-location` ini tidak berfungsi di cmd. Dalam beberapa kasus, seperti `help`, keluarannya pun berbeda juga dengan fungsi dasarnya, yaitu `/?`.

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

(b) Antarmuka Windows PowerShell

Gambar 2.11: Tampang kedua antarmuka *command line*bawaan di sistem operasi Windows.

## Navigasi

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

- **cd (chdir)** [8]

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

Format dari perintah **cd** adalah sebagai berikut.

```
cd [/d] [<drive>:] [<path>]
```

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

- **/d**

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

- **<drive>:**

Kode huruf dari partisi yang akan diproses.

- **<path>**

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

- **dir**

**dir** merupakan sebuah perintah yang mengeluarkan/menampilkan sebuah daftar berisi file-file yang ada di suatu direktori, termasuk subdirektori. Jika tidak disertai parameter apa pun, perintah ini akan menampilkan label volume dan nomor serial *disk*, dilanjutkan dengan daftar direktori dan file di dalamnya. Untuk file, akan ditampilkan nama beserta ukurannya. Perintah ini juga akan menampilkan jumlah direktori dan file yang didaftarkan, ukuran kumulatifnya, dan sisa dari *disk* yang tidak terpakai (dalam *bytes*) [8].

Format dari perintah **dir** adalah sebagai berikut [6].

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

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

- /A[:<attributes>]  
Menampilkan file-file dengan atribut tertentu, seperti file yang disembunyikan, file sistem, file *read-only*, dan sebagainya.
- /B  
Menghilangkan *heading* dan ringkasan informasi dari keluaran, atau dengan kata lain, hanya menampilkan file-file dan direktori, tanpa informasi tambahan apapun.
- /C  
Menggunakan separator koma untuk tiap angka ribuan di ukuran file. Jika opsi yang dimasukkan adalah /-C, separator koma justru akan dihilangkan.
- /D  
Menampilkan keluaran dengan format yang lebih lebar. Jika opsi ini diikutkan, keluaran akan ditampilkan dengan urutan berdasarkan kolom.
- /L  
Seluruh teks dalam keluaran akan menggunakan huruf kecil. Jika opsi ini tidak digunakan, keluaran akan mengandung huruf besar dan huruf kecil *mixed case*.
- /N  
Menampilkan daftar dengan format panjang, dengan nama file berada di ujung paling kanan.
- /O[:<sortorder>]  
Menampilkan daftar direktori yang diurutkan berdasarkan urutan tertentu, seperti berdasarkan ekstensi file, berdasarkan tanggal dibuat, berdasarkan nama, dan sebagainya. Jika tidak diikutkan tanda minus (-) sebelum huruf O pada perintah, daftar yang muncul akan diurutkan secara menaik.
- /P  
Memberhentikan keluaran selama beberapa waktu singkat (memberi jeda kecil) setelah setiap halaman informasi.
- /Q  
Menambahkan informasi mengenai pemilik file dalam keluaran.
- /R  
Menampilkan *data stream* alternatif, jika ada.
- /S  
Mendaftarkan seluruh file di direktori dan subdirektori yang diproses. Tiap-tiap direktori akan memiliki *header* tersendiri dalam keluarannya.
- /T[:<timefield>]  
Menspesifikasi *time field* mana yang akan tampil dan digunakan sebagai urutan, jika aturan pengurutan lain tidak ditentukan. *Time field* yang dapat digunakan adalah waktu pembuatan file, kapan terakhir file diakses, dan kapan file terakhir dimodifikasi. Jika parameter ini tidak dispesifikasikan, *time field* yang digunakan adalah kapan file terakhir dimodifikasi.
- /W  
Menampilkan keluaran dengan format yang lebih lebar. Opsi ini hampir sama dengan /D, hanya saja untuk /W, jika opsi ini diikutkan, keluaran akan ditampilkan dengan urutan berdasarkan baris, dan bukan kolom.
- /X  
Menampilkan nama pendek yang dibuat untuk nama-nama file non-8.3. Opsi ini memiliki format tampilan yang sama seperti opsi /N, hanya saja nama pendeknya ditampilkan di keluaran sebelum nama panjangnya.
- /4  
Menampilkan angka tahun dengan format angka empat digit.

## 2.3 Fungsi dan *Library* Bahasa C

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

### 2.3.1 getopt [9]

`getopt` merupakan sebuah fungsi yang dapat mengautomasikan pekerjaan-pekerjaan yang berhubungan dengan penerimaan opsi-opsi untuk *command line* berbasis UNIX.

Fungsi `getopt` dapat dipanggil dengan format sebagai berikut.

```
getopt (int argc, char *const *argv, const char *<options>)
```

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.

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<sup>6</sup>.

- `opterr`  
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.
- `optopt`  
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.
- `optind`  
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 argumen tambahan yang tidak terpakai. Nilai dari variabel ini dimulai dari 1.
- `optarg`  
Jika opsi yang sedang diproses memerlukan argumen, variabel ini adalah tempat dimana argumen tersebut akan disimpan.
- `<options>`  
Variabel ini merupakan salah satu variabel yang tertera di format pemanggilan `getopt` di atas. Variabel ini berupa *string* yang menandakan karakter-karakter apa saja yang menjadi opsi yang mungkin dalam perkakas tersebut, beserta tipenya. Jika karakter opsi:
  - Diikuti dengan titik dua (:), maka opsi tersebut memiliki argumen yang bersifat wajib.
  - Diikuti dengan titik dua ganda (::), maka opsi tersebut memiliki argumen yang bersifat opsional.
  - Tidak diikuti apa-apa, maka opsi tersebut merupakan opsi tidak berargumen.

Untuk memberikan gambaran yang lebih baik mengenai penggunaan fungsi `getopt`, Kode 2.1 merupakan contoh perkakas CLI sederhana yang menggunakan fungsi tersebut. Penjelasan singkat dari perkakas ini adalah sebagai berikut.

- Perangkat lunak ini akan menerima masukan berupa opsi tidak berparameter `-a` dan/atau opsi berparameter `-b`.

<sup>6</sup>[https://www.gnu.org/software/libc/manual/html\\_node/Using-Getopt.html](https://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html)

- Untuk opsi `-a`, perangkat lunak ini akan mengeluarkan nilai 0 jika opsi `-a` tidak dipakai, dan 1 jika opsi tersebut dipakai.
- Untuk opsi `-b`, perangkat lunak ini akan mengeluarkan isi argumen opsi, atau `NULL` jika opsi `-b` tidak dipakai di masukan.
- Perangkat lunak ini menerima dua variabel awal dari masukan, yaitu `argc` (jumlah opsi dalam masukan) dan `**argv` (array yang berisi opsi-opsi dalam masukan). Variabel `argv` merupakan *pointer* ke *pointer* akibat implementasi dari `getopt` (variabel merupakan pointer konstan ke variabel fungsi `main`, dan variabel fungsi `main` merupakan pointer ke masukan).
- `args` diisi opsi yang sedang diperiksa di dalam masukan. Opsi-opsi (serta argumennya, jika ada) akan terus diiterasi sampai habis.
- Jika iterasi opsi dalam masukan selesai, tampilkan keluarannya.
- Jika ada argumen yang tidak didukung (bukan `-a` atau `-b`), tampilkan semuanya setelah keluaran utama.
- Keluaran akan ditampilkan melalui *stream* `stdout`.
- Keluaran yang berupa pesan *error* fatal akan dikeluarkan melalui *stream* `stderr`.

Kode 2.1: Contoh sederhana penggunaan getopt

```

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

### getopt\_long

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

`getopt_long` juga memiliki seluruh variabel-variabel yang dimiliki oleh `getopt`, hanya saja `getopt_long` memiliki sebuah variabel tambahan berupa struktur, yaitu `long_options`. Variabel ini merupakan sebuah struktur berupa *array* yang berisi beberapa *array* lainnya, di mana *array*-*array*

lain ini merupakan masing-masing opsi dari fungsi `getopt_long` tersebut. Tiap-tiap *array* tersebut memiliki variabel-variabel berikut:

- **name**  
Variabel ini merupakan nama panjang dari opsi.
- **has\_arg**

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

- **flag & val**

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

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

Untuk menjelaskan lebih lanjut mengenai cara penggunaan `getopt_long`, Kode 2.2 merupakan contoh perkakas CLI sederhana yang menggunakan fungsi tersebut. Penjelasan singkat dari perkakas ini adalah sebagai berikut.

- Perangkat lunak ini akan menerima masukan dari penggunaan opsi-opsi serta parameternya.
- Keluaran dari perangkat lunak ini adalah opsi apa saja yang dipilih, serta parameter yang diberikan (jika ada).
- Opsi pertama yang disediakan adalah `-a` atau `--args` yang merupakan opsi tidak berargumen.
- Opsi kedua yang disediakan adalah `-n` atau `--noargs`, yang merupakan opsi yang membutuhkan sebuah argumen.
- Opsi ketiga dan keempat merupakan penanda mode keluaran, yaitu `--short` dan `--long`. Jika opsi `--long` digunakan, maka perangkat lunak ini akan mengeluarkan versi panjang dari keluaran, sedangkan jika opsi `--short` digunakan, maka perangkat akan mengeluarkan versi pendek dari keluaran.
- Perangkat lunak ini menerima dua variabel awal dari masukan, yaitu `argc` (jumlah opsi dalam masukan) dan `**argv` (array yang berisi opsi-opsi dalam masukan). Variabel `argv` merupakan *pointer* ke *pointer* akibat implementasi dari  `getopt` (variabel merupakan pointer konstan ke variabel fungsi `main`, dan variabel fungsi `main` merupakan pointer ke masukan).
- `option` diisi opsi yang sedang diperiksa di dalam masukan. Opsi-opsi (serta argumennya, jika ada) akan terus diiterasi sampai habis.
- Keluaran akan ditampilkan per iterasi sesuai dengan masing-masing kasus (opsi).
- Jika ada argumen yang tidak didukung (bukan `-a` atau `-b`), tampilkan semuanya setelah keluaran utama.
- Seluruh keluaran akan ditampilkan melalui *stream stdout*.

Kode 2.2: Contoh sederhana penggunaan `getopt_long`

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <getopt.h>
4
5 int main(int argc, char **argv) {
6     int option;
7     static int verbose;
8
9     while (1) {
10         static struct option long_options[] = {
11             {"long", 0, &verbose, 1},
12             {"short", 0, &verbose, 0},
13             {"noargs", 0, 0, 'n'},
14             {"args", 1, 0, 'a'},
15             {0, 0, 0, 0}};
16         int option_index = 0;
17         option = getopt_long(argc, argv, "na:", long_options, &option_index);
18
19         if (option == -1)
20             break;
21         switch (option) {
22             case 0:

```

```

23|         if (verbose) {
24|             printf("Print_mode_is_set_to:_%s", long_options[option_index].name);
25|         }
26|     else
27|         printf("Print_mode_is_set_to:_%s", long_options[option_index].name);
28|     putchar('\n');
29|     break;
30|
31|     case 'n':
32|         if (verbose == 1) {
33|             printf("Option_%s_was_picked._This_option_does_not_require_any_arguments.", long_options[option_index].name)
34|             ;
35|         }
36|     else
37|         printf("Argumentless_option_%s_was_picked.", long_options[option_index].name);
38|     putchar('\n');
39|     break;
40|
41|     case 'a':
42|         if (verbose == 1) {
43|             printf("Option_%s_was_picked_with_argument_%s.", long_options[option_index].name, optarg);
44|         }
45|     else
46|         printf("a=%s", optarg);
47|     putchar('\n');
48|     break;
49|
50|     case '?':
51|         break;
52|
53|     default:
54|         abort();
55|     }
56|
57|     if (optind < argc)
58|     {
59|         printf("Arguments_passed_without_a_corresponding_option_(argv):_");
60|         while (optind < argc) {
61|             printf("%s ", argv[optind++]);
62|         }
63|         putchar('\n');
64|     }
65|
66|     exit(0);
67}

```

### 2.3.2 libcurl [10]

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

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

#### *Easy Handle*

*Easy handle* dari libcurl dapat diinisialisasi dengan memanggil fungsi `curl_easy_init()`. Setelah itu, untuk mengatur opsi-opsi yang perlu diatur sesuai kebutuhan pengguna, seperti URL yang dituju, protokol yang ingin dipakai, koneksi ke port spesifik, dan sebagainya<sup>7</sup>, pengguna harus mengaturnya dengan fungsi `curl_easy_setopt()`. *Handle* ini berhasil diatur opsinya apabila fungsi `curl_easy_setopt()` tadi mengembalikan CURLE\_OK. Terakhir, untuk menjalankan transfernya, fungsi yang perlu dipanggil adalah `curl_easy_perform(<easy handle>)`, dengan variabel `<easy handle>` diisi dengan nama dari *easy handle* yang ingin dimulai transfernya.

*Handle* yang telah diatur ini dapat digunakan berulang kali dengan konfigurasi yang sama, sampai entah pengguna mengganti konfigurasi opsi-opsinya kembali, atau atau *handle*-nya direset dengan pemanggilan fungsi `curl_easy_reset()`.

<sup>7</sup>[https://curl.se/libcurl/c/curl\\_easy\\_setopt.html](https://curl.se/libcurl/c/curl_easy_setopt.html)

## Multi Handle

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

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

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

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

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

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

1. Inisialisasi variabel `transfers_running`

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

2. Menjalankan transfer dalam *multi handle*

Langkah selanjutnya adalah menjalankan transfer dalam *multi handle*, dengan memanggil fungsi `curl_multi_perform`. Adapun fungsi tersebut harus dipanggil dengan format berikut:

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

3. Menunggu transfer untuk selesai sebelum data diekstraksi

Tentunya sebelum data hasil transfer dapat diekstraksi untuk dipakai, proses transfernya sendiri harus selesai terlebih dahulu—untuk kasus dalam *multi handle* libcurl, seluruh *handle* di dalam *multi handle* harus selesai melakukan transfer terlebih dahulu, atau sampai terjadi *timeout*. Adapun langkah ini dapat diselesaikan entah dengan menggunakan `curl_multi_wait()` atau `curl_multi_poll()`, atau dengan cara manual, yaitu dengan memasukkan deskriptor-deskriptor file serta nilai *timeout* secara manual, dan kemudian menggunakan `select()`, walaupun metode ini tidak dianjurkan karena keterbatasan jumlah deskriptor yang dapat digunakan<sup>8</sup>.

Implementasi singkat dari ketiga langkah ini dapat dilihat di Kode 2.3.

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

```
1 do {
2     curl_multi_wait (multi_handle, NULL, 0, 1000, NULL);
3     curl_multi_perform (multi_handle, &transfers_running);
4 } while (transfers_running);
```

<sup>8</sup>[https://curl.se/libcurl/c/curl\\_multi\\_poll.html](https://curl.se/libcurl/c/curl_multi_poll.html)

### Multi Socket Handle

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

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

Adapun seluruh *handle* ini dapat dipanggil dengan fungsi `curl_multi_socket_action()`, dan cara untuk melihat apakah seluruh transfer sudah selesai atau masih ada transfer yang berlangsung pada suatu waktu sama dengan *multi transfer* biasa. Contoh implementasi singkat dari seluruh fungsi-fungsi tersebut dapat dilihat di Kode 2.4.

