BAB 4
IMPLEMENTASI ALGORITMA DAN HASIL PENGUJIAN

4.1 Implementasi

Tahap implementasi dilakukan setelah proses analisis selesai dilakukan selanjutnya diimplementasikan ke dalam bahasa pemrograman. Setelah implementasi algoritma dilakukan maka dilakukan pengujian terhadap algoritma yang digunakan pada aplikasi yang dibuat dengan tujuan untuk mengetahui apakah algoritma dan aplikasi yang dibuat dapat berjalan sesuai dengan tujuan atau tidak.

4.1.1 Implementasi Algoritma

Pada implementasi algoritma menjelaskan implementasi dan melakukan pengujian algoritma logika fuzzy terhadap perilaku menyerang dengan *auto-attack* dan *route finding* pada *game* RTS serta menjelaskan skenario dari hasil pengujian algoritma logika fuzzy yang diterapkan pada *game* bertipe RTS.

4.1.1.1 Pengujian Algoritma dan Skenario Pertama

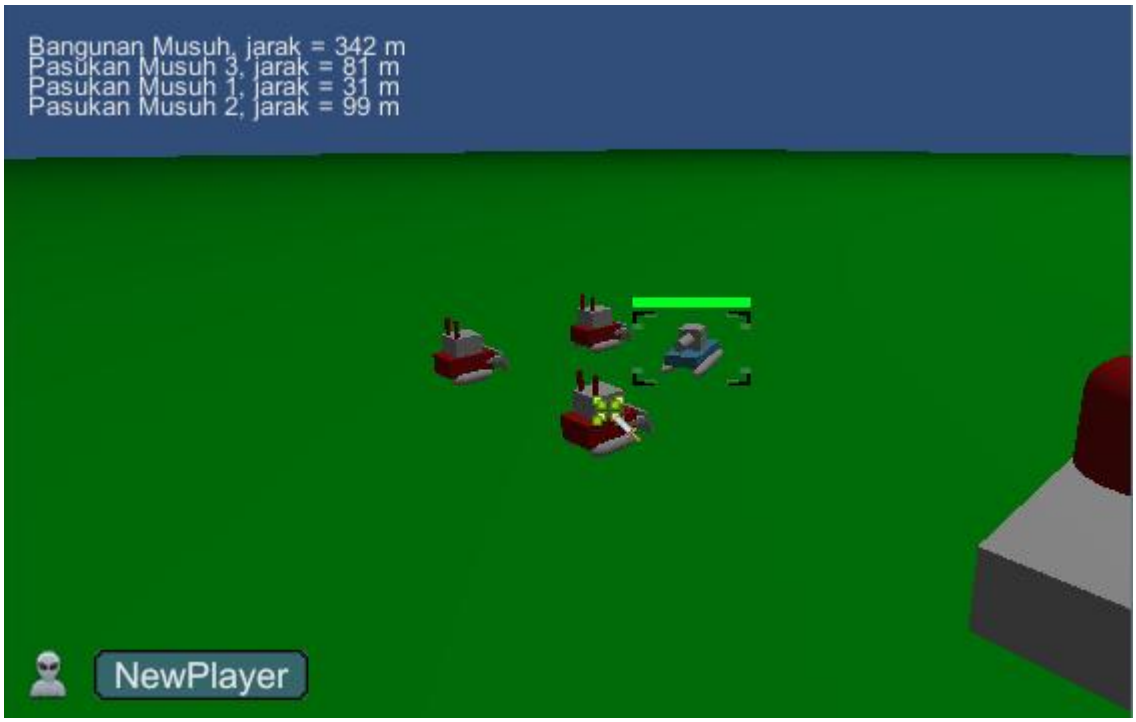
1. Pasukan *Player* Menyerang Tiga Pasukan Musuh

Pasukan musuh menyerang tiga pasukan musuh yang berada pada posisi tertentu dan mempunyai nilai masukan parameter yang berbeda. Dimana pasukan *player* merupakan pasukan yang berwarna biru sedangkan untuk pasukan musuh yang berwarna merah. Berikut nilai parameter dari pasukan musuh pada pengujian dan skenario pertama pada tabel 4.1.

