BAB IV

ANALISIS DAN PERANCANGAN APLIKASI

4.1. Penerapan Algoritma Minimax pada Permainan Reversi

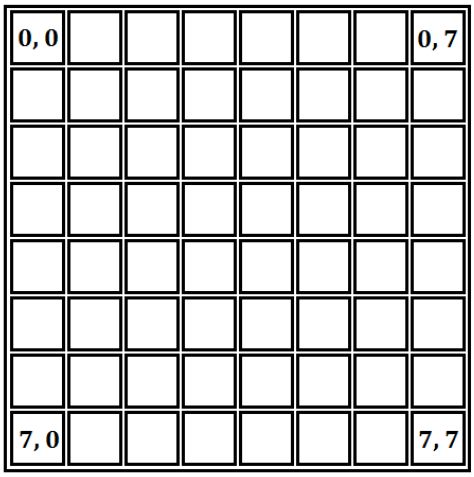
Penerapan algoritma minimax pada permainan reversi ini akan di rancang menggunakan *array* dua dimensi 8x8 yang merepresentasikan setiap kotak pada papan permainan reversi. Representasi papan main tersebut akan ditetapkan menggunakan sebuah variabel global bernama *board*[baris, kolom]. Pada keseluruhan *flow chart* pada sub-bagian 4.1.2 akan dapat dilihat penggunaan *board*[i, j], i dan j tersebut menandakan bahwa keseluruhan proses manipulasi *board* berada di dalam perulangan bersarang (*nested loop*) dengan *count* i dan j.

4.1.1. Basis Pengetahuan Algoritma Minimax pada Permainan Reversi

Basis Pengetahuan (*knowledge based*) pada *artificial intelligence* adalah sekumpulan informasi yang berupa fakta-fakta, teori-teori, dan perhitungan yang akan membantu untuk mengambil kesimpulan menemukan langkah terbaik dari setiap langkah yang diberikan.

Adapun *klowledge based* pada permainan reversi ini adalah :

1. Diketahui 4 posisi *corner* (sudut) pada papan direpresentasikan dengan *board* [0, 0], [0, 7], [7, 0], [7, 7]. Jumlah posisi *corner* ini akan dikali dengan poin 5 sebagai nilai evaluasi terhadap *current board*.



Gambar 4.1. Representasi posisi *corners* pada *array (board)*

2. Diketahui 24 posisi *c-squares* pada papan direpresentasikan dengan *board* [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 1], [7, 2], [7, 3], [7, 4], [7, 5], [7, 6], [1, 7], [2, 7], [3, 7], [4, 7], [5, 7], [6, 7].

	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	
1, 0							1, 7
2, 0							2, 7
3, 0							3, 7
4, 0							4, 7
5, 0							5, 7
6, 0							6, 7
	7, 1	7, 2	7, 3	7, 4	7, 5	7, 6	

Gambar 4.2. Representasi posisi *c-squares* pada *array (board)*

3. Diketahui 12 posisi *x-squares* pada papan direpresentasikan dengan *board* [1, 1], [2, 2], [3, 3], [4, 4], [5, 5], [6, 6], [1, 6], [2, 5], [3, 4], [4, 3], [5, 2], [6, 1].

	1, 1					1, 6	
		2, 2			2, 5		
			3, 3	3, 4			
			4, 3	4, 4			
		5, 2			5, 5		
	6, 1					6, 6	

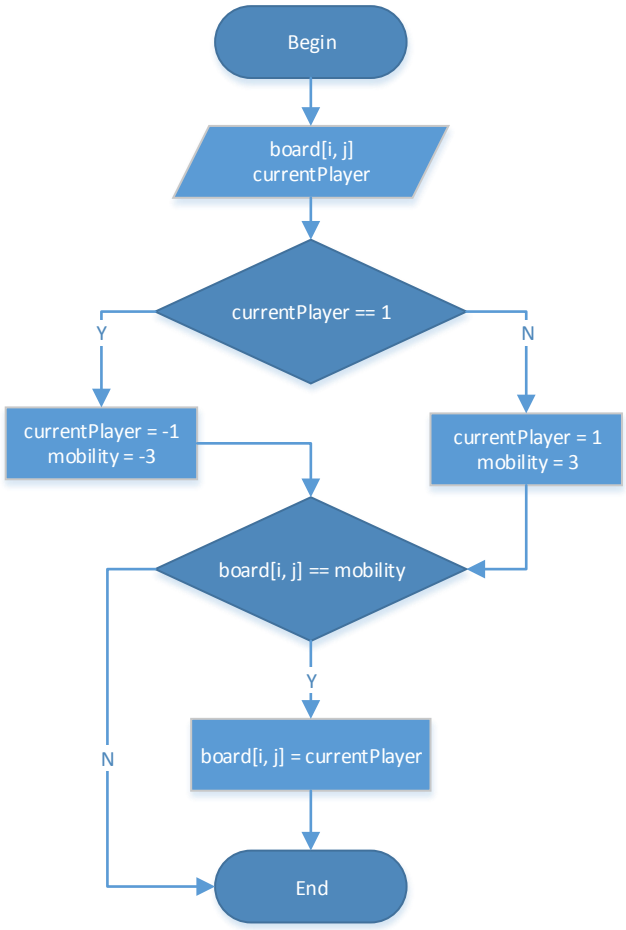
Gambar 4.3. Representasi posisi *x-squares* pada *array (board)*

