BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Sistem

Implementasi adalah tahapan di mana hasil dari analisis dan perancangan dilaporkan. Implementasi pembangunan perangkat lunak diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan.

4.1.1. Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk pengembangan dan pengoperasian perangkat lunak yang dibangun antara lain:

1. Prosesor Pentium 2.2 GHz
2. Memori 1 GB
3. Monitor 14"
4. *Harddisk* 50 MB
5. *Keyboard* qwerty laptop

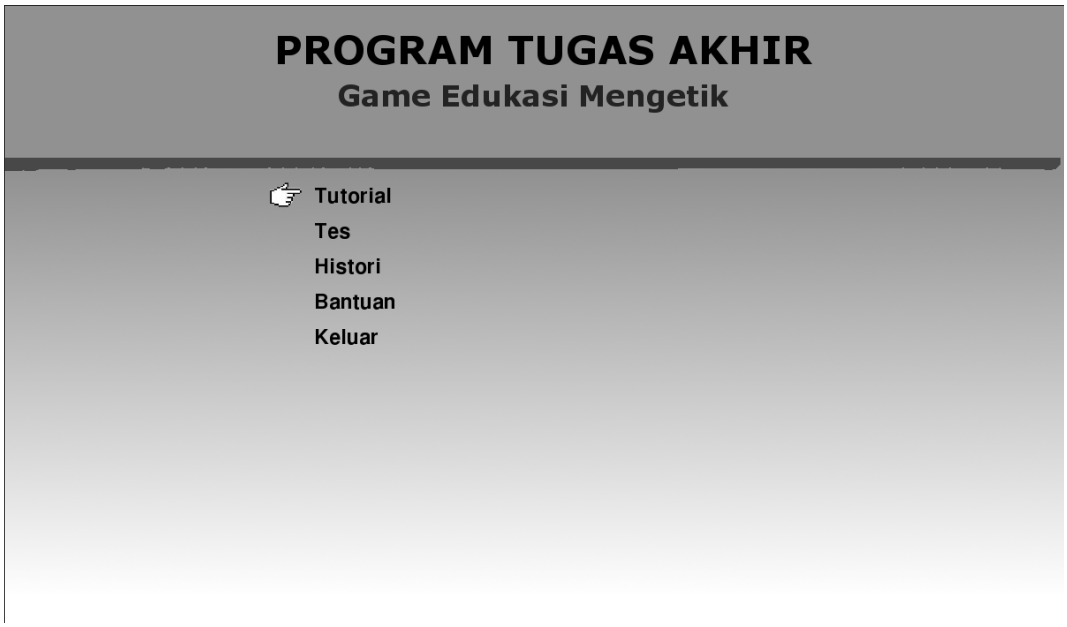
4.1.2. Implementasi Perangkat Lunak

Program *game* edukasi dibangun memakai *software* pendukung yang dijabarkan sebagai berikut:

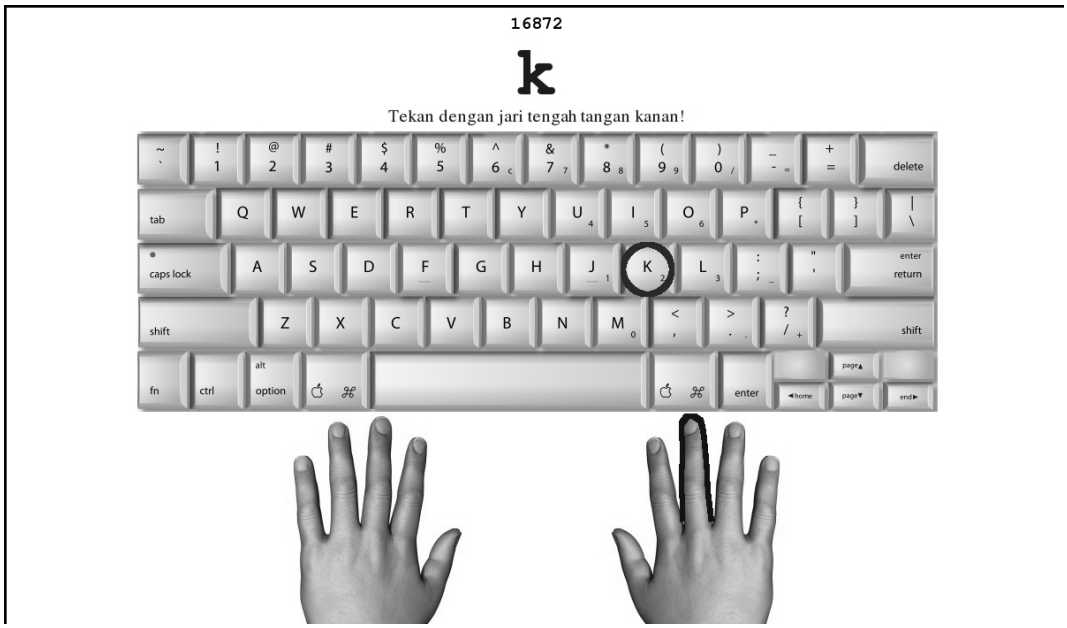
1. Sistem operasi *Windows*
2. Editor *Dev C++*
3. *Library Simple DirectMedia Layer* (SDL)