Tabel 4.1 Parameter Pasukan Musuh pada Skenario Pertama

No	Nama Pasukan Musuh	Nilai Masukan Tiap Parameter Pasukan Musuh
1	Pasukan Musuh 1	*Range* (Jarak) : 31 *Life* (Nyawa) : 45 *Power* (Kekuatan) : 25

2	Pasukan Musuh 2	*Range* (Jarak) : 99 *Life* (Nyawa) : 100 *Power* (Kekuatan) : 70
3	Pasukan Musuh 3	*Range* (Jarak) : 81 *Life* (Nyawa) : 30 *Power* (Kekuatan) : 42



Gambar 4.1 Posisi dan Jarak Pasukan Musuh Skenario Pertama

2. Skenario Pengujian

Pemain memilih pasukan *player*, kemudian pemain memilih pasukan musuh 2 yang berada di sebelah kiri bangunan musuh untuk diserang oleh pemain..

***Output* logika fuzzy :**

1. jarak (Dekat, 31) = 0.95, nyawa (Lemah) = 0.75, power (Lemah) = 1, Min (0.95, 0.75, 1) = 0.75, Prioritas ke-1, Z = 23,25

2. jarak (Jauh, 99) = 1, nyawa (Lemah) = 1, power (Lemah) = 0.9, Min(0.95, 1, 0.9) = 0.9, Prioritas ke-1, Z = 41.9
3. Z = 55.1475 / 1.65 = 33.42

Maka dilihat dari *output* dari logika fuzzy yang menjadi prioritas serangan adalah pasukan musuh 1 dengan jarak (Dekat, 31) = 0.95, nyawa (Lemah) = 0.75, power (Lemah) = 1, Min (0.95, 0.75, 1) = 0.75, Prioritas ke-1, dengan nilai Z = 23.25 dan hasil dari perhitungan defuzzikasi 33.42.

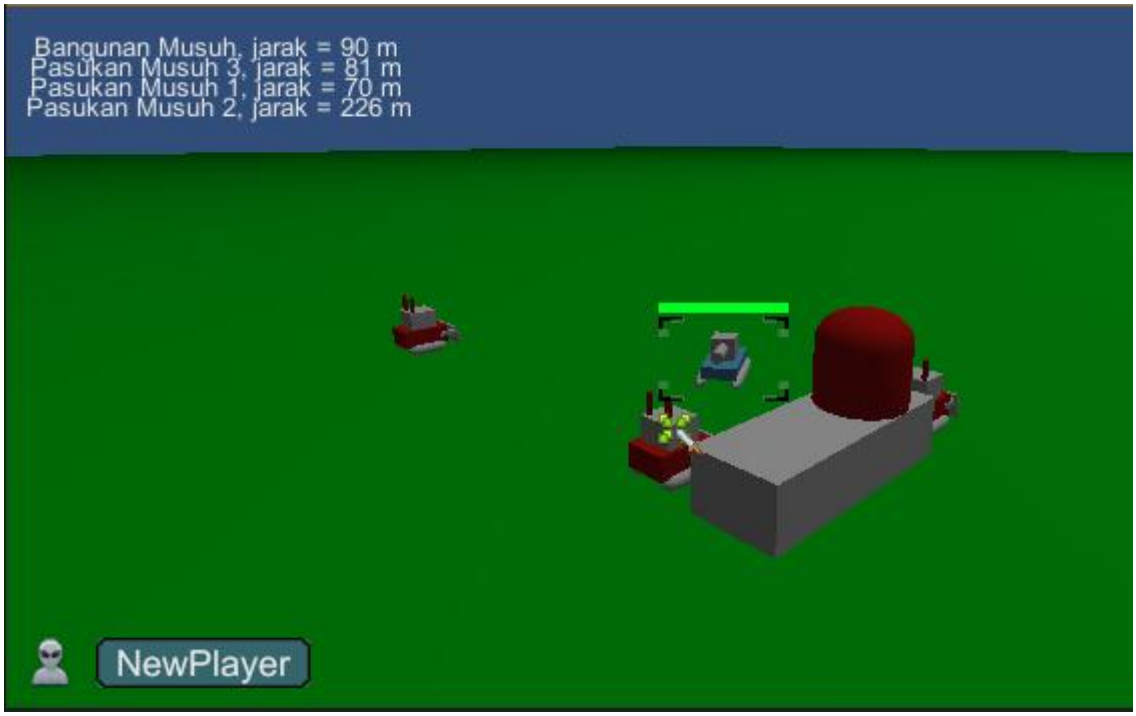
4.1.1.2 Pengujian Algoritma dan Skenario Kedua

1. Pasukan Menyerang Dua Pasukan dan Satu Bangunan Musuh

Pasukan musuh menyerang dua pasukan musuh dan satu bangunan musuh yang berada pada posisi tertentu dan mempunyai nilai parameter yang berbeda-beda. Dimana pasukan *player* merupakan pasukan yang berwarna biru sedangkan untuk pasukan musuh yang berwarna merah dan bangunan musuh yang memiliki puncak bangunan berwarna merah. Berikut nilai parameter dari pasukan musuh dan bangunan musuh pada pengujian dan skenario kedua pada tabel 4.2.