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

```

1  /* socket callback */
2  int socket_callback(CURL *easy,          /* easy handle */
3                      curl_socket_t s, /* socket */
4                      int what,           /* aksi apa yang ditunggu */
5                      void *userp,        /* penunjuk callback privat */
6                      void *socketp)      /* penunjuk socket privat */
7
8  curl_multi_setopt(multi_handle, CURLMOPT_SOCKETFUNCTION, socket_callback);
9
10 /* timer callback */
11 int timer_callback(multi_handle,    /* multi handle */
12                     timeout_ms,      /* waktu tunggu dalam milidetik */
13                     userp)           /* penunjuk callback privat */
14
15 curl_multi_setopt(multi_handle, CURLMOPT_TIMERFUNCTION, timer_callback);
16
17 /* multi socket action */
18 curl_multi_socket_action(multi, CURL_SOCKET_TIMEOUT, 0, &running);

```

### 2.3.3 cJSON<sup>9</sup>

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

#### Instalasi

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.

<sup>9</sup><https://github.com/DaveGamble/cJSON>

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

Menyalakan *SafeStack*. Pada saat skripsi ini dibuat, fitur ini hanya didukung untuk *compiler* Clang.

- `-DBUILD_SHARED_LIBS`

**Nilai awal:** Off

Membangun semua *shared library* yang tersedia.

- `-DBUILD_SHARED_AND_STATIC_LIBS`

**Nilai awal:** On

Membangun *shared library* dan *static library* yang tersedia.

- `-DCMAKE_INSTALL_PREFIX`

**Nilai awal:** -

Mengatur *prefix* direktori tempat instalasi cJSON.

- `-DENABLE_LOCALES`

**Nilai awal:** On

Memungkinkan penggunaan metode `localeconv`.

- `-DCJSON_OVERRIDE_BUILD_SHARED_LIBS`

**Nilai awal:** On

Memungkinkan penimpaan nilai dari opsi `-BUILD_SHARED_LIBS` menggunakan nilai dari opsi `-DCJSON_BUILD_SHARED_LIBS`.

- `-DENABLE_CJSON_VERSION_SO`

**Nilai awal:** On

Menyalakan versi so dari cJSON.

- Makefile

Jika CMake tidak tersedia, cJSON juga dapat dibangun dengan menggunakan GNU Make, dengan menggunakan perintah `make all`, dan menginstal *library-library* yang sudah ter-

*compile* dengan perintah `make install`. Akan tetapi, perlu diingat bahwa metode instalasi ini sudah tidak lagi diperbarui (*deprecated*), dan dukungannya hanya sebatas pembetulan *bug*.

- `vcpkg`

Melalui `vcpkg`, cJSON dapat diunduh dan diinstal secara langsung. Versi `vcpkg` dari cJSON terus diperbarui oleh tim Microsoft dan kontributor-kontributor dari komunitas publik, jadi *library* cJSON yang diinstal melalui `vcpkg` akan selalu merupakan versi terbarunya.

## Penggunaan

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

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

## Struktur Data

cJSON merepresentasikan sebuah nilai JSON dengan struktur data `cJSON`, yang dapat dilihat di Kode 2.5.

Kode 2.5: Struktur data cJSON

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

Dengan variabel-variabel dalam struktur tersebut sebagai berikut:

- `next` dan `prev`

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

- `child`

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

- `type`

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

- `cJSON_Invalid`

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

- `cJSON_True`

Merepresentasikan nilai boolean *true*.

- `cJSON_False`

Merepresentasikan nilai boolean *false*.

- `cJSON_Number`

Merepresentasikan nilai numerik apa pun. Jika nilainya merupakan nilai *double*, maka nilai tersebut akan disimpan di dalam variabel `valuedouble`. Sedangkan jika nilainya merupakan nilai *integer*, maka nilai tersebut akan disimpan di dalam variabel `valueint`.

- `cJSON_String`

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

- **cJSON\_Array**  
Merepresentasikan nilai *array*. *Array* dalam cJSON diimplementasikan dengan menunjukkan isi dari variabel **child** tadi ke sebuah *linked list* dari objek-objek cJSON. Jika objek cJSON ini merupakan elemen dalam sebuah *array*, maka isi dari variabel **prev** dan **next** merupakan salah satu dari elemen-elemen lain dalam *array* yang menjadi elemen sebelum dan sesudah elemen ini.
- **cJSON\_Object**  
Merepresentasikan nilai objek general. Implementasinya sama dengan *array*, hanya saja untuk objek general, kuncinya diletakkan di dalam variabel **string**.
- **cJSON\_Raw**  
Merepresentasikan objek JSON apa pun yang disimpan dalam bentuk *array* yang diakhiri oleh *null terminator*.
- **cJSON\_NULL**  
Merepresentasikan sebuah nilai *null*.
- **valuestring/valueint-valuemode**  
Merupakan variabel-variabel yang menyimpan nilai dari struktur cJSON. Variabel apa yang diisi dan apa isinya tergantung dari tipe data yang disimpan.

## Fungsi-Fungsi Dasar

- Tipe-tipe data dasar  
Untuk setiap tipe data cJSON, ada sebuah fungsi **cJSON\_Create....**. Semua fungsi tersebut akan mengalokasikan sebuah struktur cJSON yang nantinya dapat dihapus dengan **cJSON\_Delete**. Perlu diingat juga bahwa seluruh struktur yang dibuat harus dihapus agar tidak terjadi kebocoran memori. Adapun fungsi-fungsi **cJSON\_Create** yang dapat digunakan untuk tipe-tipe data yang tersedia adalah sebagai berikut.
  - *null* dibuat dengan **cJSON\_CreateNull**.
  - *boolean* dibuat dengan **cJSON\_CreateBool**, atau jika ingin membuat data *boolean* langsung dengan nilainya, dapat menggunakan **cJSON\_CreateTrue** atau **cJSON\_CreateFalse**.
  - Bilangan apapun dibuat dengan **cJSON\_CreateNumber**. Penggunaan fungsi ini akan langsung mengisi nilai variabel **valueint** dan **valuedouble**.
  - *string* apa pun dibuat dengan **cJSON\_CreateString** (membuat sebuah string langsung sebagai nilai data JSON), atau **cJSON\_CreateStringRef** (mereferensikan *string* yang sudah ada sebagai nilai data JSON).
- *Array*  
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, di mana 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*.  
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.

Untuk menimpa/mengganti nilai suatu objek di dalam *array*, fungsi `cJSON_ReplaceItemInArray` dapat dipanggil dengan indeks dari objek yang ingin diganti sebagai parameternya. Selain fungsi tersebut, fungsi `cJSON_ReplaceItemViaPointer` juga dapat digunakan—fungsi ini bekerja dengan memutuskan (*detach*) objek lama yang ingin diganti, menghapus objek tersebut, dan menyisipkan objek yang baru ke tempat objek lama yang sudah dihapus tadi.

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

- Objek

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

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

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

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

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

- *Parsing*

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

```
cJSON_Parse(<JSON>)
```

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

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

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

### 2.3.4 CMake [11]

CMake merupakan sebuah perkakas generator *build system* yang memungkinkan pengembang 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 di luar 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.

Kode 2.6: Kode utama operasional CMake

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

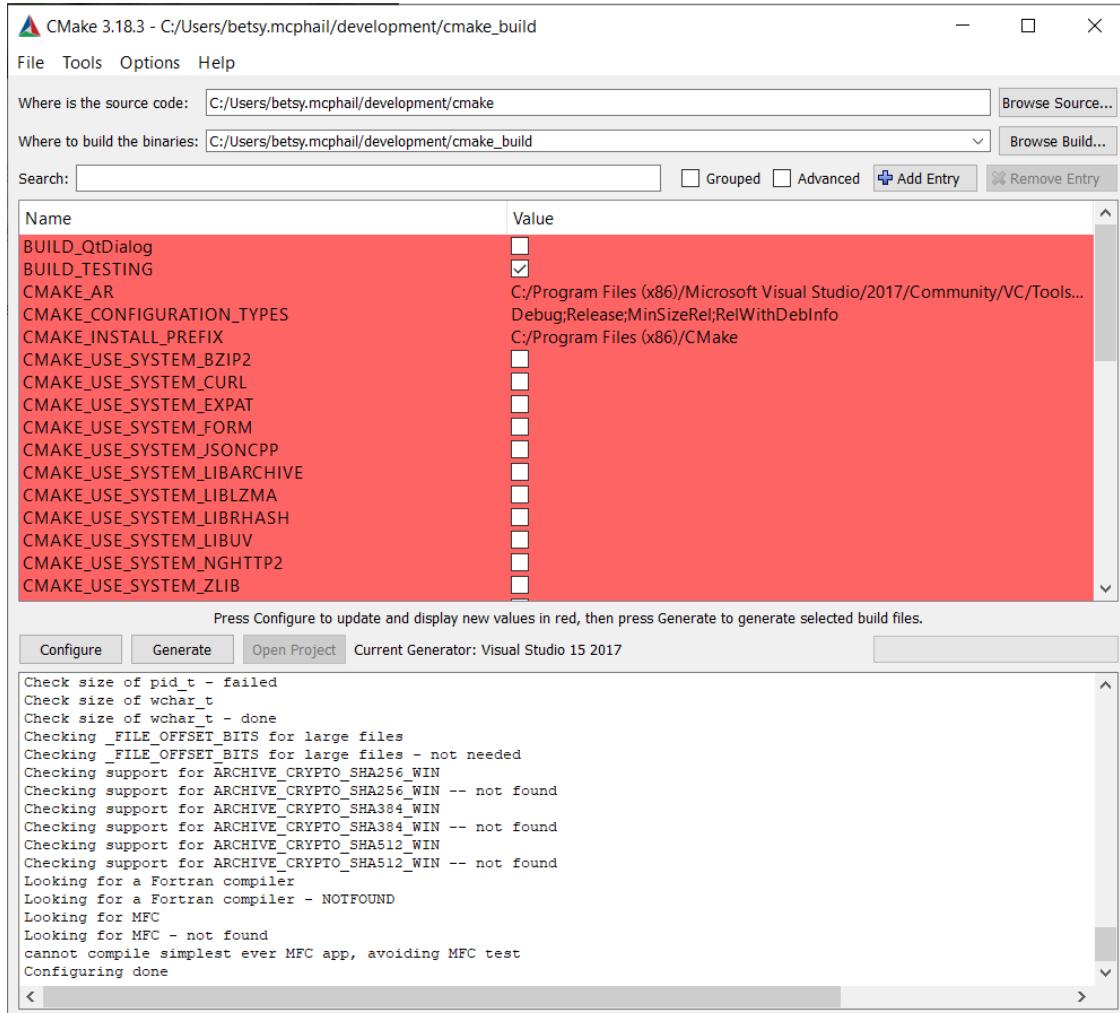
Penjelasan dari baris-baris berikut adalah sebagai berikut.

- 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, *add\_executable* merupakan sebuah perintah yang akan menambahkan sebuah file *executable* untuk menjalankan proyek yang sudah dibangun tersebut.
2. CMake membuat dan mengkonfigurasikan 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 2.12.



Gambar 2.12: Tampilan aplikasi `cmake-gui`<sup>10</sup>.

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

- `ccmake`

Di mayoritas sistem operasi berbasis UNIX, jika *library curses* didukung, maka CMake memiliki sebuah perangkat lunak lain yang dapat digunakan, yaitu `ccmake`. Untuk menjalankan `ccmake`, pengguna dapat menjalankannya melalui *command line*, dan direktori tempat `ccmake` dijalankan ini harus merupakan direktori tempat file-file *binary* nantinya ingin disimpan. Ketika aplikasi ini dijalankan, akan keluar tampilan seperti di Gambar 2.13. Adapun instruksi singkat penggunaan dari aplikasi ini dapat dilihat di bagian bawah dari tampilan tersebut.

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

The screenshot shows the CMake configuration interface. At the top, it says "Page 1 of 3". Below that is a list of CMake variables and their values:

```

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_LIBLZMA  *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

```

At the bottom of the interface, there is a message bar with the following text:

```

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 2.13: Tampilan aplikasi ccmake<sup>11</sup>.

- Langsung dari *command line*  
CMake juga dapat dijalankan melalui *command line*. Untuk menjalankan CMake dari *command line*, direktori tempat terminal berada lagi-lagi harus diatur ke direktori tempat file-file *binary* akan disimpan. Kemudian jalankan perintah `cmake` dengan opsi-opsi yang diinginkan, diikuti dengan direktori tempat *source code* dari perangkat lunak yang ingin dibangun berada. Walaupun begitu, perlu diingat bahwa metode ini direkomendasikan untuk digunakan hanya untuk proyek-proyek yang memiliki sedikit, atau bahkan tidak ada opsi sama sekali.
- 3. Pengguna membangun proyek tersebut dengan perkakas pembangunan masing-masing.  
CMake dapat memberikan beberapa opsi dalam pembangunan perangkat lunak yang dapat diatur oleh penggunanya masing-masing. Adapun dua opsi utama dari seluruh opsi-opsi tersebut adalah sebagai berikut.
  - Menspesifikasikan *compiler* yang akan digunakan  
Di beberapa sistem, bisa jadi terdapat lebih dari satu *compiler*, atau *compiler* yang ada tidak berada di tempat *compiler* tersebut biasa diinstal. Di kasus-kasus ini, CMake perlu diberitahu secara manual dimana letak *compiler* yang ingin digunakan. Ada tiga cara 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*.

<sup>11</sup><https://cmake.org/cmake/help/book/mastering-cmake/chapter/Getting%20Started.html>

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 menggunakan `ccmake`, maka perintah untuk menjalankan CMake harus ditulis seperti dapat dilihat di Kode 2.7.

Kode 2.7: Contoh kode pembangunan CMake lebih dari satu mode

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

Setelah CMake dijalankan, proyek tersebut siap untuk dibangun. Jika generator yang dipilih merupakan generator berbasis Makefile, 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.



## BAB 3

# ANALISIS

### 3.1 Analisis Aplikasi Sejenis

Untuk pembuatan perangkat lunak dalam skripsi ini, ada empat buah perkakas *command line* yang akan diamati sebagai aplikasi sejenis. Dua dari empat aplikasi pertama adalah *Chrome Web Store Item Property CLI* dan *iTunes Search API*. Selain dua perkakas tersebut, ada dua perkakas lainnya yang bisa digunakan sebagai referensi, tetapi tidak dapat dieksplorasi, karena kedua aplikasi tersebut tidak berhasil dijalankan dengan sempurna, yaitu *Uber CLI* dan *Google Maps Direction CLI*. Keempat perkakas ini akan dilihat dan diteliti untuk analisis bagaimana format serta karakteristik opsi-opsi dari perkakas *command line* pada umumnya.

#### 3.1.1 Chrome Web Store Item Property CLI<sup>1</sup>

Perkakas pertama yang akan diteliti adalah *Chrome Web Store Item Property CLI* yang dibuat oleh pengguna GitHub dengan nama “pandawing”. Perkakas ini merupakan ekstensi dari sebuah aplikasi lain (dari pembuat yang sama) yang memiliki fungsi yang sama, yaitu *Chrome Web Store Item Property*<sup>2</sup>. Artinya, perkakas ini memiliki fungsi yang sama dengan aplikasi dasarnya, yaitu memanggil fungsi API untuk mendapatkan metadata dari sebuah ekstensi pada *web store* peramban Google Chrome. Perbedaan dari perkakas ini dengan aplikasi dasarnya adalah bahwa perkakas ini dapat digunakan sebagai perkakas *command line*, sedangkan aplikasi dasarnya hanya bisa digunakan dalam perangkat lunak lainnya sebagai pemanggil fungsi API.

#### Penggunaan

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

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

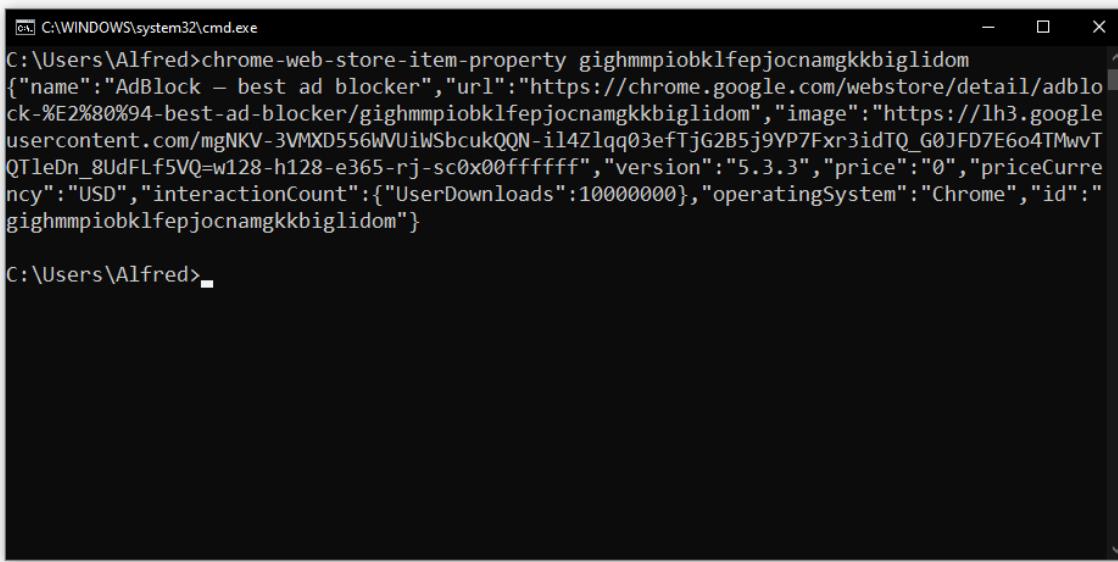
Dengan *identifier* berupa ID dari ekstensi yang diinginkan. Jadi, misalkan pengguna memasukkan `gighmmpioblkfepjocnamgkkbiglidom` sebagai ID yang akan digunakan sebagai *identifier*, maka perkakas ini akan mengembalikan metadata dari ekstensi “AdBlock” sebagai keluarannya. Contoh penggunaan perkakas ini dapat dilihat di Gambar 3.1.

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

- **name**  
Nama dari ekstensi yang dicari *metadata*-nya.
- **url**  
URL halaman web dari ekstensi yang dicari di *web store* Google Chrome.
- **image**  
Logo (dan ikon *thumbnail*) dari ekstensi yang dicari *metadata*-nya.

<sup>1</sup><https://github.com/pandawing/node-chrome-web-store-item-property-cli>, versi 2.0.1

<sup>2</sup><https://github.com/pandawing/node-chrome-web-store-item-property>, versi 1.2.0