4.1.3. Implementasi Antarmuka

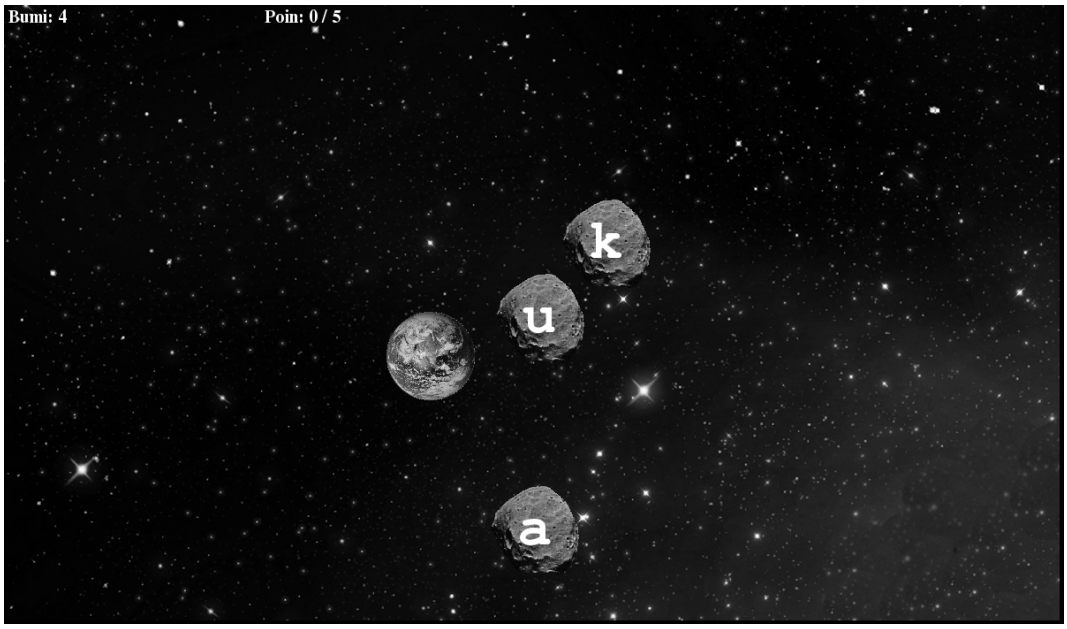
Implementasi antarmuka merupakan bagian yang memperlihatkan bahwa program berjalan dengan baik lewat tampilan program di layar monitor. Berikut ini adalah *screenshot* dari program yang dibangun dari Gambar 4.1 sampai dengan Gambar 4.8.



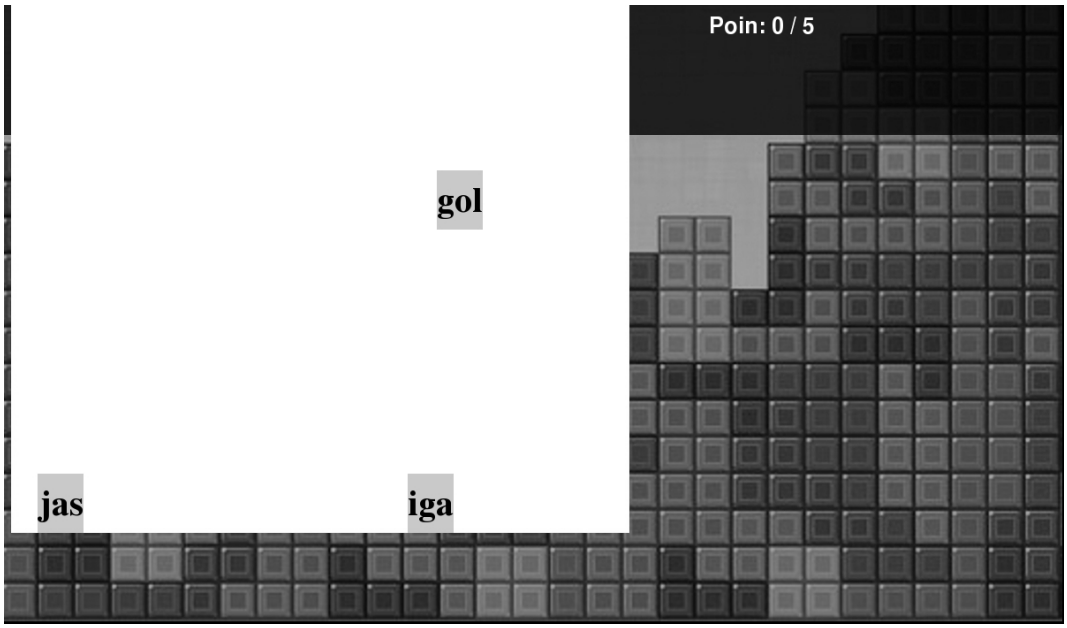
Gambar 4.1. Tampilan menu utama.