Tabel 4.2 Parameter Musuh pada Skenario Kedua

No	Posisi Musuh	Nilai Masukan Tiap Parameter Pasukan Musuh
1	Pasukan Musuh 3	*Range* (Jarak) : 90 *Life* (Nyawa) : 100 *Power* (Kekuatan) : 70
2	Bangunan Musuh	*Range* (Jarak) : 90 *Life* (Nyawa) : 85 *Power* (Kekuatan) : 85
3	Pasukan Musuh 1	*Range* (Jarak) : 70 *Life* (Nyawa) : 45 *Power* (Kekuatan) : 25



Gambar 4.2 Posisi dan Jarak Pasukan Musuh Skenario Kedua

2. Skenario pengujian

Pemain memilih pasukan *player*, kemudian pemain memilih pasukan musuh yang berada di sebelah kiri bangunan musuh untuk diserang oleh pemain. Kemudian pasukan *player* menyerang pasukan musuh 3 yang berada di sebelah kiri bangunan musuh.

***Output* logika fuzzy :**

1. jarak (Jauh, 90) = 0, nyawa (Kuat) = 1, power (Kuat) = 1, Min(0, 1, 1) = 0, Prioritas Ke - 2, Z = 45.5
2. jarak (Jauh, 70) = 1, nyawa (Lemah) = 0.75, power (Lemah) = 1, Min(1, 0.75, 1) = 0.75, Prioritas Ke - 1, Z = 43.25
3. Z = 55.1875 / 0.75 = 44.15

Maka dilihat dari *output* dari logika fuzzy yang menjadi prioritas serangan adalah pasukan musuh 1 dengan jarak (Jauh, 70) = 1, nyawa (Lemah) = 0.75,

power (Lemah) = 1, Min(1, 0.75, 1) = 0.75, Prioritas Ke - 1, Z = 44.15, dengan nilai Z = 44 dan hasil dari perhitungan defuzzikasi 44.15.

4.2 Pengujian *White-Box*

Pengujian *white-box* merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan dengan menggunakan stuktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Tujuan dari pengujian *white-box* untuk mengetahui cara kerja suatu aplikasi atau perangkat lunak secara internal dan untuk menjamin operasi-operasi internal sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur kontrol dari prosedur yang dirancang. Berikut adalah pengujian *white-box* yang dilakukan pada algoritma logika fuzzy :

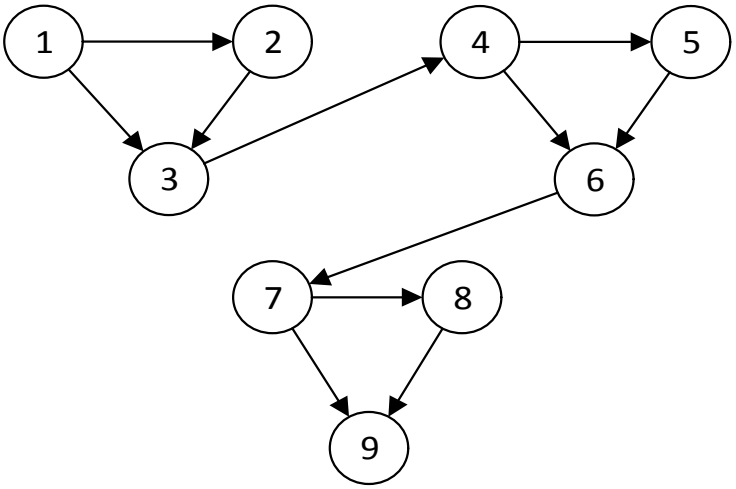
a. Pengujian *white-box* terhadap pembentukan himpunan fuzzy