```
C:\Users\Alfred>chrome-web-store-item-property gighmmpioblkfepjocnamgkkbiglidom
{"name": "AdBlock – best ad blocker", "url": "https://chrome.google.com/webstore/detail/adblock-%E2%80%94-best-ad-blocker/gighmmpioblkfepjocnamgkkbiglidom", "image": "https://lh3.googleusercontent.com/mgNKV-3VMXD556WVUiWSbcukQON-i14Zlqq03efTjG2B5j9YP7Fxr3idTQ_G0JFD7E6o4TMwvTQTleDn_8UdFLf5VQ=w128-h128-e365-rj-sc0x00fffff", "version": "5.3.3", "price": "0", "priceCurrency": "USD", "interactionCount": {"UserDownloads": 10000000}, "operatingSystem": "Chrome", "id": "gighmmpioblkfepjocnamgkkbiglidom"}
```

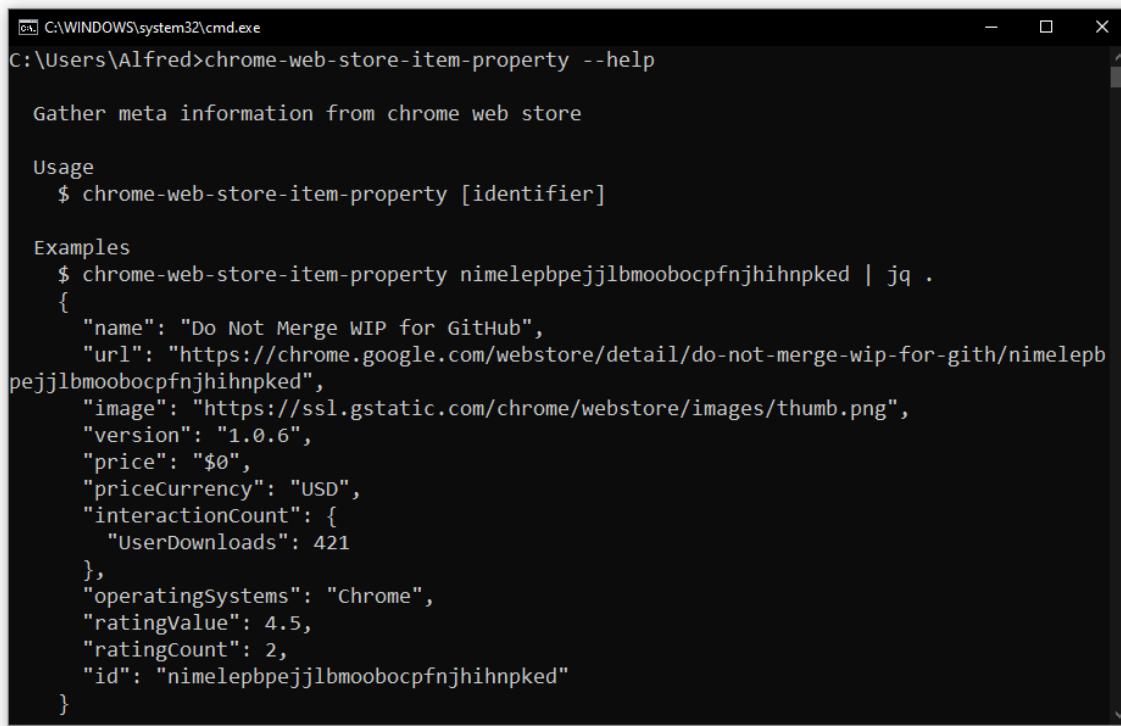
Gambar 3.1: Contoh penggunaan perkakas Chrome *Web Store Item Property* CLI untuk mencari ekstensi “AdBlock”.

- **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 mana pun.
- **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.

Selain penggunaan normalnya, pengguna perkakas ini juga dapat meminta perkakas untuk menampilkan halaman bantuannya, yaitu dengan perintah sebagai berikut.

```
chrome-web-store-item-property --help
```

Hasil dari eksekusi perintah ini dapat dilihat di Gambar 3.2.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Alfred>chrome-web-store-item-property --help

Gather meta information from chrome web store

Usage
$ chrome-web-store-item-property [identifier]

Examples
$ chrome-web-store-item-property nimelepbpejjlbmoobocpfnjhihnpked | jq .

{
  "name": "Do Not Merge WIP for GitHub",
  "url": "https://chrome.google.com/webstore/detail/do-not-merge-wip-for-gith/nimelepbpejjlbmoobocpfnjhihnpked",
  "image": "https://ssl.gstatic.com/chrome/webstore/images/thumb.png",
  "version": "1.0.6",
  "price": "$0",
  "priceCurrency": "USD",
  "interactionCount": {
    "UserDownloads": 421
  },
  "operatingSystems": "Chrome",
  "ratingValue": 4.5,
  "ratingCount": 2,
  "id": "nimelepbpejjlbmoobocpfnjhihnpked"
}
```

Gambar 3.2: Opsi `--help` pada perkakas Chrome *Web Store Item Property* CLI.

### 3.1.2 *iTunes Search API*<sup>3</sup>

Perkakas kedua yang akan diteliti adalah *iTunes Search API* oleh pengguna GitHub dengan nama “awcross”. Perkakas ini berfungsi untuk melakukan pencarian dengan mengutilisasikan API iTunes, sehingga seakan-akan pengguna langsung melakukan pencarian di iTunes sendiri. Hasil pencarian yang dilakukan termasuk judul lagu, nama artis, ataupun nama album, dan pengguna dapat memilih secara spesifik objek apa yang ingin dicari.

#### Penggunaan

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

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

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

- **--help**

Opsi ini merupakan opsi untuk menampilkan bantuan penggunaan perkakas, di mana ketika pengguna menjalankan perkakas dengan opsi ini, maka perkakas akan mengeluarkan format penggunaan perkakas, seluruh daftar opsi yang dapat digunakan, serta satu contoh penggunaannya. Adapun penggunaan serta keluaran dari opsi ini dapat dilihat di Gambar 3.3. Opsi ini juga tidak memiliki argumen apapun yang (**<options>** tidak perlu diikutkan).

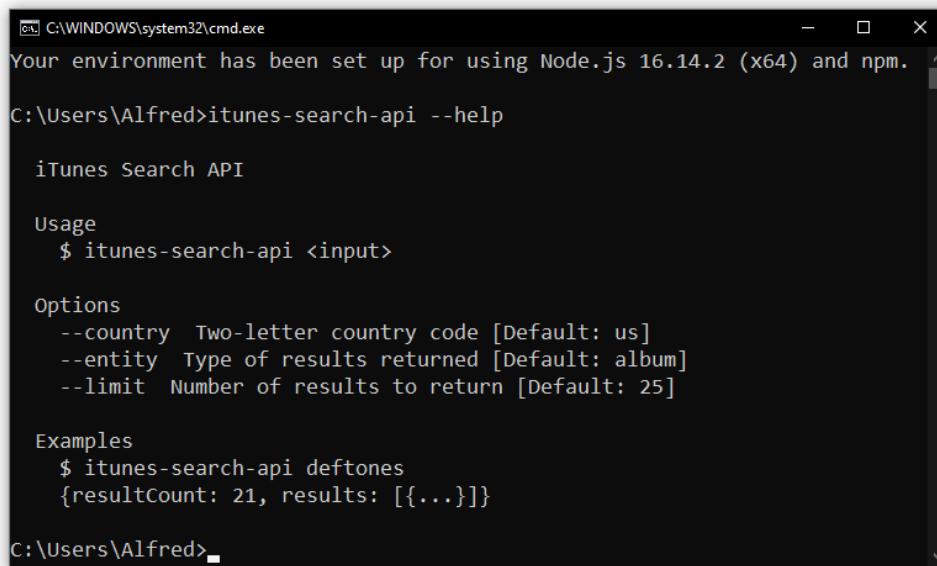
- **--country**

**Kemungkinan nilai:** Kode negara dua huruf

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

---

<sup>3</sup><https://github.com/awcross/itunes-search-api>, versi 1.0.1



```
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 --help

iTunes Search API

Usage
  $ itunes-search-api <input>

Options
  --country Two-letter country code [Default: us]
  --entity Type of results returned [Default: album]
  --limit Number of results to return [Default: 25]

Examples
  $ itunes-search-api deftones
  {resultCount: 21, results: [{...}]}

C:\Users\Alfred>
```

Gambar 3.3: Opsi --help pada perkakas *iTunes Search API*.

- **--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 apa pun yang memiliki kemiripan dengan input dalam salah satu dari ketiga properti ini akan muncul dalam hasil pencarian.

- **--limit**

**Kemungkinan nilai:** Bilangan bulat positif<sup>4</sup>

Batas maksimal hasil pencarian yang ingin ditampilkan dalam keluaran.

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

- **resultCount**

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

- **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.

Adapun contoh penggunaan dan hasil keluaran perkakas ini dapat dilihat di Gambar 3.4.

### 3.1.3 Uber CLI<sup>5</sup>

Uber CLI merupakan sebuah perkakas *command line* yang dibuat oleh pengguna GitHub dengan nama “jaebradley”. Perkakas ini yang dapat digunakan untuk dua fungsi utama. Fungsi pertama dari perkakas ini adalah untuk mendapatkan estimasi untuk seberapa lama waktu yang diperlukan untuk servis taksi *online* dari Uber untuk mencapai lokasi yang ingin dituju, sedangkan fungsi keduanya adalah untuk mengestimasi harga yang harus dibayarkan untuk memakai servis tersebut.

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

<sup>5</sup><https://github.com/jaebradley/uber-cli>, versi 1.0.2

```
C:\WINDOWS\system32\cmd.exe
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', v
    }
  ]
}
```

Gambar 3.4: Contoh penggunaan perkakas *iTunes Search API* untuk mencari artis bernama “*The Presets*”.

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

```
uber time <alamat>
```

`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 3.5. Keluaran yang dihasilkan oleh perkakas ini akan meliputi seluruh jenis servis yang disediakan oleh Uber, dan menunjukkan estimasi waktu tempuh perjalanan untuk tiap-tiap servisnya.

Waktu	Servis
2 min.	uberPOOL, uberX
3 min.	uberXL
5 min.	UberBLACK, uberSUV
7 min.	TAXI

Gambar 3.5: Contoh penggunaan fitur prediksi waktu perjalanan untuk perkakas Uber CLI.<sup>6</sup>

<sup>6</sup><https://github.com/jaebradley/uber-cli>

Hal lain yang perlu diperhatikan bahwa di keluaran perkakas ini ada beberapa gambar grafis. Gambar-gambar ini disebut *emoji unicode*, yang merupakan karakter-karakter *unicode* berupa simbol piktorial (piktograf). Karena pada dasarnya *emoji-emoji unicode* ini merupakan karakter *unicode*, mereka didukung oleh CLI apapun yang mendukung karakter *unicode*<sup>7</sup>.

Untuk memanggil fungsi kedua dari perkakas ini, pengguna dapat dilakukan dengan memanggil perintah dengan format berikut.

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

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

Keluaran dari fungsi kedua ini dapat dilihat di Ggambar 3.6. Sama seperti keluaran untuk fungsi pertamanya, keluaran untuk fungsi kedua perkakas ini juga meliputi seluruh jasa yang disediakan oleh Uber.

21:00 \$ uber price -s '25 first street cambridge ma' -e '114 line street somerville ma'				
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 3.6: Contoh penggunaan fitur prediksi harga perjalanan untuk perkakas Uber CLI<sup>8</sup>.

## Permasalahan

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

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

<sup>7</sup><https://www.unicode.org/emoji/techindex.html>

<sup>8</sup>Gambar diambil dari <https://github.com/jaebradley/uber-cli>

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

### 3.1.4 Google Maps Direction CLI<sup>10</sup>

*Google Maps Direction CLI* merupakan sebuah perkakas *command line* yang dibuat oleh pengguna GitHub dengan nama “yujinlim”. Perkakas ini memiliki kegunaan yang mirip dengan KIRI, hanya saja perkakas ini tidak secara spesifik mengharuskan penggunaan angkot, atau transportasi umum lainnya. Singkatnya, perkakas ini memiliki fungsi seperti sebuah GPS. Untuk menggunakannya, pengguna harus memasukkan perintah sebagai berikut.

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

Setelah pengguna memasukkan perintah tersebut dengan benar, perkakas ini akan mengirim permintaan ke API *Google Maps*, di mana jika prosesnya berhasil, keluarannya akan berupa langkah-langkah yang harus ditempuh untuk sampai ke lokasi akhir, beserta di jarak berapa langkah tersebut harus diambil, relatif terhadap langkah sebelumnya. Perintah untuk perkakas ini juga dapat mengikutkan kunci API *Google Maps* melalui opsi `-k` (atau `--key`) untuk ramalan kemacetan yang lebih akurat. Adapun penggunaan dari perkakas ini dapat dilihat di gambar 3.7.

```

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 Beringin Go 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 Penyuraian 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/Jln. 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 Dato 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/E12 Toll road (0.3 km)
Take the exit toward KLCC Partial toll road (0.6 km)

```

Gambar 3.7: Contoh penggunaan perkakas *Google Maps Direction CLI*<sup>11</sup>.