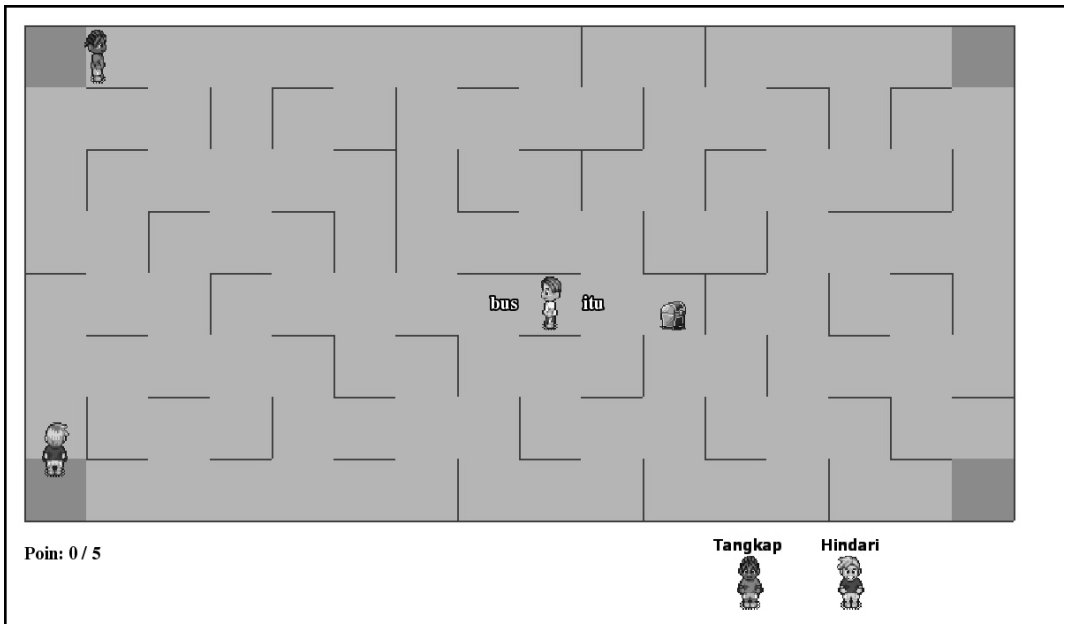
Gambar 4.2. Tampilan permainan tutorial



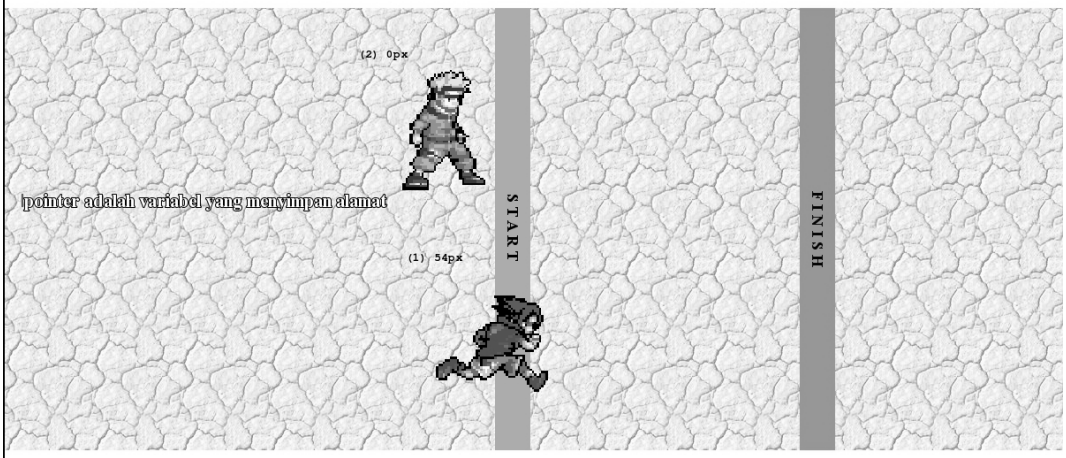
Gambar 4.3. Tampilan permainan tes 1.



Gambar 4.4. Tampilan permainan tes 2.



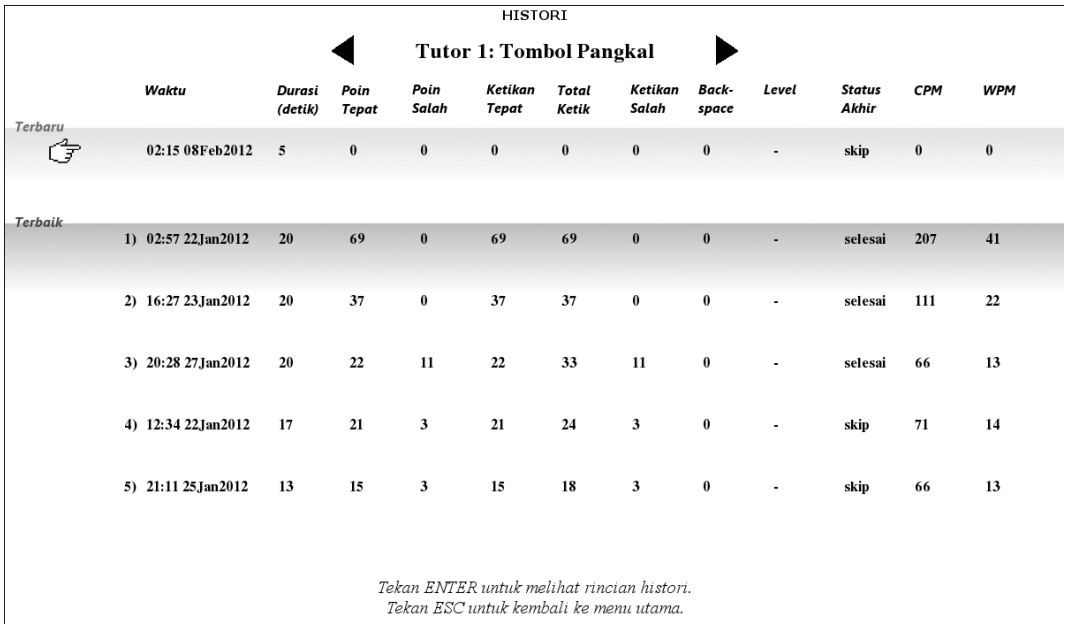
Gambar 4.5. Tampilan permainan tes 3.