Berikut adalah *pseudocode* untuk pembentukan himpunan fuzzy pada gambar 4.3.

```
1  Public float [] jarak_dekat = masukan array [];
2  Public float [] jarak_jauh = masukan array [];
3  Simpan masukan array jarak_dekat dan jarak_jauh
4  Public float [] nyawa_lemah = masukan array [];
5  Public float [] nyawa_kuat = masukan array [];
6  Simpan masukan array nyawa_lemah dan nyawa_kuat
7  Public float [] power_lemah = masukan array [];
8  Public float [] power_kuat = masukan array [];
9  Simpan masukan array power_lemah dan power_kuat
```

Gambar 4.3 *Pseudocode* Pembentukan Himpunan Fuzzy

Berdasarkan *pseudocode* pembentukan himpunan fuzzy maka diagram alirnya pada gambar 4.4.



Gambar 4.4 Diagram Alir *Pseudocode* Pembentukan Himpunan Fuzzy

Berdasarkan diagram alir *pseudocode* pembentukan himpunan fuzzy maka dapat dihitung *cyclometic complexity*, yaitu :

1. Flowgraph mempunyai 3 region.
2. V(G) = 11 edge – 9 node + 2 = 4
3. V(G) = 3 predicate node + 1 = 4

Dari hasil perhitungan *cyclometic complexity* terdapat 3 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4

Path 2 : 4 – 5 – 6 – 7

Path 3 : 6 – 7 – 8 – 9

b. Pengujian *white-box* terhadap fungsi implikasi parameter *Range* (Jarak)

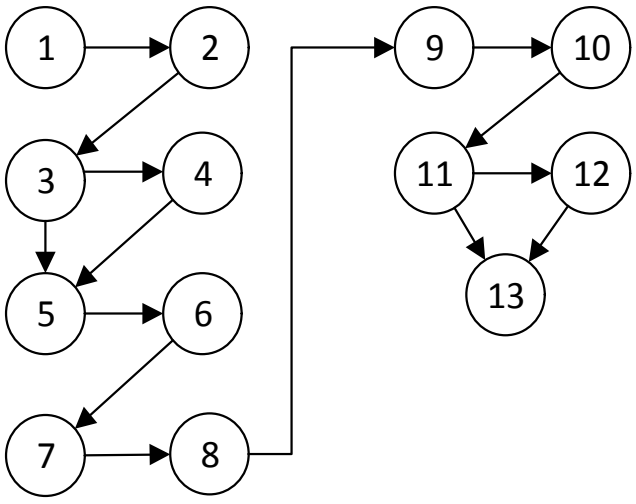
Berikut adalah *pseudocode* terhadap fungsi implikasi parameter *Range* (Jarak) pada gambar 4.5.

1	void fuzzy _jarak (float jarak)
2	jarak2 = jarak;
3	if (jarak2<=jarak_dekat [1]) and jarak>=jarak_dekat [0]) {
4	hasil_jarak_dekat=1;
5	hasil_jarak_jauh=0;

6	}endif if (jarak2>jarak_dekat[1]) and jarak2<=jarak_dekat [2]) {
7	hasil_jarak_dekat = - (jarak2-jarak_dekat [2]) / (jarak_dekat [2] –
8	jarak_dekat [1]; } endif
9	if (jarak2>jarak_jauh [0]) and jarak2<jarak_jauh [1]) {
	hasil_jarak_jauh = ((jarak2-jarak_jauh [0]) / (jarak_jauh [1] –
10	jarak_jauh [0]); } endif
11	if (jarak2>=jarak_jauh [1] and jarak2<=jarak_jauh[2]) {
	hasil jarak_dekat = 0;
12	hasil jarak_jauh = 1;
13	} endif

Gambar 4.5 *Pseudocode* Terhadap Fungsi Implikasi Parameter *Range* (Jarak)

Berdasarkan *pseudocode* terhadap fungsi implikasi parameter *Range* (Jarak) diatas maka diagram alirnya pada gambar 4.46.