<sup>9</sup><https://github.com/jaebradley/uber-cli/issues/87>

<sup>10</sup><https://github.com/yujinlim/google-maps-direction-cli>, versi 1.1.0

## Permasalahan

Seperti tertulis di awal bab ini, perkakas ini juga tidak bisa digunakan. Alasan perkakas ini tidak dapat digunakan lagi-lagi merupakan masalah teknis, yaitu diperbaruiannya API Google *Maps*. Lebih spesifiknya, semenjak 2018, Google tidak lagi memperbolehkan penggunaan API Google *Maps* tanpa kunci API, yang sayangnya mendasari pemakaian perkakas ini. Selain itu, ada ketentuan tambahan dari Google bahwa bahwa kunci API ini tidak lagi bisa didapatkan tanpa membayarkan sebuah biaya tertentu. Oleh karena itu, perkakas ini dianggap tidak bisa dijalankan.

## 3.2 Analisis API KIRI

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

### 3.2.1 *Search Place*

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

- **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.

- **mode**

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

- **region**

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

- **querystring**

Parameter ini akan berisi masukan kata kunci untuk mencari lokasi yang ingin ditemukan.

- **apikey**

Parameter ini akan berisi sebuah nilai kunci API yang sudah dibuat sebelumnya, seperti dijelaskan di subbab 2.1.2, parameter ini akan bernilai 68C...97C<sup>12</sup>.

### 3.2.2 *Routing*

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

- **version**

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

- **mode**

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

---

<sup>11</sup>Gambar diambil dari <https://github.com/yujinlim/google-maps-direction-cli>

<sup>12</sup>Bagian tengah dari kunci API diganti dengan “...” untuk alasan keamanan.

- **locale**

Isi dari parameter ini akan ditentukan pada saat pemakaian perkakas.

- **start**

Parameter ini merupakan nilai *latitude* dan *longituded*ari titik awal perjalanan pengguna, yang akan diberikan sebagai masukan pada saat pemakaian perkakas.

- **finish**

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

- **presentation**

Parameter ini hanya digunakan untuk fitur *backwards compatibility*. Jika parameter ini ingin digunakan, maka isi dari parameter ini hanya ada satu kemungkinan, yaitu **desktop**. Parameter ini ada karena versi lama dari halaman *web* KIRI memerlukan tampilan yang berbeda untuk perangkat-perangkat *desktop* dan perangkat-perangkat *mobile*. Walaupun begitu, sekarang parameter ini tidak lagi berpengaruh ke keluaran API.

- **apikey**

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

Layanan ini dapat digunakan sebagai lanjutan dari layanan *search place*, terutama karena fitur pencarian rute hanya menerima nilai *latitude* dan *longituded*ari lokasi-lokasi yang akan digunakan sebagai masukan, yang tidak hanya akan menyusahkan pengguna dalam memakai perkakas *command line* ini, tetapi juga kedua nilai tersebut juga merupakan salah satu dari keluaran layanan pencarian lokasi. Jadi, dengan menggunakan kedua layanan tersebut secara berurutan, maka pengguna akan bisa mencari lokasi awal dan akhir yang diinginkan hanya dengan kata-kata kunci, dan selanjutnya pengguna akan dapat menerima langkah-langkah yang harus ditempuh dalam rute sebagai keluaran akhir dari perkakas.

### 3.3 Analisis Perkakas yang Dibuat

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

#### 3.3.1 Analisis Fitur Perkakas

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

Walaupun begitu, aplikasi ini akan memiliki tiga fitur utama yang berhubungan satu sama lain, yaitu pencarian lokasi dengan kata kunci pencarian, pencarian rute dengan koordinat *latitude* dan *longitude* lokasi awal dan akhir, dan pencarian rute dengan kata kunci pencarian lokasi awal dan akhir. Selain ketiga fitur utama ini, perkakas juga akan memiliki fitur untuk menampilkan halaman bantuan penggunaannya, seperti layaknya perkakas *command line* pada umumnya.

- Pencarian lokasi dengan kata kunci pencarian

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

- Pencarian rute angkot dengan koordinat *latitude* dan *longitude* lokasi awal dan akhir  
Pencarian rute angkot dengan koordinat merupakan fungsi kedua dari perkakas ini. Dalam fungsi ini, perkakas akan meminta masukan dari pengguna berupa nilai *latitude* dan *longitude* dari lokasi awal serta lokasi akhir dari perjalanan pengguna. Setelah masukan ini berhasil diterima dan diproses, perkakas akan mengeluarkan keluaran akhir berupa langkah-langkah yang harus diambil oleh pengguna untuk pergi dari lokasi awal ke lokasi akhir, dengan memanfaatkan jasa angkot yang ada.
- Pencarian rute angkot dengan kata kunci pencarian lokasi awal dan akhir  
Pencarian rute angkot dengan kata kunci pencarian merupakan fungsi ketiga dari perkakas ini. Dalam fungsi ini, perkakas akan meminta masukan dari pengguna berupa kata kunci untuk pencarian lokasi awal dan lokasi akhir dari perjalanan yang ingin dilakukan. Setelah masukan ini berhasil diterima dan diproses, perkakas akan langsung mengeluarkan keluaran akhir berupa langkah-langkah yang harus diambil oleh pengguna untuk pergi dari lokasi awal ke lokasi akhir, dengan memanfaatkan jasa angkot yang ada.

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

## Aturan Penamaan

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

1. Opsi-opsi perkakas yang akan dibuat akan dinamakan berdasarkan penamaan opsi-opsi dalam perkakas-perkakas yang telah dianalisis. Hal ini juga berarti bahwa nama-nama opsi perkakas yang akan dibuat kemungkinan besar tidak akan sama dengan nama opsi-opsi perkakas-perkakas yang dianalisis.
2. Nama pendek (satu huruf) opsi umumnya akan dinamakan sesuai dengan huruf pertama dari fitur/variabel yang direpresentasikan oleh opsi tersebut.
3. Nama panjang dari opsi umumnya akan dinamakan sesuai dengan nama fitur/variabel yang direpresentasikan oleh opsi tersebut.
4. Argumen dari opsi yang merepresentasikan fitur perkakas akan dinamakan sesuai dengan nama dari mode operasionalnya masing-masing di API KIRI, kecuali `direct`, karena fitur ini merupakan mode tambahan yang bukan merupakan fitur langsung dari API KIRI.
5. Nama-nama opsi dan, baik pendek maupun panjang, yang jika mengikuti ketiga aturan pertama akan menghasilkan nama yang sama dengan nama opsi lain, akan dinamakan dengan nama buatan yang artinya tidak terlalu jauh dengan variabel apa yang direpresentasikan olehnya.

## Opsi-opsi Perkakas

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

- `-h` (`--help`)

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

Mengikuti nama opsi-opsi bantuan dari perkakas-perkakas yang telah dianalisis, maka opsi ini akan dinamakan `--help`, atau `-h` untuk versi pendeknya.

- `-m <mode> (--mode)`

Opsi ini merupakan opsi berparameter yang menentukan fitur mana yang ingin digunakan oleh pengguna. Nama dari opsi ini ditentukan sebagai `-m` atau `--mode` karena opsi ini merupakan opsi yang mengatur mode penggunaan perkakas. Adapun isi dari parameter `<mode>` yang dapat digunakan adalah sebagai berikut:

- `searchplace`

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

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

Opsi ini merupakan opsi yang akan menerima parameter berupa kode area (region) dari lokasi yang ingin dicari.

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

Opsi ini merupakan opsi yang akan menerima sebuah *string* sebagai parameternya. *String* ini akan digunakan oleh perkakas sebagai kata kunci untuk pencarian lokasi yang ingin ditemukan oleh pengguna. Opsi ini diberi nama demikian karena nama parameter ini di API KIRI adalah “*querystring*”.

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

Opsi ini akan menerima parameter yang mengatur bahasa (*locale*) yang akan digunakan oleh perkakas di keluarannya nanti. Opsi ini merupakan tambahan spesifik dari perkakas ini, karena keluaran dari perkakas mengandung kata-kata yang bukan merupakan nama/koordinat lokasi hasil. Walaupun begitu, opsi ini tidak akan mempengaruhi nama lokasi dalam keluaran.

- `findroute`

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

- \* `-s <lokasi awal> (--start)`

Opsi ini merupakan opsi yang akan menerima parameter berupa lokasi awal (*start*) perjalanan pengguna nantinya. Opsi ini hanya menerima masukan lokasi berupa nilai *latitude* dan *longitude* dari lokasi tersebut.

- \* `-f <lokasi akhir> (--finish)`

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

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

Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh perkakas di keluarannya nanti. Opsi ini adalah opsi yang sama dengan opsi yang digunakan dalam mode `searchplace`, dan mengikuti aturan penamaan yang sama.

- `direct`

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

1. Perkakas akan menerima lokasi awal dan akhir berupa kata kunci dari pengguna yang dimasukkan ke dalam parameter dari opsi masing-masing.
2. Pencarian lokasi akan dilakukan dua kali—untuk lokasi awal dan lokasi akhir, dan hasilnya akan langsung ditampilkan ke pengguna. Perlu ditekankan bahwa:

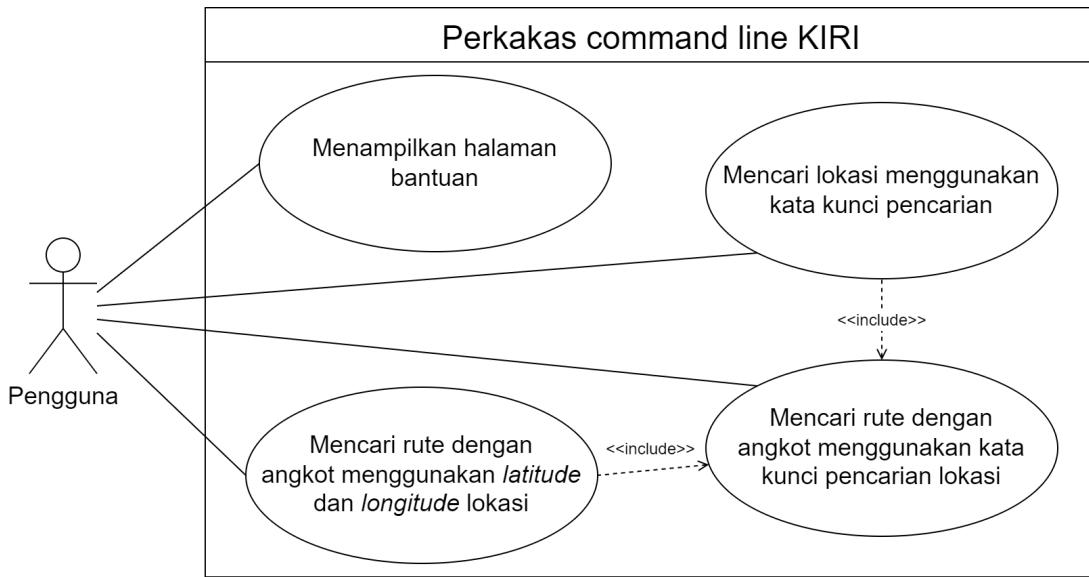
- \* Perkakas hanya akan mengambil hasil pertama saja di tiap pencarinya, jadi jika terdapat banyak hasil dari salah satu proses pencarian, hanya lokasi pertama saja yang akan digunakan untuk proses selanjutnya.
  - \* Perkakas tidak peduli apakah hasil pencarian tersebut benar merupakan hasil yang diinginkan pengguna atau tidak. Jika satu atau lebih lokasi ternyata tidak sesuai, pengguna dapat melakukan ulang seluruh prosesnya.
3. Koordinat *latitude* dan *longitude* dari kedua lokasi tersebut kemudian akan digunakan sebagai masukan untuk langkah kedua dari fitur ini, yaitu pencarian rute.
  4. Hasil dari pencarian rute akan ditampilkan sebagai keluaran akhir dari fitur ini. Secara esensi, fitur ini merupakan fitur yang didesain untuk bekerja menyerupai fitur *smart direction*, yang merupakan salah satu fitur dari KIRI sendiri, tetapi bukan merupakan fitur yang dapat digunakan melalui API KIRI. Adapun parameter yang diperlukan untuk fitur ini merupakan gabungan dari opsi-opsi kedua fitur sebelumnya, yaitu:
    - \* **-S <region tempat lokasi awal berada> (--restart)**  
Opsi ini akan menerima parameter berupa kode region dari lokasi awal. Opsi ini tidak dapat diberi huruf **-r** seperti pada mode **searchplace**, karena pemberian nama tersebut akan membuat konflik dengan opsi kode region lokasi akhir. Oleh karena itu, opsi ini diberikan huruf **-S** (s kapital), dengan nama panjang **--restart**, yang berarti “*region start*”.
    - \* **-F <region tempat lokasi akhir berada> (--regfinish)**  
Opsi ini akan menerima parameter berupa kode region dari lokasi akhir. Opsi ini tidak dapat diberi huruf **-r** seperti pada mode **searchplace**, karena pemberian nama tersebut akan membuat konflik dengan opsi kode region lokasi awal. Oleh karena itu, opsi ini diberikan huruf **-F** (f kapital), dengan nama panjang **--regfinish**, yang berarti “*region finish*”.
    - \* **-s <kata kunci untuk lokasi awal> (--start)**  
Opsi ini akan menerima parameter berupa kata kunci untuk pencarian lokasi awal yang akan digunakan. Penamaan opsi ini sama seperti opsi ekuivalennya di mode **findroute**.
    - \* **-f <kata kunci untuk lokasi akhir> (--finish)**  
Opsi ini akan menerima parameter berupa kata kunci untuk pencarian lokasi akhir yang akan digunakan. Penamaan opsi ini sama seperti opsi ekuivalennya di mode **findroute**.
    - \* **-l <kode bahasa> (--locale)**  
Opsi ini akan menerima parameter yang mengatur bahasa yang akan digunakan oleh perkakas di keluarannya nanti. Opsi ini merupakan opsi yang sama dengan opsi yang digunakan dalam kedua mode lainnya, dan juga mengikuti aturan penamaan yang sama.

### 3.3.2 Analisis Kebutuhan Perkakas

Perkakas *command line* ini akan memiliki tiga fitur utama, 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 empat *use case*, yang dapat dilihat di Gambar 3.8.

Berikut merupakan empat tabel *scenario case*, yang masing-masing akan menjelaskan secara singkat keempat *use case* dari perkakas yang dibuat. Tabel-tabel tersebut adalah sebagai berikut.

- Tabel 3.1: *Scenario case* untuk fitur penampilan bantuan penggunaan perkakas.
- Tabel 3.2: *Scenario case* untuk fitur pencarian lokasi dengan kata kunci pencarian.
- Tabel 3.3: *Scenario case* untuk fitur pencarian rute dengan *latitude* dan *longitude* lokasi awal dan akhir.
- Tabel 3.4: *Scenario case* untuk fitur pencarian rute dengan kata kunci pencarian lokasi awal dan akhir.

Gambar 3.8: Diagram *use case* dari perkakas yang dibangun.Tabel 3.1: *Scenario case* untuk fitur halaman bantuan.

<b>Nama</b>	Menampilkan halaman bantuan
<b>Deskripsi</b>	Fitur untuk menampilkan bantuan penggunaan perkakas.
<b>Aktor</b>	Pengguna aplikasi
<b>Pre-kondisi</b>	-
<b>Pos-kondisi</b>	Halaman bantuan penggunaan.
<b>Skenario utama</b>	<ol style="list-style-type: none"> <li>Perkakas membaca masukan pengguna, yaitu perintah untuk menampilkan bantuan penggunaan perkakas.</li> <li>Perkakas menampilkan bantuan penggunaan perkakas kepada pengguna.</li> </ol>
<b>Skenario eksepsi</b>	Perintah yang dimasukkan pengguna tidak sesuai format/tidak valid.