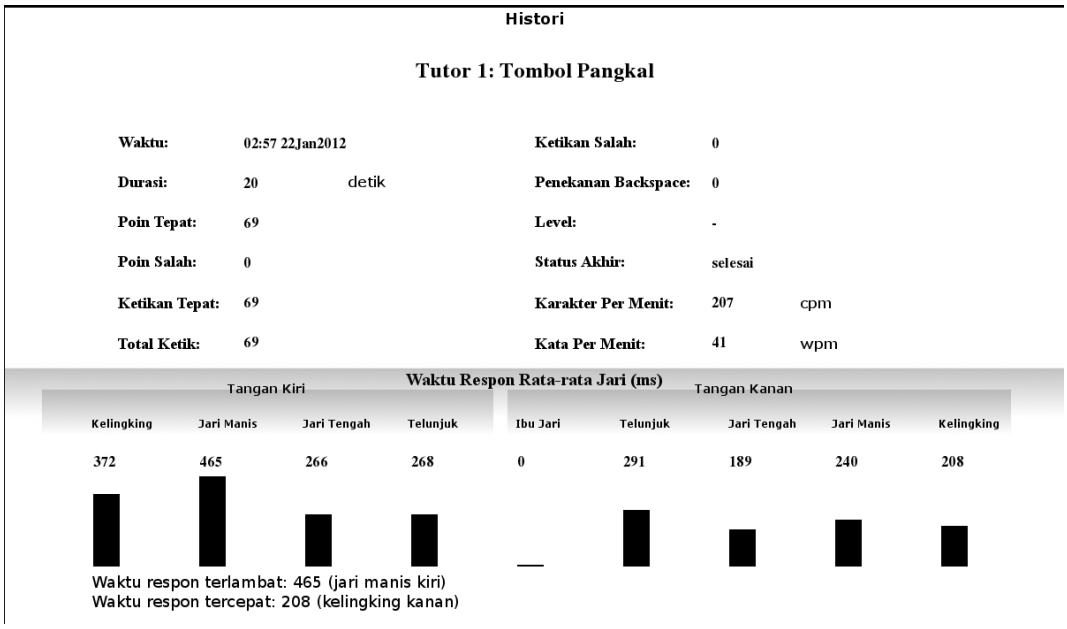
Gambar 4.6. Tampilan permainan tes 4.



Gambar 4.7. Tampilan bantuan



Gambar 4.8. Tampilan histori.



Gambar 4.9. Tampilan histori detil.

4.2. Pengujian Sistem

Sebuah aplikasi komputer yang dibangun haruslah bebas dari kesalahan atau *error*. Oleh karena itu, aplikasi harus diuji coba terlebih dahulu untuk mengetahui apakah masih terdapat *error* atau tidak .

4.2.1. Pengujian Alpha

Pengujian alpha dilakukan pada sisi pengembang yang dianggap pemakai dan merekam semua kesalahan dan masalah pemakaian. Pengujian alpha dilakukan pada sebuah lingkungan yang terkendali.

Pendekatan perancangan kasus uji menggunakan dua metode, yaitu *white box* dan *black box*. *White box* adalah metode perancangan kasus uji yang menggunakan struktur kontrol program. *Black box* adalah metode pengujian yang berfokus pada persyaratan fungsional perangkat lunak. Rencana pengujian dapat dilihat pada Tabel 4.1.

Tabel 4.1. Rencana pengujian.

No. Urut	Yang Diuji	Metode Pengujian
1	Pencarian rute terpendek	*White box*
2	Navigasi menu utama	*Black box*
3	*Play* tutorial	*Black box*
4	*Play* tes 1	*Black box*
5	*Play* tes 2	*Black box*
6	*Play* tes 3	*Black box*
6	*Play* tes 4	*Black box*

4.2.2. Hasil Pengujian Alpha

Berikut ini adalah hasil pengujian alpha yang sasarannya telah direncanakan sebelumnya.

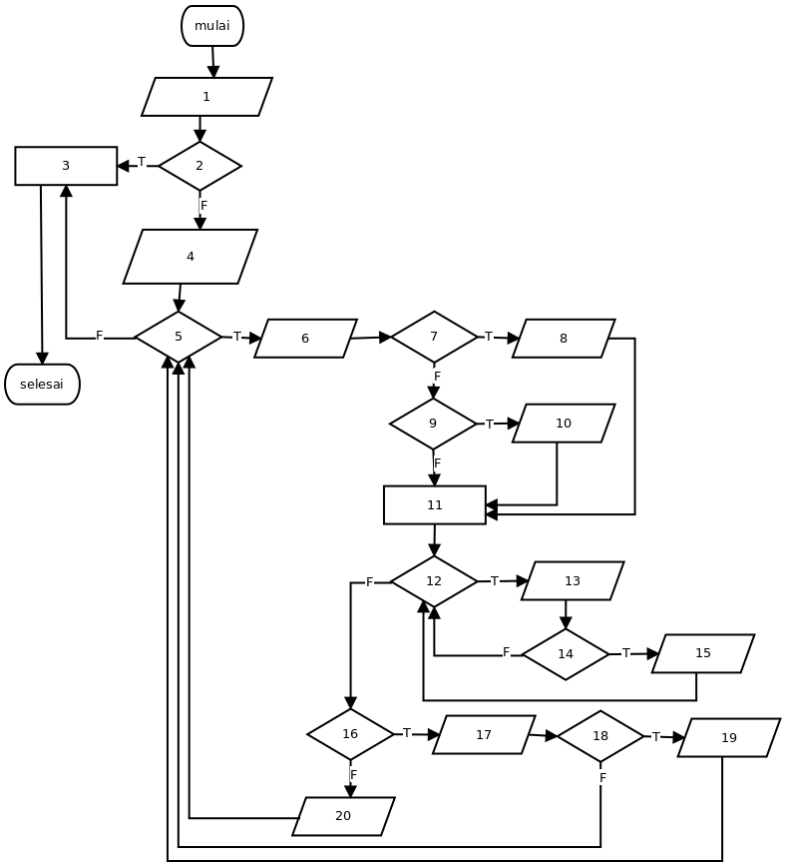
1. Pencarian rute terpendek

Pengujian secara *white box* terhadap proses pencarian rute terpendek dilakukan untuk meyakinkan semua perintah dan kondisi perangkat lunak *game* edukasi dieksekusi secara minimal. Teknik yang digunakan yaitu *Flow Graph* yang digunakan untuk menggambarkan alur dari algoritma dan *Graph* Matriks yang digunakan untuk merepresentasikan *flow graph* dalam bentuk tabel. Struktur program pencarian rute terpendek dapat dilihat di bawah ini:

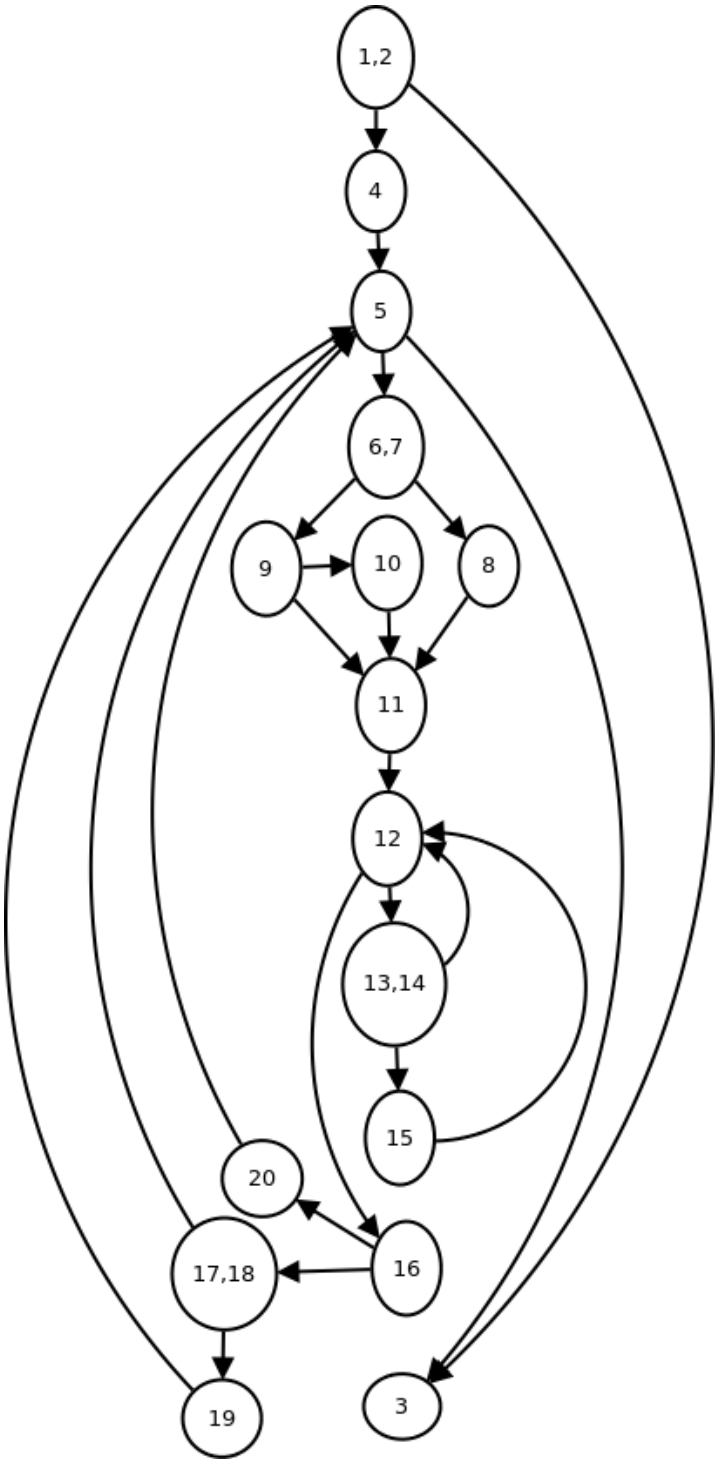
```
1.    short int hitB = posisiInitialState;
      short int tgt = posisiGoalState;
2.    if(hitB == tgt)  // cek apakah inisial = gol
3.          return;  // hentikan prosedur
4.    short int bCnt = map->bCnt;  // jumlah blok maksimal
      _rgRoMaBlock **b = map->b;  // dapatkan nilai blok labirin
      short int tCnt = 0, maxTCnt = 30;  // jumlah elemen rute
      _track track[bCnt];  // deklarasi untuk rute
      _rgRoMaBlock *bl;  // deklarasi blok labirin
      short int blId = hitB;  // id blok, blok berikutnya
      trPush(track, &tCnt, -1, blId);  // awal simpul pada rute
      char dir[4];  // deklarasi empat arah
      short int nextBlock = hitB;  // set blok tujuan
      char i, dirI = 0;  // deklarasi tanda empat arah
5.    while(tCnt > 0) {  // ulang sampai tidak ada lagi arah
6.          bl = *(b + blId);  // fokus ke blok yang ditelusuri
7.          if(tCnt >= (maxTCnt-1)){  // batasi hanya empat arah
8.                dirI = 4;  // tanda empat arah sudah diproses
9.          }else if(blId == tgt){  // cek apakah sudah tujuan
10.               maxTCnt = tCnt;  // nilai rute terpendek
                  nextBlock = track[1].d;  // set arah berikutnya
                  dirI = 4;  // tanda empat arah sudah diproses
            }
11.         roQudrant(blId, tgt, dir);  // evaluasi arah utama
12.         for(i=dirI; i<4; i++) {  // telusuri empat arah
13.               blId = bl->con[ dir[i] ];  // dapatkan satu blok
14.               if(blId >= 0 && trPush(track, &tCnt, i, blId)){
15.                     i = 7;  // set simpul baru
                  }
```

```
            }
16.         if( i == 4 ){  // cek apakah empat arah sudah diproses
17.               tCnt--;  // mundur satu elemen
                  bl->tr = 0;  // set ke arah pertama
                  dirI = track[ tCnt ].from + 1;  // ke arah lain
18.               if( tCnt )
19.                     blId = track[ tCnt-1 ].d;
            }else{
20.               dirI = 0;  // mulai dari arah pertama
            }
      }
```