Gambar 4.6 Diagram Alir *Pseudocode* Fungsi Implikasi Parameter *Range* (Jarak)

Berdasarkan diagram alir *pseudocode* fungsi implikasi terhadap parameter *Range* (Jarak) maka dapat dihitung *cyclometic complexity*, yaitu :

1. Flowgraph mempunyai 2 region.

2. V(G) = 14 edge – 13 node + 2 = 3
3. V(G) = 2 predicate node + 1 = 3

Dari hasil perhitungan *cyclometic complexity* terdapat 6 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4

Path 2 : 1 – 2 – 3 – 5

Path 3 : 3 – 4 – 6 – 7 – 8

Path 4 : 3 – 5 – 6 – 7 – 8

Path 5 : 9 – 10 – 11 – 13

Path 6 : 9 – 10 – 12 – 13

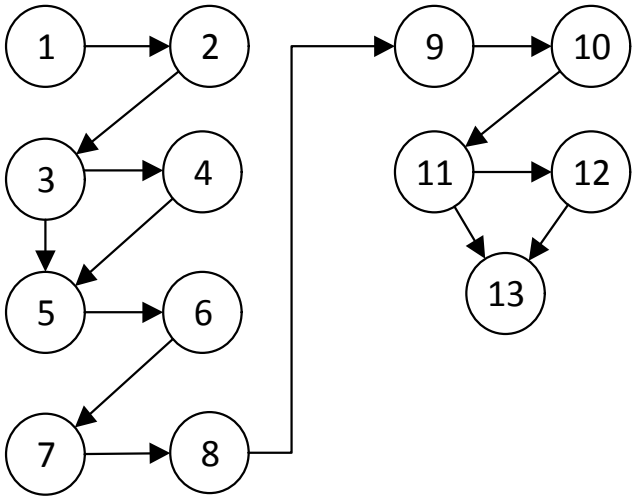
c. Pengujian *white-box* terhadap fungsi implikasi parameter *Life* (Nyawa)

Berikut adalah *pseudocode* terhadap fungsi implikasi parameter *Life* (Nyawa) pada gambar 4.7.

1	void fuzzy _nyawa (int nyawa)
2	nyawa2 = nyawa;
3	if (nyawa2<=nyawa_lemah [1]) and nyawa>=nyawa_lemah [0]) {
4	hasil_nyawa_lemah=1;
5	hasil_nyawa_kuat=0;
6	}endif if (nyawa2>nyawa_lemah[1]) and nyawa2<=nyawa_lemah
7	[2]) {
8	hasil_nyawa_lemah = - (nyawa2-nyawa_lemah [2]) / (nyawa_lemah [2] – nyawa_lemah [1]; } endif
9	if (nyawa2>nyawa_kuat [0]) and nyawa2<nyawa_kuat [1]) { hasil_nyawa_kuat = ((nyawa2-nyawa_kuat [0]) / (nyawa_kuat [1] –
10	nyawa_kuat [0]); } endif
11	if (nyawa2>=nyawa_kuat [1] and nyawa2<=nyawa_kuat [2]) { hasil nyawa_lemah = 0;
12	hasil nyawa_lemah = 1;
13	} endif

Gambar 4.7 *Pseudocode* Terhadap Fungsi Implikasi Parameter *Life* (Nyawa)

Berdasarkan *pseudocode* terhadap fungsi implikasi parameter *Life* (Nyawa) diatas maka diagram alirnya pada gambar 4.8.



Gambar 4.8 Diagram Alir *Pseudocode* Fungsi Implikasi Parameter *Life* (Nyawa)

Berdasarkan diagram alir *pseudocode* fungsi implikasi terhadap parameter *Life* (Nyawa) maka dapat dihitung *cyclometic complexity*, yaitu :

1. Flowgraph mempunyai 2 region.
2. V(G) = 14 edge – 13 node + 2 = 3
3. V(G) = 2 predicate node + 1 = 3

Dari hasil perhitungan *cyclometic complexity* terdapat 6 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4

Path 2 : 1 – 2 – 3 – 5

Path 3 : 3 – 4 – 6 – 7 – 8

Path 4 : 3 – 5 – 6 – 7 – 8

Path 5 : 9 – 10 – 11 – 13

Path 6 : 9 – 10 – 12 – 13

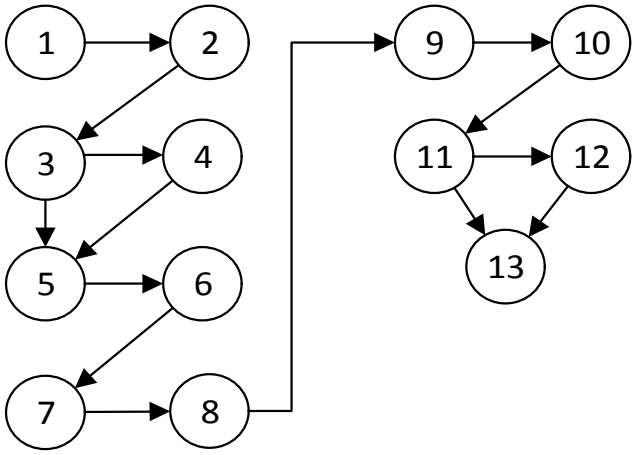
d. Pengujian *white-box* terhadap fungsi implikasi parameter *Power* (Kekuatan)