Tabel 3.2: *Scenario case* untuk fitur pencarian lokasi dengan kata kunci pencarian.

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

**Tabel 3.2** — Lanjutan dari halaman sebelumnya

	4. Perkakas menampilkan kembali keluaran tersebut kepada pengguna.
<b>Skenario eksepsi</b>	<ul style="list-style-type: none"> <li>- Pengguna memasukkan masukan yang tidak valid sebagai kata kunci pencarian.</li> <li>- Lokasi tidak berhasil ditemukan.</li> </ul>

Tabel 3.3: *Scenario case* untuk fitur pencarian rute dengan angkot, dengan nilai *latitude* dan *longitude* kedua lokasi sebagai masukan.

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

Tabel 3.4: *Scenario case* untuk fitur pencarian rute dengan angkot, dengan kata kunci pencarian lokasi awal dan akhir sebagai masukan.

<b>Nama</b>	Mencari rute dengan angkot menggunakan kata kunci pencarian lokasi
<b>Deskripsi</b>	Fitur untuk mencari cara pergi ke satu lokasi ke lokasi lainnya dengan menggunakan angkot dengan kata kunci pencarian kedua lokasi sebagai masukan.
<b>Aktor</b>	Pengguna aplikasi
<b>Pre-kondisi</b>	-
<b>Pos-kondisi</b>	Langkah-langkah yang harus ditempuh untuk perjalanan tersebut akan ditampilkan kepada pengguna.
<b>Skenario utama</b>	<ol style="list-style-type: none"> <li>1. Perkakas membaca masukan berupa kata kunci pencarian lokasi awal dan akhir dari argumen dalam perintah <i>command line</i>.</li> </ol>
	Lanjut di halaman berikutnya...

**Tabel 3.4** — Lanjutan dari halaman sebelumnya

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



## BAB 4

# PERANCANGAN

Pada bab ini akan dipaparkan berbagai macam rancangan dari perkakas *command line* yang dibuat, seperti diagram-diagram terkait, masukan serta keluaran perangkat lunak, serta detail mengenai fungsi-fungsi yang ada di dalam perkakas tersebut.

### 4.1 Rancangan Alur Kerja Perkakas

Bagian ini akan membahas mengenai alur kerja perkakas yang dibuat, dalam bentuk *activity diagram* serta *sequence diagram* dari fitur-fitur perkakas. Perkakas memiliki empat buah fitur, yaitu:

- menampilkan bantuan penggunaan,
- mencari lokasi menggunakan kata kunci pencarian (**searchplace**),
- mencari rute dengan angkot menggunakan *latitude* dan *longitude* lokasi (**findroute**), dan
- mencari rute dengan angkot menggunakan kata kunci pencarian lokasi (**direct**).

Tiap-tiap fitur akan memiliki satu dari setiap tipe diagram. Kedua tipe diagram diikutkan untuk menggambarkan dengan lebih jelas langkah-langkah kerja perkakas, dengan *activity diagram* menggambarkan *apa saja* aktivitas yang dilakukan oleh tiap-tiap aktor dalam setiap proses kerja perkakas, dan *sequence diagram* menggambarkan *kapan* aktivitas-aktivitas tersebut dilakukan oleh tiap-tiap aktor. Adapun penjelasan dari bagaimana fitur-fitur ini akan diimplementasikan adalah sebagai berikut.

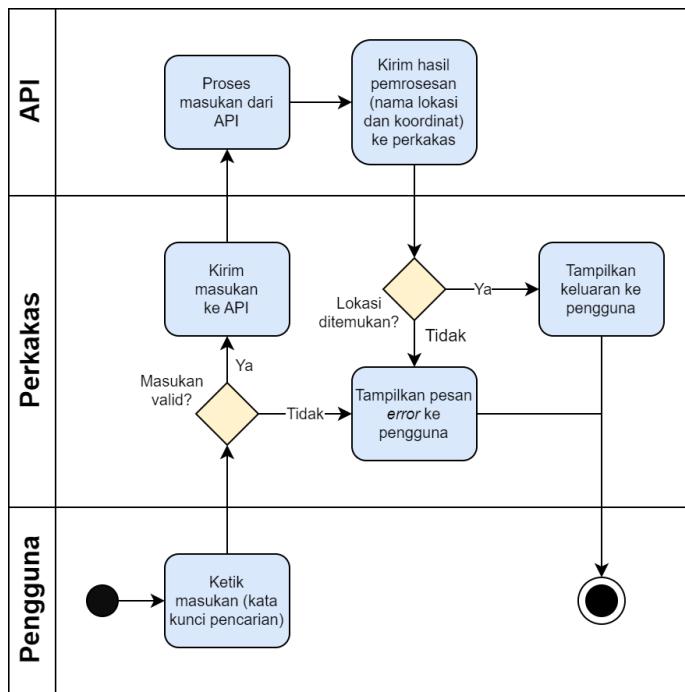
#### 4.1.1 Mencari lokasi menggunakan kata kunci pencarian

Alur kerja untuk fitur pencarian lokasi menggunakan kata kunci pencarian adalah sebagai berikut.

1. Perkakas akan meminta masukan dari pengguna berupa kata kunci dari lokasi yang ingin dicari.
2. Masukan akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan hanya akan dianggap tidak valid apabila pengguna tidak memasukkan apa pun sebagai masukan perkakas (masukan kosong).
3. Jika kata kunci masukan:
  - tidak valid, maka perkakas akan mengeluarkan pesan *error* dan keluar.
  - dianggap valid, kata kunci 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 lokasi:
  - ditemukan, maka nama dan koordinat *latitude* dan *longitude* lokasi akan ditampilkan ke pengguna sebagai keluaran akhir.
  - tidak ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.

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

Dengan alur tersebut, maka dapat dibuat sepasang diagram untuk fitur ini, di mana *activity diagram*-nya dapat dilihat di Gambar 4.1, dan *sequence diagram*-nya dapat dilihat di Gambar 4.2.



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

#### 4.1.2 Mencari rute dengan angkot menggunakan *latitude* dan *longitude* lokasi awal dan akhir

Alur kerja untuk fitur pencarian rute angkot menggunakan koordinat *latitude* dan *longitude* lokasi awal dan akhir adalah sebagai berikut.

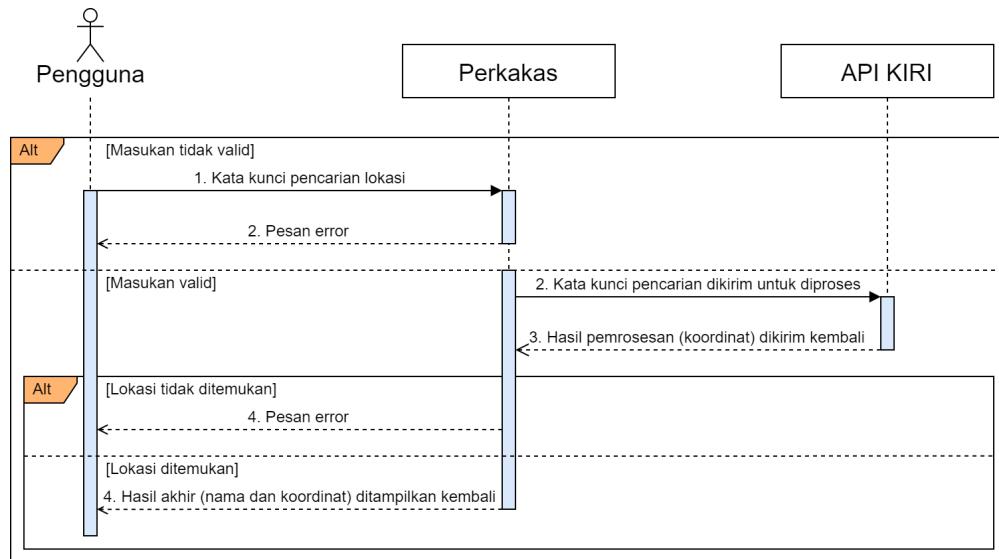
1. Perkakas akan meminta dua buah masukan dari pengguna, berupa koordinat *latitude* dan *longitude* dari lokasi awal (mulai) dan lokasi akhir (tujuan).
2. Masukan akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan akan dianggap tidak valid apabila ada masukan yang bukan berupa koordinat *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.

Dengan mengikuti alur kerja fitur yang telah dipaparkan, maka dapat dibuat sepasang diagram, yaitu *activity diagram* yang dapat dilihat di Gambar 4.3, dan *sequence diagram* yang dapat dilihat di Gambar 4.4.

#### 4.1.3 Mencari rute dengan angkot menggunakan kata kunci pencarian lokasi

Alur kerja dari fitur pencarian rute angkot menggunakan kata kunci pencarian lokasi adalah sebagai berikut.

1. Perkakas akan meminta masukan dari pengguna berupa kata kunci dari lokasi awal dan lokasi akhir yang ingin dicari.
2. Masukan untuk lokasi awal akan dicek oleh perkakas validitasnya. Dalam kasus ini, masukan hanya akan dianggap tidak valid apabila pengguna tidak memasukkan apa pun (masukan kosong), atau memasukkan koordinat *latitude* dan *longitude* sebagai masukan perkakas.



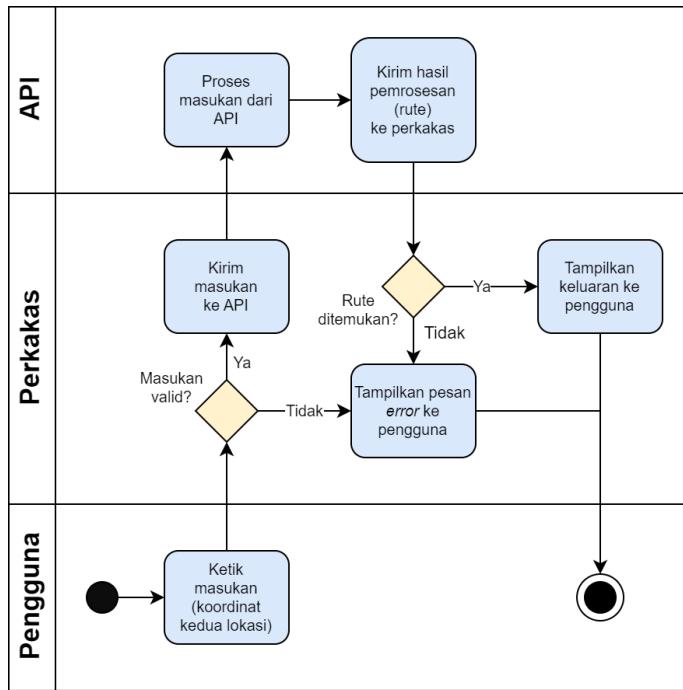
Gambar 4.2: *Sequence diagram* dari fitur pencarian lokasi menggunakan kata kunci pencarian.

3. Jika kata kunci masukan:
  - tidak valid, maka perkakas akan mengeluarkan pesan *error* dan keluar.
  - dianggap valid, kata kunci 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 lokasi:
  - tidak ditemukan, maka perkakas akan mengeluarkan pesan *error* dan keluar.
  - ditemukan, maka nama dan koordinat *latitude* dan *longitude* lokasi awal akan disimpan sebagai variabel untuk dipakai di langkah selanjutnya.
6. Langkah 2 sampai 5 akan diulang kembali untuk lokasi akhir.
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.
8. Setelah selesai diproses, keluaran berupa rute dari API akan dikembalikan ke perkakas.
9. 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.

Dengan alur kerja seperti yang telah dijelaskan, maka untuk fitur ini dapat dibuat sebuah *activity diagram*, yang dapat dilihat di Gambar 4.5, serta sebuah *sequence diagram*, yang dapat dilihat di Gambar 4.6.

## 4.2 Rancangan Implementasi Perkakas

Bagian ini akan membahas hal-hal terkait rancangan implementasi alur kerja perkakas, seperti cara kerja perkakas, kapan dan untuk apa *library-library* yang telah dibahas di Bab 2 akan digunakan, variabel-variabel utama yang diperlukan di dalam perkakas, serta hal-hal seputar fungsi-fungsi yang ada di dalam perkakas, seperti nama fungsinya, apa tujuan dari fungsi tersebut, masukan dan keluarannya, serta garis besar dari cara kerja fungsi tersebut.

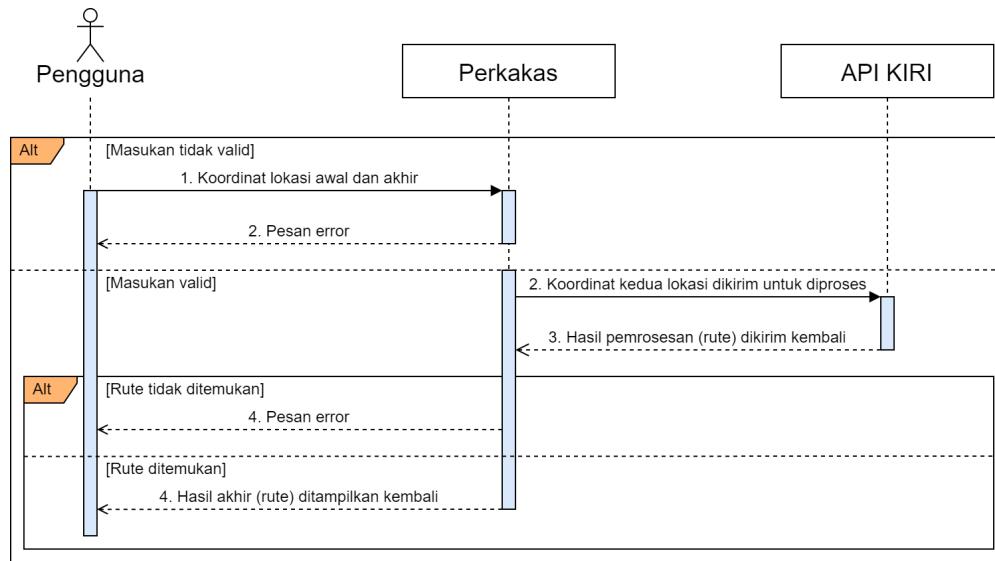


Gambar 4.3: *Activity diagram* dari fitur pencarian rute angkot menggunakan *latitude* dan *longitude* lokasi.

#### 4.2.1 Cara Kerja Perkakas

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

1. Penerimaan opsi dan argumen
  - Perkakas akan menggunakan fungsi `getopt_long` untuk menerima opsi-opsi serta argumen-argumennya dari masukan pengguna.
  - Pemeriksaan dasar akan dilakukan di bagian ini. Maksud dari “pemeriksaan dasar” adalah perkakas akan mengecek apakah masukan pengguna sesuai harapan atau tidak, seperti pengecekan sintaks perintah, jumlah argumen yang dimasukkan, serta validitas opsi dan argumen yang dimasukkan.
  - Kesalahan apapun selain kategori-kategori yang termasuk pengecekan dasar di atas hanya akan ditandai.
2. Verifikasi kebenaran opsi dan argumen
  - Perkakas akan melakukan pengecekan lanjutan terhadap argumen-argumen yang dimasukkan pengguna.
  - Perkakas akan berhenti dan mengeluarkan pesan *error* yang sesuai apabila ada opsi atau argumen yang tidak valid.
3. Pengiriman permintaan GET ke API serta penerimaan kembali respons
  - Perkakas akan membangun URL yang diperlukan untuk permintaan GET.
  - Permintaan GET ini akan dilakukan dengan bantuan *library* libcurl.
  - *Library* libcurl ini juga yang akan memfasilitasi penerimaan kembali respons dari API.
4. Verifikasi akhir respons API
  - Isi dari respons API akan diperiksa di langkah ini.
  - Respons dari API akan berupa objek JSON, dan *library* cJSON akan digunakan untuk mengkonversikan respons API ke bentuk yang dapat diproses langsung oleh perkakas.
  - Segala abnormalitas yang terdapat di dalam isi respons tersebut akan ditangani langsung oleh perkakas dalam bentuk pesan-pesan *error* untuk tiap kasusnya.
  - Jika tidak ada hal-hal yang tidak diinginkan di dalam responsnya, maka keluaran akan ditampilkan seperti biasanya oleh perkakas.