Berdasarkan sintaks sebelumnya, dapat dibuat bagan alir dan *flow graph* untuk menggambarkan ilustrasi aliran struktur proram pencarian rute terpendek. Bagan alir dan *Flow graph* tersebut dapat dilihat pada Gambar 4.10 dan keterangan *flow graph* pada Tabel 4.2.



(a)



(b)

Gambar 4.10. (a) Bagan alir; (b) *Flow graph*.

Tabel 4.2. Keterangan *flow graph*

Simbol *Flow graph*	**Keterangan**
	*Node* (N)
	Busur (E)

Jumlah region pada *flow graph* dihasilkan dari jumlah daerah yang berada di dalam *node* ditambah satu daerah di luar *node*. Rumus penjumlahannya:

Region = daerah_dalam + daerah_luar

= 8 + 1

= 9

Cyclomatic Complexity (V(G)) diperoleh dari perhitungan jumlah busur dikurangi jumlah *node* ditambah konstanta dua. Perhitungannya:

V(G) = E - N + 2

= 23 - 16 + 2

= 9

Independent path ditentukan oleh penelusuran dari *node* awal sampai *node* akhir. Berikut ini adalah sejumlah *path* yang ditemukan dari *flow graph* sebelumnya:

Path 1 = 1-2-3

Path 2 = 1-2-4-5-3

Path 3 = 1-2-4-5-6-7-9-10-11-12-16-20-5-3

Path 4 = 1-2-4-5-6-7-9-11-12-16-20-5-3

Path 5 = 1-2-4-5-6-7-8-11-12-16-20-5-3

Path 6 = 1-2-4-5-6-7-8-11-12-13-14-12-16-20-5-3

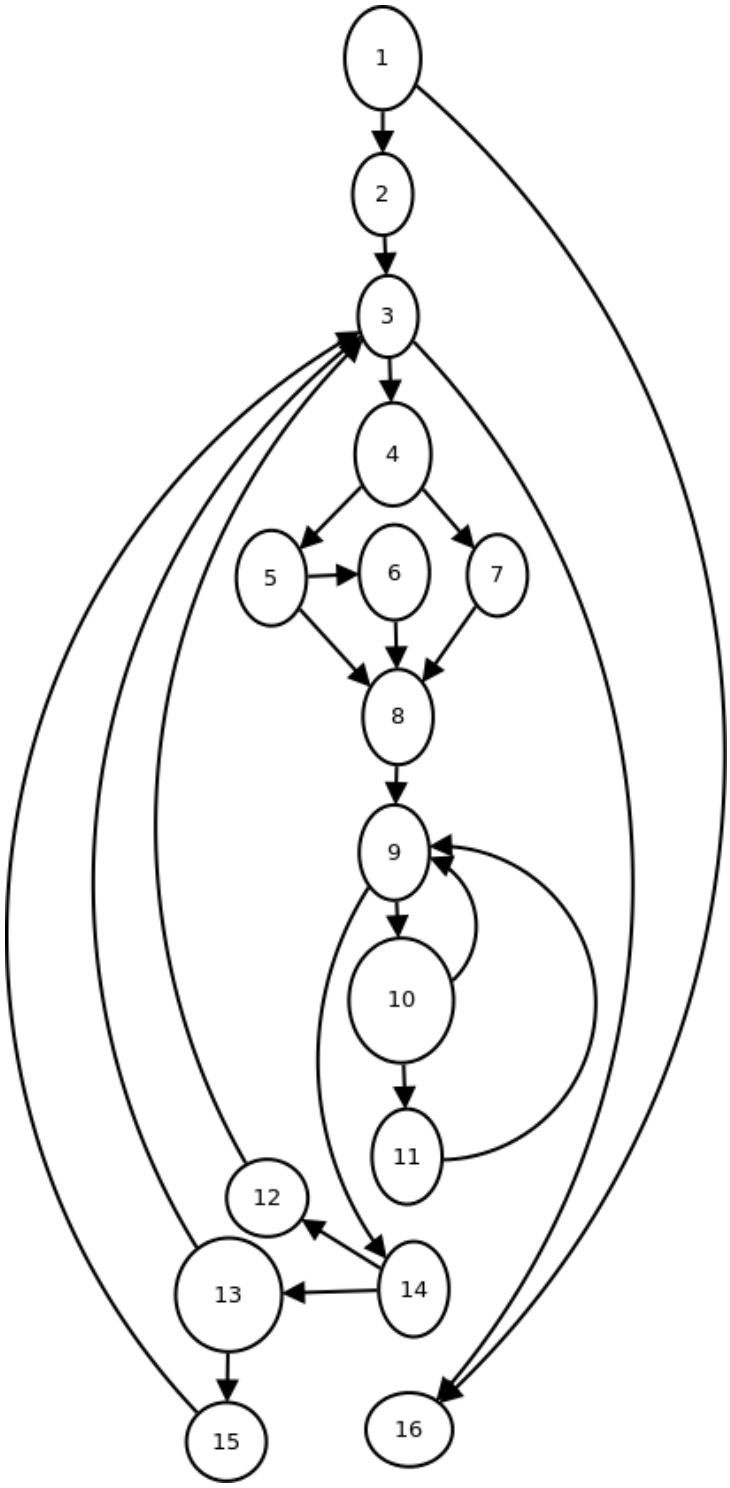
Path 7 = 1-2-4-5-6-7-8-11-12-13-14-15-12-16-20-5-3