Berikut adalah *pseudocode* terhadap fungsi implikasi parameter *Power* (Kekuatan) pada gambar 4.9.

```
1   void fuzzy _power (int power)
2   power2 = power;
3   if (power2<=power_lemah [1]) and power>=power_lemah [0] ) {
4   hasil_power_lemah=1;
5   hasil_power_kuat=0;
6   }endif  if (power2>power_lemah[1]) and power2<=power_lemah
7   [2]){
8   hasil_power_lemah = - (power2-power_lemah [2]) / (power_lemah [2]
    – power_lemah [1]; } endif
9   if  (power2>power_kuat [0]) and power2<power_kuat [1]) {
    hasil_power_kuat = ((power2-power_kuat [0]) / (power_kuat [1] –
10  power_kuat [0]); } endif
11  if (power2>=power_kuat [1] and power2<=power_kuat[2]) {
    hasil power_lemah = 0;
12  hasil power_kuat = 1;
13  } endif
```

Gambar 4.9 *Pseudocode* Terhadap Fungsi Implikasi Parameter *Power* (Kekuatan)

Berdasarkan *pseudocode* terhadap fungsi implikasi parameter *Power* (Kekuatan) diatas maka diagram alirnya pada gambar 4.10.



Gambar 4.10 Diagram Alir *Pseudocode* Fungsi Implikasi Parameter *Power* (Kekuatan)

Berdasarkan diagram alir *pseudocode* fungsi implikasi terhadap parameter *Power* (Kekuatan) maka dapat dihitung *cyclometic complexity*, yaitu :

1. Flowgraph mempunyai 2 region.
2. V(G) = 14 edge – 15 node + 2 = 6
3. V(G) = 5 predicate node + 1 = 6

Dari hasil perhitungan *cyclometic complexity* terdapat 6 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4

Path 2 : 1 – 2 – 3 – 5

Path 3 : 3 – 4 – 6 – 7 – 8

Path 4 : 3 – 5 – 6 – 7 – 8

Path 5 : 9 – 10 – 11 – 13

Path 6 : 9 – 10 – 12 – 13

e. Pengujian *white-box* terhadap *rule* atau kaidah fuzzy

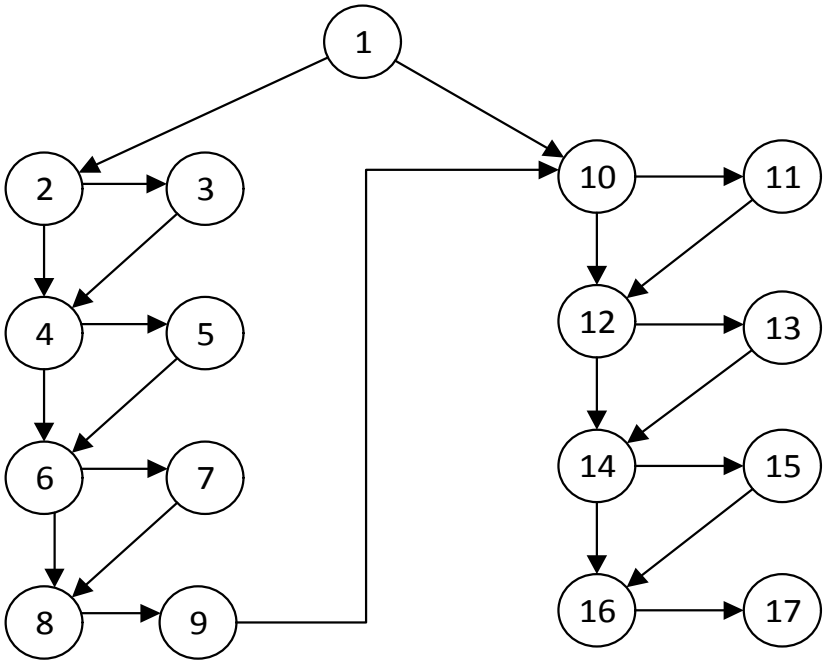
Berikut adalah *pseudocode* terhadap rule atau kaidah fuzzy pada gambar 4.11.