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

Perhatikan bahwa *library* CMake tidak disebut sama sekali di cara kerja. Hal ini dikarenakan *library* ini hanya dipakai untuk fungsi *cross-platform* dari perkakas, dan tidak dipakai sama sekali di seluruh proses kerja utamanya sendiri.

#### 4.2.2 Tipe Data Tambahan

Bagian ini akan mendetailkan sebuah tipe data tambahan yang dibuat khusus untuk pemakaian dalam perkakas ini. Tipe data buatan ini, yang merupakan salah satu dari variabel global yang ada di perkakas ini, adalah **chunk**, yang merupakan sebuah **struct** (struktur) yang mengandung dua variabel internal, yaitu sebagai berikut.

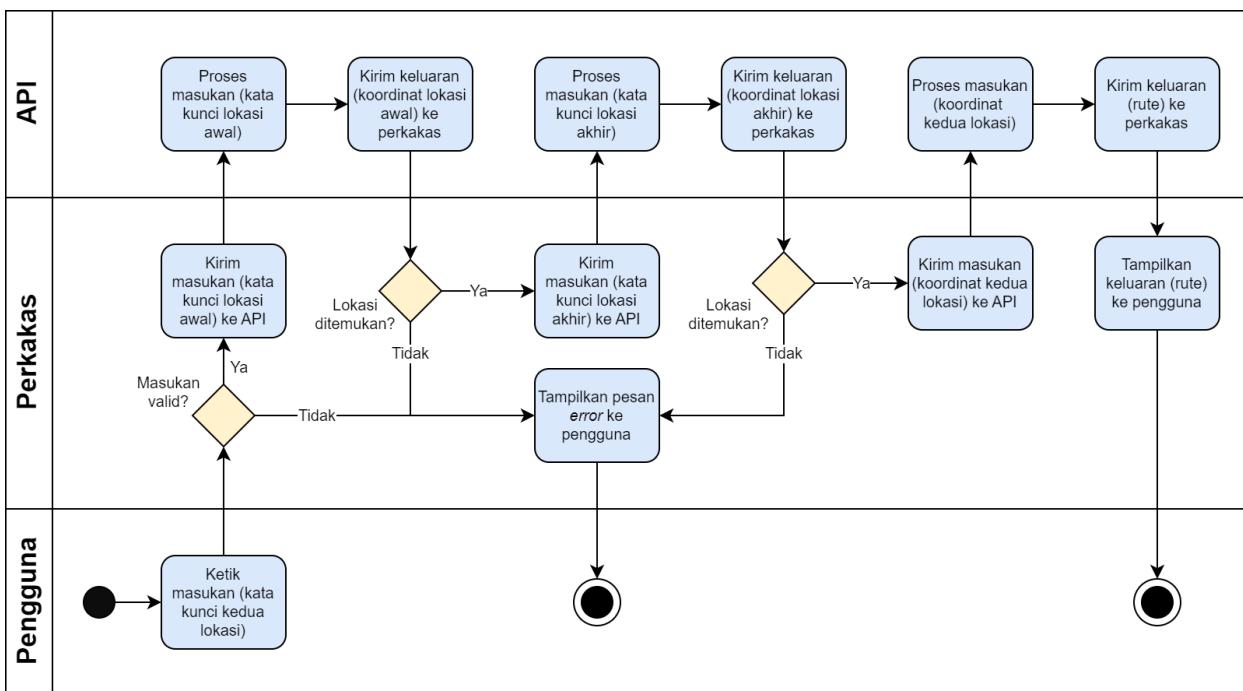
- **data** (`char *`)
- **size** (`size_t`)

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

#### 4.2.3 Variabel Global

Perkakas ini memiliki beberapa variabel global, yang sengaja diletakkan sebagai variabel global karena variabel-variabel ini digunakan di hampir keseluruhan dari keempat proses di subbab 4.2.1. Adapun variabel-variabel ini adalah sebagai berikut:

- **responsedata** (**chunk**)  
Variabel ini akan diisi oleh respons dari API KIRI. Seperti yang telah dipaparkan secara 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-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.



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

- **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)
- **2:** bdo (Bandoeng/Bandung)
- **3:** mlg (Malang)
- **4:** sub (Surabaya)

- **query (char[])**

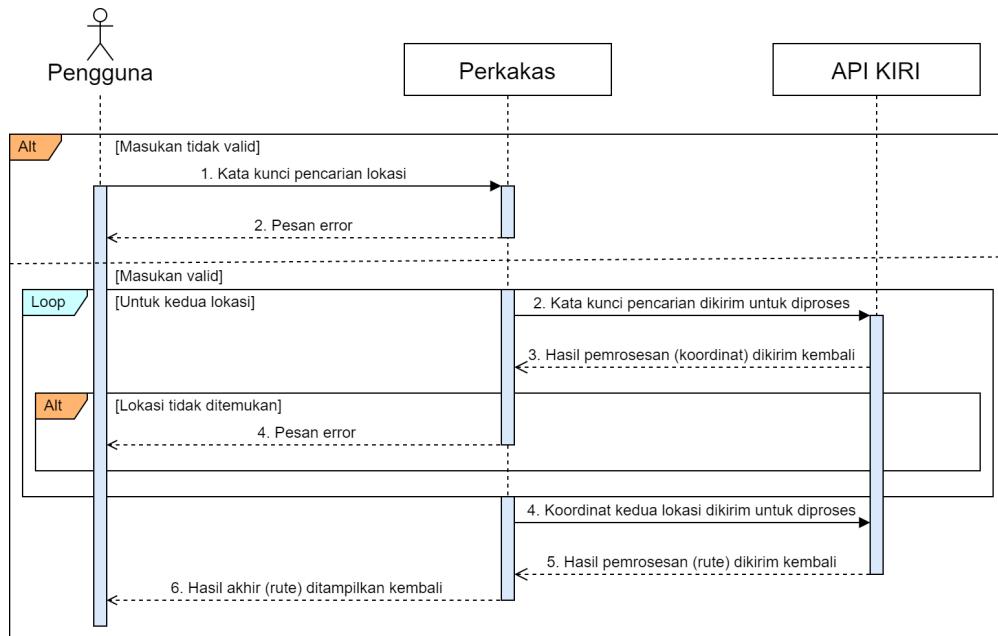
*String* (berupa *array* karakter) yang menyimpan kata kunci pencarian lokasi untuk mode **searchplace**.

- **start (char[])**

*String* (berupa *array* karakter) yang menyimpan koordinat lokasi awal (**findroute**) atau kata kunci pencarian lokasi awal (**direct**).

- **finish (char[])**

*String* (berupa *array* karakter) yang menyimpan koordinat lokasi akhir (**findroute**) atau kata kunci pencarian lokasi akhir (**direct**).



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

- **escape (char [])**  
Variabel ini dibutuhkan karena pengkodean URL (untuk permintaan GET) tidak mendukung karakter spasi (' '), melainkan "%20". String yang karakter spasinya sudah diganti dengan "%20" akan disimpan sementara di variabel ini, sebelum isinya disalin kembali ke variabel awalnya.
- **restart (int)**  
Kode region lokasi awal berupa bilangan bulat untuk mode `direct`. Bilangan ini memiliki rentang dari -1 hingga 4, dengan tiap bilangan bulat merepresentasikan region yang sama dengan kode bilangan untuk variabel `region`.
- **finish (int)**  
Kode region lokasi akhir berupa bilangan bulat untuk mode `direct`. Bilangan ini memiliki rentang dari -1 hingga 4, dengan tiap bilangan bulat merepresentasikan region yang sama dengan kode bilangan untuk variabel `region`.
- **locale (int)**  
Kode bahasa berupa bilangan bulat. Bilangan ini memiliki rentang dari 0 hingga 2, dengan tiap-tiap bilangan merepresentasikan arti berikut.
  - **0:** id (Indonesia)
  - **1:** en (Inggris)
  - **2:** Kode bahasa tidak valid
- Variabel ini juga merupakan satu-satunya variabel kode `integer` yang, jika tidak diubah nilai awalnya (**0**), tidak akan menyebabkan `error`.
- **step (int)**  
Kode khusus untuk mode `direct` yang menandakan proses mana yang sedang berlangsung—pencarian lokasi awal, pencarian lokasi akhir, atau pencarian rute.
- **error (int)**  
Kode yang menandakan apakah sebuah `error` telah terjadi. Nilai awal dari variabel ini adalah **0**, dan jika variabel ini diganti menjadi **1**, di pengecekan kode `error` selanjutnya, perkakas akan dihentikan.

#### 4.2.4 print\_help()

Fungsi ini merupakan fungsi yang akan dipanggil ketika pengguna memilih opsi standar `--help`. Keterangan singkat dari fungsi ini dapat dilihat di Tabel 4.1.

Tabel 4.1: Detail dari fungsi `print_help()`.

<b>Nama fungsi</b>	<code>print_help()</code>
<b>Tujuan</b>	Menampilkan bantuan penggunaan perkakas.
<b>Input</b>	Fungsi ini tidak memerlukan masukan apapun.
<b>Output</b>	Sekumpulan array karakter yang merupakan bantuan penggunaan perkakas.
<b>Pseudocode</b>	<i>Pseudocode</i> tidak ditulis karena seluruh proses dalam fungsi ini hanya menampilkan sekumpulan <i>array</i> karakter.

#### 4.2.5 replace\_space()

Fungsi ini merupakan fungsi yang berguna untuk mengganti semua karakter spasi (' ') dalam variabel-variabel kata kunci pencarian sebelum isi dari variabel-variabel tersebut dimasukkan ke dalam URL untuk permintaan GET. Hal ini perlu dilakukan karena pengkodean HTML yang digunakan untuk format URL tidak mendukung karakter spasi, dan melainkan mensubstitusi seluruh kejadian karakter tersebut dalam suatu URL menjadi "%20". Fungsi ini akan menerima sebuah *array* karakter berisi URL permintaan sebagai masukannya, mengganti semua karakter spasi yang ada di dalamnya menjadi "%20", dan meletakkan hasilnya di variabel global `escape`. Adapun keterangan singkat dari fungsi ini dapat dilihat di Tabel 4.1.

Tabel 4.2: Detail dari fungsi `replace_space()`.

<b>Nama fungsi</b>	<code>replace_space()</code>
<b>Tujuan</b>	Meng- <i>escape</i> seluruh karakter spasi di <i>string</i> masukan.
<b>Input</b>	Sebuah <i>string</i> berupa <i>array</i> karakter.
<b>Output</b>	String tersebut yang sudah di- <i>escape</i> karakter spasinya, dimasukkan ke variabel global <code>escape</code> .
<b>Pseudocode</b>	Lihat di Algoritma 1.

#### 4.2.6 build\_url\_searchplace()

Fungsi ini merupakan fungsi yang digunakan untuk pembangunan URL permintaan GET untuk proses pencarian lokasi. Keterangan dari fungsi ini dapat dilihat di Tabel 4.3.

Selain itu, fitur ini juga memiliki implementasi tambahan yang tidak ada di API KIRI, yaitu implementasi variabel `locale` untuk fitur pencarian lokasi, yang memungkinkan pemilihan bahasa Indonesia (`id`) atau Inggris (`en`) untuk keluaran serta pesan-pesan `error` yang berhubungan dengan proses tersebut.

---

**Algoritma 1** – Algoritma fungsi `replace_space()`

---

**Input:** *string***Output:** *string* yang sudah diganti semua spasinya ke “%20”

```

j ← 0
for i ← 0 to size of string do                                ▷ Loop sampai i mencapai karakter terakhir array
    if string[i] == '\0' then          ▷ Berhenti jika i 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
return escape

```

---

Tabel 4.3: Detail dari fungsi `build_url_searchplace()`.

Nama fungsi	<code>build_url_searchplace()</code>
<b>Tujuan</b>	Membangun URL untuk permintaan GET pencarian lokasi ke API KIRI.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>locale</code>: Kode bahasa</li> <li>- <code>region</code>: Region dari lokasi yang ingin dicari</li> <li>- <code>query</code>: Kata kunci pencarian lokasi</li> <li>- <code>error</code> (variabel global)</li> </ul>
<b>Output</b>	<code>url</code> (variabel global)
<b>Pseudocode</b>	Lihat di Algoritma 2.

#### 4.2.7 `build_url_findroute()`

Fungsi ini merupakan fungsi yang digunakan untuk pembangunan URL permintaan GET untuk proses pencarian rute angkot. Adapun penjelasan dari fungsi ini dapat dilihat di Tabel 4.4.

Tabel 4.4: Detail dari fungsi `build_url_findroute()`.

Nama fungsi	<code>build_url_findroute()</code>
<b>Tujuan</b>	Membangun URL untuk permintaan GET pencarian rute ke API KIRI.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>locale</code> (variabel global)</li> <li>- <code>start</code>: Koordinat lokasi awal</li> <li>- <code>finish</code>: Koordinat lokasi akhir</li> </ul>
<b>Output</b>	<ul style="list-style-type: none"> <li>- <code>error</code> (variabel global)</li> <li>- <code>url</code> (variabel global)</li> </ul>
<b>Pseudocode</b>	Lihat di Algoritma 3.

#### 4.2.8 reset\_url()

Fungsi ini digunakan untuk mengembalikan isi dari variabel `url` ke nilai semula, dengan cara mengganti isi dari variabel tersebut kembali ke nilai awalnya, yaitu “<https://projectkiri.id/api?version=2>”. Penjelasan singkat dari fungsi ini terdapat di Tabel 4.5.

Tabel 4.5: Detail dari fungsi `reset_url()`.

<b>Nama fungsi</b>	<code>reset_url()</code>
<b>Tujuan</b>	Mengembalikan URL ke bentuk umumnya.
<b>Input</b>	<code>url</code> (variabel global)
<b>Output</b>	<code>url</code> yang sudah dikembalikan nilainya
<b>Pseudocode</b>	<i>Pseudocode</i> fungsi ini tidak ditulis karena fungsi ini hanya berisi satu buah proses penggantian nilai <code>array</code> karakter.

#### 4.2.9 execute\_curl()

Fungsi ini digunakan untuk proses pengiriman permintaan ke API, serta penerimaan respons dari API. Semua utilitas dari library cURL yang dipakai dalam perkakas ini, seperti *handler* kelebihan memori data yang masuk, dan aturan variabel apa yang akan diisi dengan data respons API, akan diimplementasikan hanya di dalam fungsi ini. Adapun keterangan singkat dari fungsi ini tertera di Tabel 4.6.

Tabel 4.6: Detail dari fungsi `execute_curl()`.

<b>Nama fungsi</b>	<code>execute_curl()</code>
<b>Tujuan</b>	Menjalankan proses cURL.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>responsesdata</code> (variabel global)</li> <li>- <code>mode</code> (variabel global)</li> <li>- <code>step</code> (variabel global)</li> </ul>
<b>Output</b>	<code>responsesdata</code> yang sudah diisi dengan data respons cURL
<b>Pseudocode</b>	Lihat Algoritma 4.