Path 8 = 1-2-4-5-6-7-8-11-12-16-17-18-5-3

Path 9 = 1-2-4-5-6-7-8-11-12-16-17-18-19-5-3

Jadi, pada teknik *flow graph*, jumlah region, *Cyclomatic Complexity* dan *Independent Path* yang dihasilkan bernilai sama.

Padak teknik *graph* matriks, satu *node* pada *flow graph* diwakilkan oleh satu nomor agar memudahkan perhitungan matriks. Gambar 4.11 merupakan *flow graph* hasil pengubahan nomor pada *node*.



Gambar 4.11. *Flow graph* pada *graph* matriks.

Tabel 4.3 merupakan matriks yang menyatakan hubungan antara simpul dengan simpul dari busur selanjutnya berdasarkan Gambar 4.11. Kolom *Sum*-1 dihitung berdasarkan jumlah *node* yang ada dikurang satu. Jika *node* dari busur selanjutnya tidak ada, maka dinyatakan sebagai elemen kosong dan bernilai nol.

Tabel 4.3. *Graph* matriks.

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	*Sum*-1
1		1														1	1
2			1														0
3				1												1	1
4					1		1										1
5						1		1									1
6								1									0
7								1									0
8									1								0
9										1				1			1
10									1		1						1
11									1								0
12			1														0
13			1												1		1
14												1	1				1
15			1														0
16																	0
																Jumlah	8

V(G) = jumlah_matriks + 1

= 8 + 1

= 9

Setelah mendapatkan *Cyclomatic Complexity* dari *graph* matriks, perlu dicari nilai *Cyclomatic Complexity* berdasarkan *predicate node* (P), yaitu jumlah simpul yang memiliki lebih dari satu busur.

Berdasarkan Gambar 4.11, simpul-simpul yang memiliki jumlah busur lebih dari satu ialah

1, 3, 4, 5, 9, 10, 13, dan 14

sehingga

Predicate node (P) = 8

Setelah nilai P didapat, untuk mendapatkan nilai *Cyclomatic Complexity* dari *predicate node*, rumus dan perhitungan yang digunakan ialah

V(G) = P + 1

= 8 + 1

= 9

Jadi, nilai hasil dari *graph* matriks dan *predicate node* masing-masing adalah 9.

2. Navigasi menu utama

Hasil pengujian secara *black box* terhadap fungsional navigasi menu utama dapat dilihat pada Tabel 4.4.

Tabel 4.4. Hasil pengujian navigasi menu utama.

Aksi	Harapan	Hasil	Kesimpulan
Jika menunjuk entri pertama, tekan tombol bawah	Penunjuk entri berpindah ke posisi di bawahnya	Penunjuk entri bergerak ke bawah	[x] Diterima [ ] Ditolak
Jika menunjuk entri pertama, tekan tombol atas	Penunjuk entri berpindah ke posisi di entri terbawah	Penunjuk entri bergerak ke entri terbawah	[x] Diterima [ ] Ditolak

3. *Play* tutorial

Hasil pengujian secara *black box* terhadap fungsional *play* tutorial dapat dilihat pada Tabel 4.5.

Tabel 4.5. Hasil pengujian *play* tutorial.

Aksi	Harapan	Hasil	Kesimpulan
Jika karakter a harus diketik, tekan tombol a	Program merespon benar	Menampilkan karakter lain yang harus diketik	[x] Diterima [ ] Ditolak
Jika karakter b harus diketik, tekan tombol a	Program merespon salah	Menjalankan bunyi salah	[x] Diterima [ ] Ditolak
Tekan tombol *Esc*	Program menghentikan sementara permainan tutorial	Muncul menu pause dan permainan tutorial dihentikan sementara	[x] Diterima [ ] Ditolak

4. *Play* tes 1

Hasil pengujian secara *black box* terhadap fungsional *play* tes 1 dapat dilihat pada Tabel 4.6.

Tabel 4.6. Hasil pengujian *play* tes 1.

Aksi	Harapan	Hasil	Kesimpulan
Jika ada karakter a, tekan tombol a	Program merespon benar	Karakter a lenyap dan muncul karakter baru	[x] Diterima [ ] Ditolak
Jika tidak ada karakter a, tekan bombol a	Program mengabaikan efek salah	Tampilan tidak berubah	[x] Diterima [ ] Ditolak
Tekan tombol *Esc*	Program menghentikan sementara permainan tutorial	Muncul menu pause dan permainan tutorial dihentikan sementara	[x] Diterima [ ] Ditolak

5. *Play* tes 2

Hasil pengujian secara *black box* terhadap fungsional *play* tes 2 dapat dilihat pada Tabel 4.7.

Tabel 4.7. Hasil pengujian *play* tes 2.