1	if (jarak = dekat and nyawa = lemah and power = lemah)
2	Target = Prioritas ke-1;

3	if (jarak = dekat and nyawa = lemah and power = kuat)
4	Target = Prioritas ke-2 ;
5	if (jarak = dekat and nyawa = kuat and power = lemah)
6	Target = Prioritas ke-1;
7	if (jarak = dekat and nyawa = kuat and power = kuat)
8	Target = Prioritas ke-2
9	if (jarak = jauh and nyawa = lemah and power = lemah)
10	Target = Prioritas ke-1;
11	if (jarak = jauh and nyawa = lemah and power = kuat)
12	Target = Prioritas ke-2;
13	if (jarak = jauh and nyawa = kuat and power = lemah)
14	Target = Prioritas ke-1;
15	if (jarak = jauh and nyawa = kuat and power = kuat)
16	Target = Prioritas ke-2;

Gambar 4.11 *Pseudocode* Terhadap *Rule* atau Kaidah Fuzzy

Berdasarkan *pseudocode* terhadap *rule* atau kaidah fuzzy diatas maka diagram alirnya pada gambar 4.12.



Gambar 4.12 Diagram Alir *Pseudocode Rule* atau Kaidah Fuzzy

Berdasarkan diagram alir *pseudocode* terhadap *rule* atau kaidah fuzzy maka dapat dihitung *cyclometic complexity*, yaitu :

1. Flowgraph mempunyai 2 region.
2. V(G) = 22 edge – 17 node + 2 = 6
3. V(G) = 2 predicate node + 1 = 3

Dari hasil perhitungan *cyclometic complexity* terdapat 6 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

Path 2 : 1 – 2 – 4 – 5 – 6 – 7 – 8 – 9

Path 3 : 1 – 2 – 4 – 6 – 8 – 9

Path 4 : 9 – 10 – 12 – 14 – 16 – 17

Path 5 : 9 – 10 – 12 – 13 – 14 – 15 – 16 – 17

Path 6 : 1 – 10 – 12 – 13 – 14 – 15 – 16 – 17

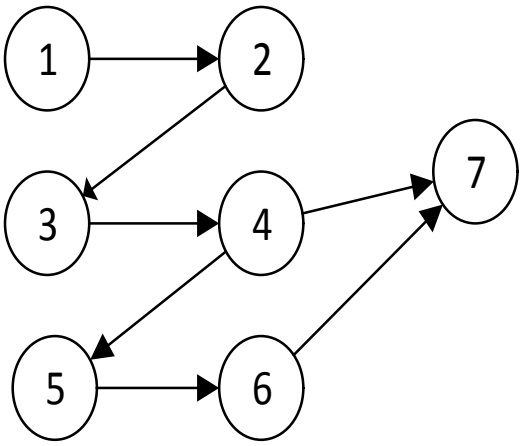
f. Pengujian white-box terhadap proses defuzzikasi

Berikut adalah pseudocode untuk proses defuzzikasi pada gambar 4.13

```
1  if(rjarak == "Dekat") {
2  z = jarak_dekat[1] - (min * 8);
3  ztemp = ztemp + (z * min);
4  } else {
5  z = jarak_jauh[1] - (min * 8);
6  ztemp = ztemp + (z * min);
7  }
```

Gambar 4.13 Pseudocode Untuk Proses Defuzzikasi

Berdasarkan *pseudocode* proses defuzzikasi diatas maka diagram alirnya pada gambar 4.14.



Gambar 4.14 Diagram Alir *Pseudocode* Proses Defuzzikasi

Berdasarkan diagram alir *pseudocode* terhadap proses defuzzikasi maka dapat dihitung *cyclometic complexity*, yaitu :

4. Flowgraph mempunyai 1 region.
5. V(G) = 7 edge – 7 node + 2 = 2
6. V(G) = 1 predicate node + 1 = 2

Dari hasil perhitungan *cyclometic complexity* terdapat 2 *independent path* yaitu :

Path 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7

Path 2 : 1 – 2 – 3 – 4 – 7