#### 4.2.10 print\_curl\_error()

Fungsi ini merupakan fungsi sederhana yang akan mengeluarkan pesan *error* apabila terjadi *error* koneksi ketika perkakas ingin melakukan komunikasi dengan API KIRI. Penjelasan lebih menyeluruh untuk fungsi ini ada di Tabel 4.7.

Tabel 4.7: Detail dari fungsi `print_curl_error()`.

<b>Nama fungsi</b>	<code>print_curl_error()</code>
<b>Tujuan</b>	Mengeluarkan pesan <i>error</i> untuk kegagalan dalam proses cURL.
<b>Input</b>	<code>error</code> (variabel global)
<b>Output</b>	Pesan <i>error</i> yang memberitahu pengguna bahwa telah terjadi <i>error</i> dalam proses cURL.
<b>Pseudocode</b>	Lihat Algoritma 5.

#### 4.2.11 write\_memalloc()

Fungsi ini adalah fungsi tambahan untuk libcurl yang bertugas menerima data yang masuk dari API. Di fungsi ini juga diimplementasikan sebuah *handler* pengalokasian memori, yang akan menghindari

terjadinya *error* akibat ukuran data yang diterima melebihi alokasi memori yang diperbolehkan untuk variabel yang akan diisi dengan data tersebut. Adapun penjelasan singkat dari fungsi ini dapat dilihat di Tabel 4.8.

Tabel 4.8: Detail dari fungsi `write_memalloc()`.

Nama fungsi	<code>write_memalloc()</code>
<b>Tujuan</b>	<ul style="list-style-type: none"> <li>- Mengurus penerimaan data dari hasil proses cURL.</li> <li>- Mencegah <i>error</i> akibat kegagalan alokasi memori data yang diterima.</li> </ul>
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>incomingdata</code>: Variabel sementara data yang diterima</li> <li>- <code>size</code>: Ukuran dari satu <i>block</i> data</li> <li>- <code>nmenb</code>: Berapa banyak <i>block</i> data yang diterima</li> <li>- <code>userdata</code>: Variabel yang diisi dengan data yang diterima</li> </ul>
<b>Output</b>	<code>realsize</code> : Untuk verifikasi keutuhan data yang diterima
<b>Pseudocode</b>	Lihat Algoritma 6.

#### 4.2.12 `write_searchplace()`

Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian lokasi dari mode `searchplace`. Penjelasan singkat dari fungsi ini tertera di Tabel 4.9.

Tabel 4.9: Detail dari fungsi `write_searchplace()`.

Nama fungsi	<code>write_searchplace()</code>
<b>Tujuan</b>	Memproses respons API untuk fitur pencarian lokasi dan menampilkan hasilnya ke pengguna.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>responsesdata</code> (variabel global)</li> <li>- <code>responseJSON</code> (variabel global)</li> <li>- <code>error</code> (variabel global)</li> <li>- <code>locale</code> (variabel global)</li> </ul>
<b>Output</b>	Keluaran akhir perkakas untuk mode <code>searchplace</code>
<b>Pseudocode</b>	Lihat Algoritma 7.

#### 4.2.13 `write_findroute()`

Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian rute angkot. Perlu diperhatikan bahwa ada sebuah *bug* dari API KIRI di mana durasi rute dalam menit tidak diterjemahkan dari bahasa Inggris, apabila variabel `locale` diatur ke bahasa Indonesia. Potongan kode untuk mengatasi *bug* ini juga akan terdapat di dalam fungsi ini. Adapun penjelasan lebih lanjut dari fungsi ini dapat dilihat di Tabel 4.10.

Tabel 4.10: Detail dari fungsi `write_findroute()`.

<b>Nama fungsi</b>	<code>write_findroute()</code>
<b>Tujuan</b>	Memproses respons API untuk fitur pencarian rute dan menampilkan hasilnya ke pengguna.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>responsedata</code> (variabel global)</li> <li>- <code>responseJSON</code> (variabel global)</li> <li>- <code>error</code> (variabel global)</li> <li>- <code>locale</code> (variabel global)</li> </ul>
<b>Output</b>	Keluaran akhir perkakas untuk mode <i>findroute</i>
<b>Pseudocode</b>	Lihat Algoritma <a href="#">8</a> .

#### 4.2.14 `write_searchplace_noreturns()`

Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian lokasi dari mode `direct`. Berbeda dengan fungsi `searchplace` sebelumnya, fungsi ini akan mengambil data dari variabel `responsedata`, tetapi hanya akan menampilkan nama lokasi pertama yang ditemukan, atau jika terjadi `error` di salah satu prosesnya, maka fitur ini akan menampilkan pesan `error` yang sesuai. Adapun penjelasan lebih lanjut dari fungsi ini dapat dilihat di Tabel [4.11](#).

Tabel 4.11: Detail dari fungsi `write_searchplace_noreturns()()`.

<b>Nama fungsi</b>	<code>write_searchplace_noreturns()()</code>
<b>Tujuan</b>	Memproses respons API untuk fitur pencarian lokasi dan menampilkan hanya nama lokasi pertama yang ditemukan ke pengguna.
<b>Input</b>	<ul style="list-style-type: none"> <li>- <code>responsedata</code> (variabel global)</li> <li>- <code>responseJSON</code> (variabel global)</li> <li>- <code>error</code> (variabel global)</li> <li>- <code>step</code> (variabel global)</li> <li>- <code>locale</code> (variabel global)</li> </ul>
<b>Output</b>	Nama lokasi pertama yang ditemukan
<b>Pseudocode</b>	Lihat Algoritma <a href="#">9</a> .

#### 4.2.15 Fungsi utama (`main`)

Tentunya seluruh perangkat lunak bahasa C akan memiliki satu buah fungsi utama. Untuk perkakas ini, alur kerja dari fungsi utamanya adalah sebagai berikut.

1. Perkakas menerima seluruh opsi dan argumen dari masukan.
2. Terjemahkan setiap opsi (serta argumennya, jika ada) ke nilai variabelnya masing-masing.
3. Kalau ada kelebihan opsi/argumen atau ada opsi/argumen yang tidak tepat/valid, `return 1` sebagai penanda bahwa perkakas keluar dengan `error`, dan keluarkan pesan `error` yang sesuai.
4. Kalau semua masukan valid, lanjutkan proses sesuai variabel-variabel yang telah diterjemahkan tadi, sampai keluaran akhir dapat ditampilkan.

*Pseudocode* dari alur kerja ini dapat dilihat di Algoritma [10](#).

---

**Algoritma 2** – Algoritma fungsi `build_url_searchplace()`

**Input:** *locale, region, query, error*

**Output:** *url* permintaan GET untuk API

```
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
return url
```

---

---

**Algoritma 3** – Algoritma fungsi `build_url_findroute()`

---

**Input:** *locale, start, finish, error*

**Output:** *url* permintaan GET untuk API

```
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
return url
```

---

---

**Algoritma 4** – Algoritma fungsi `execute_curl()`

---

**Input:** *responsedata, mode, step***Output:** *responsedata* yang sudah diisi dengan respons API

```

curl ← penunjuk ke handle cURL
curlcode ← kode respons cURL
kosongkan responsedata                                ▷ 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 atau 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
return responsedata
```

---



---

**Algoritma 5** – Algoritma fungsi `print_curl_error()`

---

**Input:** *error***Output:** Pesan *error* terjadinya *error* cURL

```

if error == 1 then
    print pesan error sesuai locale: telah terjadi error cURL
end if
```

---

**Algoritma 6 – Algoritma fungsi write\_memalloc()****Input:** *incomingdata, size, nmemb, 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

```

**Algoritma 7 – Algoritma fungsi write\_searchplace()****Input:** *responsesdata, responseJSON, error, locale***Output:** Keluaran akhir fitur pencarian lokasi

```

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
        for each resultitem in result do                      ▷ Untuk setiap lokasi dalam reestresult...
            resultitemname ← "placename" dalam resultitem
            resultitemlocation ← "location" dalam resultitem
            print resultitemname dan resultitemlocation
            indexitem ← indexitem + 1
        end for
    end if
end if

```

---

**Algoritma 8** – Algoritma fungsi `write_findroute()`

---

**Input:** `responsedata`, `responseJSON`, `error`, `locale`

**Output:** Keluaran akhir fitur pencarian rute

```

responseJSON ← data dalam responsedata
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 result...
        routesteps ← "steps" dalam resultitem
        routetime ← "traveltimes" 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
                routetimestring ← hasil terjemahan "minute" dalam routetime ke "menit"
            else
                routetimestring ← routetime
            end if
            print routetimestring
            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
```

---

---

**Algoritma 9** – Algoritma fungsi `write_searchplace_noreturns()`

---

**Input:** *responsesdata, responseJSON, error, step, locale*

**Output:** Nama lokasi pertama yang ditemukan

```
    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
end if
```

---

---

**Algoritma 10** – Alur kerja fungsi utama perkakas

---

**Input:** Semua variabel global di subbab [4.2.3](#)**Output:** *Exit code* perkakas.

```

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

```

---

Lanjut di halaman berikutnya...

---

**Algoritma 10** – Lanjutan dari halaman sebelumnya

```

if ada kelebihan argumen then
    error ← 1
    print pesan error sesuai locale: Ada kelebihan argumen
else
    switch funct do
        case -1
            error ← 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 ← 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 ← 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 ← 2
            panggil fungsi: build_url_findroute(locale, start, finish)
            panggil fungsi: execute_curl()
        default
            error ← 1
            print pesan error sesuai locale: mode tidak valid
            hentikan perkakas
    end switch
end if
return 0

```

---

## BAB 5

# IMPLEMENTASI DAN PENGUJIAN

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

### 5.1 Implementasi Kode

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

#### 5.1.1 `print_help()`

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

#### 5.1.2 `replace_space()`

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

#### 5.1.3 `build_url_searchplace()`

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

#### 5.1.4 `build_url_findroute()`

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

#### 5.1.5 `reset_url()`

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

### 5.1.6 `execute_curl()`

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

### 5.1.7 `print_curl_error()`

Fungsi ini akan dipanggil apabila proses curl dari fungsi `execute_curl()` tidak mengembalikan "OK" sebagai kode respons cURL-nya. Cara kerja fungsi ini adalah dengan mengecek kode bahasa (variabel `locale`) dan mengeluarkan pesan *error* yang sesuai. Implementasi fungsi ini dapat dilihat di lampiran A.2, di baris 396 sampai dengan 408.

### 5.1.8 `write_memalloc()`

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

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

- Fungsi ini akan memiliki *return value* bertipe *unsigned integer size\_t*. Hal ini merupakan kewajiban dari library cURL sendiri.
- Seperti yang sudah disebutkan di bagian rancangan implementasi, fungsi ini akan memiliki empat variabel masukan, yaitu:
  - `incomingdata`  
Variabel ini merupakan data keluaran dari API yang diterima dalam proses cURL.
  - `size`  
Variabel ini merupakan ukuran dari satu buah objek data. Variabel ini selalu bernilai 1, yang juga merupakan kewajiban dari libcurl.
  - `nmemb`  
Variabel ini merupakan ukuran dari data keluaran tersebut.
  - `userdata`  
Variabel ini merupakan penunjuk ke variabel dalam perkakas yang akan diisi oleh data yang diterima proses cURL.
- Implementasi cara kerja fungsi ini adalah sebagai berikut:
  1. **Baris 34:** Hitung ukuran dari data yang masuk.
  2. **Baris 35–40:** Cek apakah ukuran data melebihi yang diperbolehkan.
  3. **Baris 41–44:** Setor data yang dimasukkan ke dalam variabel tujuan.
  4. **Baris 46:** Kembalikan ukuran data yang masuk untuk verifikasi keutuhan data.

### 5.1.9 `write_searchplace()`

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

### 5.1.10 `write_findroute()`

Fungsi ini adalah fungsi yang bertugas untuk memproses respons dari API untuk fitur pencarian rute angkot. Sama seperti fungsi `write_searchplace()`, penerapan fungsi ini hanya meliputi pengambilan data yang diterima, konversi format data tersebut dari JSON, dan kemudian menampilkan hasilnya ke pengguna. Implementasi dari fungsi ini dapat dilihat di lampiran kode murni perkakas, baris 218 sampai baris 334. Perlu ditekankan kembali bahwa fitur ini memiliki pembetulan *bug* tambahan, yang diimplementasikan di baris 273 sampai dengan baris 292.

### 5.1.11 `write_searchplace_noreturns()`

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

### 5.1.12 Fungsi utama (`main`)

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

1. **Baris 646–827:** Implementasi `getopt-long` dan pentransferan argumen ke dalam variabel-variabel internal perkakas.
2. **Baris 647 & 648:** Inisialisasi `opterr` dengan nilai 0, untuk mencegah `getopt-long` mengejukan pesan *error default*-nya untuk opsi-opsi yang tidak diketahui ('?'').
3. **Baris 830–848:** Pengecekan kelebihan argumen dalam perintah masukan.
4. **Baris 850–915:** Penentuan langkah-langkah yang harus ditempuh untuk setiap kemungkinan mode operasional.
5. **Baris 917:** *Return code* 0, menandakan bahwa perkakas berhasil berjalan tanpa ada masalah.

### 5.1.13 CMakeLists

Implementasi dari file CMakeLists.txt untuk perkakas ini dapat dilihat di lampiran A.1. Adapun cara kerja file CMakeLists ini dapat dibagi menjadi beberapa bagian sebagai berikut:

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

### 5.1.14 Halaman manual (*man page*)

Kode ini diperlukan untuk fitur halaman manual (*man page*) di sistem operasi berbasis Linux, yang pada dasarnya merupakan bantuan penggunaan perkakas yang lebih panjang dan lebih detail. Implementasi dari perkakas ini ada di lampiran A.3, di mana potongan kode tersebut dapat dibagi

menjadi bagian bagian sebagai berikut, berdasarkan sintaks yang mengawali bagian-bagian dalam kode tersebut,<sup>1</sup> yaitu:

- **.TH**

Judul dari halaman manual. Tanggal dibuat, versi, serta judul dari halaman manualnya sendiri diatur di bagian ini.

- **.SH**

Setiap baris yang diawali dengan sintaks ini akan di-format menjadi judul dari tiap-tiap bab di dalam halaman manual.

- Sisa dari file yang tidak diawali dengan kedua perintah di atas merupakan deskripsi dari tiap-tiap bagian yang ada di dalam halaman manual nantinya.

## 5.2 Pengujian

Bagian ini akan menjelaskan hal-hal yang seputar pengujian perkakas yang telah dibuat—lingkungan pengujian, cara instalasi dan penggunaan perkakas, serta pengujian fungsional perkakas.

### 5.2.1 Lingkungan Perangkat Keras

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

- *Processor*: Intel® Core™ i5-10300H @ 2.50 GHz
- *RAM*: 8 GB
- *Hard disk*: SSD 512 GB (NVMe™ M.2)
- Perangkat keras pendukung: Keyboard

### 5.2.2 Lingkungan Perangkat Lunak

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

- Windows:
  - OS: Windows 10 Home Single Language (64-bit)
  - *Compiler*: MinGW (GNU GCC—versi 12.1.0)
  - *Library*:
    - \* curl (versi 7.83.1)
    - \* cmake (versi 3.24.1)
- Linux:
  - OS: Ubuntu Jammy (22.04)
  - *Compiler*: GNU GCC—versi 11.3.0
  - *Library*:
    - \* curl (versi 7.81.0)
    - \* cmake (versi 3.22.1)

### 5.2.3 Pembangunan dan Instalasi

#### Syarat Instalasi

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

- Windows

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

- vcpkg

---

<sup>1</sup><https://www.linuxhowtos.org/System/creatingman.htm>

- cURL
- CMake

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

- Linux

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

- cURL
- CMake
- libcurl4-openssl-dev
- GNU Make (opsional)

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

### Cara Instalasi

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

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

- Windows:

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

- Linux:

```
cmake ../
```

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.

- Windows

- Opsi `-DCMAKE_TOOLCHAIN_FILE` merupakan metode pengintegrasian vcpkg untuk proyek CMake. Untuk direktori persisnya (dan sintaks lengkap dari opsi ini) dapat dilihat setelah langkah “Using vcpkg with MSBuild/Visual Studio” di halaman panduan instalasi vcpkg.<sup>2</sup>
- Direkomendasikan untuk menginstal *compiler MinGW*, karena *compiler* ini sudah mengikutkan salah satu file *header* yang dibutuhkan oleh perkakas ini. Apabila pengguna menggunakan *compiler* ini, variabel <compiler> harus diisi dengan “*MinGW Makefiles*”.
- **Jangan menggunakan *compiler Visual Studio***, karena *compiler* ini tidak mengandung file *header* C yang dibutuhkan di perkakas ini. Perlu diperhatikan juga bahwa *compiler* Visual Studio ini merupakan nilai *default* dari <compiler> untuk sistem operasi Windows, jadi pengguna juga tidak boleh menghilangkan opsi `-G` tersebut begitu saja.

---

<sup>2</sup><https://vcpkg.io/en/getting-started.html>

- Linux

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

#### 4. Lanjutkan dengan instalasi perkakas.

- Windows:

Jalankan perintah berikut.