Aksi	Harapan	Hasil	Kesimpulan
Ketikan belum 20 digit, tekan tombol a	Karakter tampil dan menjadi bagian kata	Karakter ditambahkan ke kata ketikan	[x] Diterima [ ] Ditolak
Ketikan sudah 20 digit, tekan tombol a	Program mengabaikan karakter untuk dimasukkan ke kata	Tidak ditampilkan ke layar	[x] Diterima [ ] Ditolak

Tekan tombol *Esc*	Program menghentikan sementara permainan tutorial	Muncul menu pause dan permainan tutorial dihentikan sementara	[x] Diterima [ ] Ditolak

6. *Play* tes 3

Hasil pengujian secara *black box* terhadap fungsional *play* tes 3 dapat dilihat pada Tabel 4.8.

Tabel 4.8. Hasil pengujian *play* tes 3.

Aksi	**Harapan**	**Hasil**	**Kesimpulan**
Ketikan belum 20 digit, tekan tombol a	Karakter tampil di layar dan menjadi bagian kata	Karakter ditambahkan ke kata ketikan	[x] Diterima [ ] Ditolak
Ketikan sudah 20 digit, tekan tombol a	Program mengabaikan karakter untuk dimasukkan ke kata	Tidak ditampilkan ke layar	[x] Diterima [ ] Ditolak
Tekan tombol *Esc*	Program menghentikan sementara permainan tutorial	Muncul menu pause dan permainan tutorial dihentikan sementara	[x] Diterima [ ] Ditolak

7. *Play* tes 4

Hasil pengujian secara *black box* terhadap fungsional *play* tes 4 dapat dilihat pada Tabel 4.9.

Tabel 4.9. Hasil pengujian *play* tes 4.

Aksi	Harapan	Hasil	Kesimpulan
Karakter a berikutnya diketik, tekan tombol a	Program merespon benar	Karakter ketikan ditambah ke deretan karakter dari frase	[x] Diterima [ ] Ditolak
Bukan karakter a yang berikutnya diketik	Program merespon salah	Menjalankan bunyi *error*	[x] Diterima [ ] Ditolak
Tekan tombol *Esc*	Program menghentikan sementara permainan tutorial	Muncul menu pause dan permainan tutorial dihentikan sementara	[x] Diterima [ ] Ditolak

4.2.3. Kesimpulan Hasil Pengujian Alpha

Hasil pengujian alpha menyatakan bahwa aplikasi *game* yang dibangun telah bebas dari kesalahan prosedur internal program dan secara fungsional mengeluarkan hasil yang sesuai dengan yang diharapkan.

4.2.4. Pengujian Beta

Pengujian beta dilakukan pada satu atau lebih *user* yang merupakan pemakai akhir perangkat lunak yang dibangun.

Dalam pengujian beta, dilakukan penelitian dengan cara memberikan kuesioner pada pengguna yang telah mencoba aplikasi *game*. Kuesioner diberikan

kepada 10 orang pengguna yang telah mencoba aplikasi *game*. Kuesioner terdiri dari 6 pertanyaan dengan menggunakan skala 1 sampai 5.

1. Apakah dalam *game* yang dibangun terdapat edukasi mengetik?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

2. Apakah *game* yang dibangun dapat menyadarkan akan pentingnya mengetik?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

3. Apakah *game* yang pantas menjadi media belajar dan latihan mengetik?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

4. Apakah *game* yang dibangun menghibur?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

5. Apakah *game* yang dibangun dapat meningkatkan kemampuan mengetik?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

6. Apakah *game* yang dibangun dapat membuat *player* terbiasa dengan teks bahasa Indonesia?

No. Jawaban	Keterangan
1	Sangat setuju
2	Setuju
3	Ragu-ragu
4	Kurang setuju
5	Sangat tidak setuju

4.2.5. Hasil Pengujian Beta

Berikut hasil pengujian yang dilakukan melalui kuesioner dari percobaan *user*. Hasil pengujian ini didapat dari pendapat 10 orang responden. Persentase didapat dari J / R x 100% di mana:

J = Banyaknya jawaban responden tiap soal

R = Jumlah responden

1. Apakah dalam *game* yang dibangun terdapat edukasi mengetik?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	9	90 %
2	Setuju	1	10 %
3	Ragu-ragu	-	-
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

2. Apakah *game* yang dibangun dapat menyadarkan akan pentingnya mengetik?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	5	50 %
2	Setuju	3	30 %
3	Ragu-ragu	2	20 %
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

3. Apakah *game* yang dibangun pantas menjadi media belajar dan latihan mengetik?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	8	80 %
2	Setuju	2	20 %
3	Ragu-ragu	-	-
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

4. Apakah *game* yang dibangun menghibur?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	10	100 %
2	Setuju	-	-
3	Ragu-ragu	-	-
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

5. Apakah *game* yang dibangun dapat meningkatkan kemampuan mengetik?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	10	100 %
2	Setuju	-	-
3	Ragu-ragu	-	-
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

6. Apakah *game* yang dibangun dapat membuat *player* terbiasa dengan teks bahasa Indonesia?

No.	Keterangan	Responden	Persentase
1	Sangat setuju	7	70 %
2	Setuju	3	30 %
3	Ragu-ragu	-	-
4	Kurang setuju	-	-
5	Sangat tidak setuju	-	-

4.2.6. Kesimpulan Hasil Pengujian Beta

Berdasarkan hasil persentase kuesioner yang diberikan kepada *user*, program yang dibangun dinyatakan sudah mengandung edukasi mengetik, membuat sadar akan pentingnya mengetik, layak dijadikan media belajar dan latihan mengetik, dapat menghibur *user*, bisa meningkatkan kemampuan mengetik, dan membiasakan *player* berhadapan dengan teks bahasa Indonesia.