```
cmake --build .
```

- Linux:

Jalankan kedua perintah berikut.

```
cmake --build .
```

```
cmake --install .
```

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

```
make install
```

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

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

### 5.2.4 Pengujian

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

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

No.	Objek tes	Jumlah tes
1	Sintaks dasar	7
2	Mode bantuan	4
3	Mode <i>searchplace</i>	9
4	Mode <i>findroute</i>	8
5	Mode <i>direct</i>	10
6	Integrasi perkakas <i>command linelain</i>	6

Tabel 5.1: Jumlah kategori dan tes yang dilakukan.

#### Sintaks dasar

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

##### 1. Perintah tanpa opsi

- Perintah masukan:

<sup>1</sup> kiritool

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan pesan *error* yang mengingatkan pengguna untuk memasukkan mode operasional perkakas.

- Keluaran perkakas:

1 Error:  
2 Mohon masukkan mode penggunaan perkakas.

- Status tes: **Sukses**

## 2. Perintah dengan satu atau lebih opsi tidak valid

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

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

- Status tes: **Sukses**

## 3. Perintah tanpa argumen di akhir perintah

- Perintah masukan:

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

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan pesan *error* mengenai adanya opsi yang kehilangan argumennya di dalam perintah masukan.

- Keluaran perkakas:

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

- Status tes: **Sukses**

## 4. Perintah tanpa argumen di tengah perintah

- Perintah masukan:

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

- 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: id  
3 Mohon periksa kembali penulisan perintah yang anda masukkan.

- Status tes: **Sukses**

- Catatan tambahan:

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 kiritool --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 kiritool -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. Penggunaan banyak mode sekaligus

- Perintah masukan:

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

- Keluaran yang diharapkan:

Perkakas hanya akan merespons 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**

## Mode bantuan

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

### 1. Panggilan bantuan normal

- Perintah masukan:

```
1 kiritool --help
```

- Keluaran yang diharapkan:

Perkakas akan mengeluarkan bantuan penggunaan perkakas.

- Keluaran perkakas:

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

- Status tes: **Sukses**

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

- Perintah masukan:

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

- Keluaran yang diharapkan:  
Perkakas akan mengeluarkan bantuan penggunaan perkakas, tanpa memerlukan opsi serta argumen tambahan di dalam perintah.
- Keluaran perkakas:  
Keluaran dari kasus tes ini sama seperti kasus sebelumnya, yang juga dapat dilihat di lampiran A.4.
- Status tes: **Sukses**

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

- Perintah masukan:

```
1 man kiritool
```

- Keluaran yang diharapkan:  
*Terminal* akan menampilkan *man page* dari perkakas yang sesuai.
- Keluaran perkakas:  
*Terminal* menampilkan *man page* yang sesuai. Hasil dari kasus tes ini dapat dilihat di lampiran A.5. Perlu diingat bahwa *man page* memiliki *formatting* otomatisnya sendiri, jadi ketika dikonversikan ke file .txt (agar bisa dimasukkan sebagai lampiran), beberapa aspek *formatting*-nya menjadi tidak sempurna.
- Status tes: **Sukses**

### 4. Pemanggilan versi pendek dari *manual page* (*whatis*—khusus Linux)

- Perintah masukan:

```
1 whatis kiritool
```

- Keluaran yang diharapkan:  
*Terminal* akan menampilkan nama dan deskripsi singkat perkakas, sesuai dengan isi dari bagian **NAME** dalam *man page* perkakas.
- Keluaran perkakas:

```
1 kiritool (1)      - search routes using angkot
```

- Status tes: **Sukses**

## Mode *searchplace*

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

### 1. Pencarian lokasi sukses dengan satu hasil

- Perintah masukan:

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

- Keluaran yang diharapkan:  
Perkakas akan menampilkan nama dan koordinat *latitude* dan *longitudelokasi*, sesuai dengan bahasa yang diminta oleh pengguna.
- Keluaran perkakas:

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

- Status tes: **Sukses**

### 2. Pencarian lokasi sukses dengan lebih dari satu hasil

- Perintah masukan:

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

- Keluaran yang diharapkan:  
Perkakas akan menampilkan nama dan koordinat *latitude* dan *longitudesemua kemungkinan lokasi*, sesuai dengan bahasa yang diminta oleh pengguna.

- Keluaran perkakas:

```

1 Lokasi 1:
2 -----
3 Nama: Blk. A-B
4 Koordinat: -6.88612,107.65028
5
6 Lokasi 2:
7 -----
8 Nama: Blk. A-B
9 Koordinat: -6.90845,107.67641
10
11 Lokasi 3:
12 -----
13 Nama: Blk. AB
14 Koordinat: -6.89748,107.70782

```

- Status tes: **Sukses**

### 3. Pencarian lokasi gagal

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

```

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

```

- Status tes: **Sukses**

### 4. Pencarian lokasi tanpa opsi region

- Perintah masukan:

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

- Keluaran yang diharapkan:

Perkakas akan menampilkan pesan *error* yang mengingatkan pengguna untuk memasukkan kode region.

- 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

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

```

1 Lokasi:
2 -----
3 Nama: Universitas Katolik Parahyangan
4 Koordinat: -6.87520,107.60492

```

- Status tes: **Sukses**

#### 9. Pencarian lokasi dengan pengaturan bahasa yang tidak valid

- Perintah masukan:

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

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**

### Mode *findroute*

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

#### 1. Pencarian rute sukses dengan satu kemungkinan rute

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

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

- Status tes: **Sukses**

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

- Perintah masukan:

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

- Keluaran yang diharapkan:

Jauhnya rute tidak akan mempengaruhi kinerja dari perkakas.

- Keluaran perkakas:

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

- Status tes: **Sukses**

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

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

```
1 Rute 1:
2 -----
3 Estimasi waktu: 30 menit
4 -----
5 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 56 meter.
6 Langkah 2: Naik angkot Cicaheum - Ciroyom di Jalan Cihampelas, dan turun di Jalan Arjuna kurang lebih setelah 4,2
   kilometer.
7 Langkah 3: Jalan sedikit di Jalan Arjuna.
8 Langkah 4: Naik angkot Dago - Caringin di Jalan Arjuna, dan turun di Jalan Kebon Jati kurang lebih setelah 688 meter
   .
9 Langkah 5: Jalan dari Jalan Kebon Jati ke lokasi mulai Anda sejauh kurang lebih 138 meter.
10 -----
11 Rute 2:
12 -----
13 Estimasi waktu: 25 menit
14 -----
15 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 26 meter.
16 Langkah 2: Naik angkot Cisitu - Tegallega di Jalan Cihampelas, dan turun di Jalan Pasir Kaliki kurang lebih setelah
   3,7 kilometer.
17 Langkah 3: Jalan dari Jalan Pasir Kaliki ke lokasi mulai Anda sejauh kurang lebih 396 meter.
```

- Status tes: **Sukses**

#### 4. Pencarian rute gagal

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

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

- Status tes: **Sukses**

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

- Perintah masukan:

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

- Keluaran yang diharapkan:

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.

- Keluaran perkakas:

**Kasus 1:** Lokasi awal tidak dimasukkan

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

**Kasus 2:** Lokasi akhir tidak dimasukkan

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

- Status tes: **Sukses**

#### 6. Pencarian rute dengan koordinat lokasi yang tidak valid

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

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

- Status tes: **Sukses**

- Catatan tambahan:

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.

#### 7. Pencarian lokasi tanpa pengaturan bahasa

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

```
1 Rute:
2 -----
3 Estimasi waktu: 25 menit
4 -----
5 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 203 meter.
6 Langkah 2: Naik angkot Ciumbuleuit - St. Hall (belok) di Jalan Cihampelas, dan turun di 40141 kurang lebih setelah
   3,9 kilometer.
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**

## 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 --regstart 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 rutennya, 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 --regstart cgk --start untar --regfinish bdo --finish unpar --locale id
```

- Keluaran yang diharapkan:

Jauhnya rute tidak akan berpengaruh ke keluaran perkakas.

- Keluaran perkakas:

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

- Status tes: **Sukses**
3. Pencarian rute langsung sukses dengan lebih dari satu kemungkinan rute
- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

```
1 Lokasi awal: Cihampelas Walk Extention
2 Lokasi akhir: 23 Paskal Shopping Center
3
4 Rute 1:
5 -----
6 Estimasi waktu: 30 menit
7 -----
8 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 56 meter.
9 Langkah 2: Naik angkot Cicahem - Ciroyom di Jalan Cihampelas, dan turun di Jalan Arjuna kurang lebih setelah 4,2
   kilometer.
10 Langkah 3: Jalan sedikit di Jalan Arjuna.
11 Langkah 4: Naik angkot Dago - Caringin di Jalan Arjuna, dan turun di Jalan Kebon Jati kurang lebih setelah 688 meter
12 Langkah 5: Jalan dari Jalan Kebon Jati ke lokasi mulai Anda sejauh kurang lebih 138 meter.
13
14 Rute 2:
15 -----
16 Estimasi waktu: 25 menit
17 -----
18 Langkah 1: Jalan dari tujuan akhir Anda ke Jalan Cihampelas sejauh kurang lebih 26 meter.
19 Langkah 2: Naik angkot Cisitu - Tegallega di Jalan Cihampelas, dan turun di Jalan Pasir Kaliki kurang lebih setelah
   3,7 kilometer.
20 Langkah 3: Jalan dari Jalan Pasir Kaliki ke lokasi mulai Anda sejauh kurang lebih 396 meter.
```

- Status tes: **Sukses**

4. Pencarian rute langsung gagal

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

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

```

- Status tes: **Sukses**

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

- Perintah masukan:

```

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

```

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

**Kasus 1:** Lokasi awal tidak dimasukkan

```

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

```

**Kasus 2:** Lokasi akhir tidak dimasukkan

```

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

```

- Status tes: **Sukses**

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

- Perintah masukan:

```

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

```

- Keluaran yang diharapkan:

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*.

- Keluaran perkakas:

**Kasus 1:** Lokasi awal tidak valid

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

**Kasus 2:** Lokasi akhir tidak valid

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

- Status tes: **Sukses**

9. Pencarian rute langsung sukses tanpa pengaturan bahasa

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- 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**

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

- Perintah masukan:

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

## Integrasi perkakas *command linelain*

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

- Tes kasus yang akan digunakan sebagai masukan adalah pencarian rute langsung dengan satu kemungkinan rute, dari “ciwalk” (Bandung) ke “unpar” (Bandung), dalam bahasa Inggris. Masukan ini akan disamakan untuk seluruh bagian ini untuk alasan konsistensi.
- Pengujian akan dilakukan sebanyak dua kali untuk tiap kasus, untuk Windows dan Linux.

- Pengujian hanya akan dilakukan sekali untuk kasus-kasus tertentu apabila fungsi yang diuji memiliki perintah yang sama, atau fungsi tersebut tidak didukung oleh perkakas *command line*bawaan dalam sistem operasi tersebut.

Berikut merupakan hasil pengujian untuk bagian ini.

1. Memasukkan keluaran perkakas ke dalam file output

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas: Akan dihasilkan sebuah file **out.txt** yang isinya adalah sebagai 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
   later.
10 Step 3: Walk about 64 meter from 40141 to your destination.
```

- 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
   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
   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.

- Keluaran perkakas:

1 Estimated duration: 25 minutes

- Status tes: **Sukses**

#### 5. Mengeluarkan hanya durasi dari rute (Linux)

- Perintah masukan:

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

- 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 Estimated duration: 25 minutes

- Status tes: **Sukses**

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

- Perintah masukan:

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

- Keluaran yang diharapkan:

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

- Keluaran perkakas:

1 3

- Status tes: **Sukses**

- Catatan tambahan:

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

## BAB 6

### KESIMPULAN

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

#### 6.1 Kesimpulan

Berikut merupakan kesimpulan yang diambil dari penulisan skripsi ini:

- Telah berhasil dibuat sebuah perkakas bernama “*KIRI Tool*”, yang merupakan perkakas *command line* yang mengimplementasikan fitur-fitur API *KIRI*.
- Integrasi perkakas *KIRI Tool* dengan perkakas-perkakas *command line* lainnya dapat dilakukan dengan beberapa metode, dengan metode yang paling umum adalah menggunakan *grep* (atau perkakas lain dengan fungsi yang sama) untuk mengekstraksi aspek tertentu dari keluaran perkakas.

#### 6.2 Saran

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

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



## DAFTAR REFERENSI

- [1] Marsh, N. (2010) *Introduction to the Command Line: The Fat-Free Guide to Unix and Linux Commands*, 2nd edition. CreateSpace, South Carolina.
- [2] Shotts Jr., W. E. (2019) *The Linux Command Line*, 5th internet edition. <https://www.linuxcommand.org/tlcl.php>.
- [3] Matthew, N. dan Stones, R. (2007) *Beginning Linux® Programming*, 4th edition. Wiley Publishing, Inc., Indiana.
- [4] Reddy, M. (2011) *API Design for C++*, 1st edition. Elsevier, Inc., Massachussetts.
- [5] Raymond, E. S. (2003) *The Art of UNIX Programming*, 1st edition. Addison-Wesley Professional, Boston.
- [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] Fellman, T. dan Kavakli, M. (2007) A command line interface versus a graphical user interface in coding vr systems. *Proceedings of the 2nd IASTED International Conference on Human-Computer Interaction, HCI 2007*, California, March, pp. 142–147. ACTA Press.
- [8] Microsoft Docs (2021) Windows commands. <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands>. versi 01 Mei 2022.
- [9] Loosemore, S. dkk. (2022) *The GNU C Library Reference Manual*, 2.35 edition. Free Software Foundation, Inc., Massachusetts.
- [10] Stenberg, D. (2022) *Everything curl*. GitBook, Rhone-Alpes.
- [11] Martin, K. dan Hoffman, B. (2013) *Mastering CMake*, 6th edition. Kitware, Inc., New York.



# 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 locale, 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(locale, 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(locale, regstart, start);
881             execute_curl();
882             if (error == 1) {
883                 break;
884             }
885             else reset_url();
886
887             step = 1;
888             build_url_searchplace(locale, 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

```