4. Jika posisi-posisi *c-Square* dan *x-Square* ditempati bidak (koin) dan tersambung dengan posisi *corner* maka *c-Square* dan *x-Square* ini dapat dikatakan sebagai bidak stabil.
5. Ditetapkan bahwa representasi bidak hitam pada *board* dengan nilai 1, bidak putih dengan nilai -1, dan kosong dengan 0. Sedangkan untuk *mobility* bidak hitam dengan nilai 3, *mobility* bidak putih dengan -3.
6. Inisialisasi pada awal permainan yaitu *board* [3, 3] dan *board* [4, 4] bernilai -1 (bidak putih) sedangkan *board* [3, 4] dan *board* [4, 3] bernilai 1 (bidak hitam).

4.1.2. *Flow Chart* Algoritma pada Permainan Reversi

4.1.2.1. *Flow Chart* Algoritma Simpan Bidak

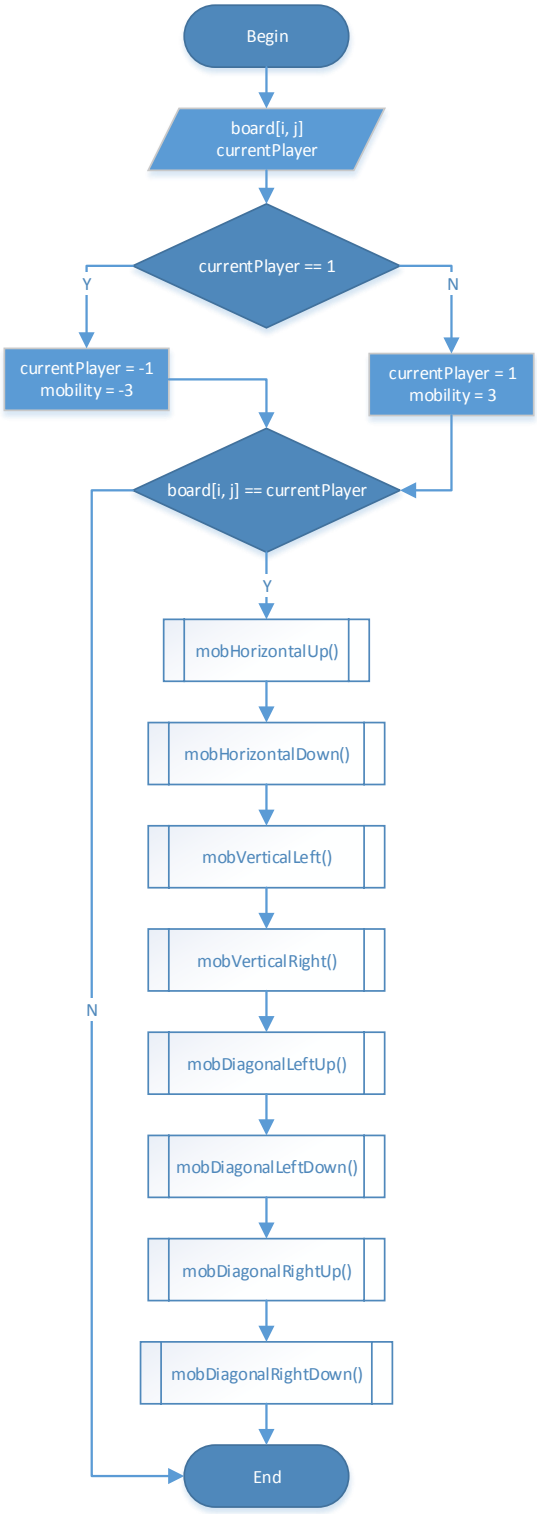
Algoritma Simpan Bidak merupakan algoritma untuk menentukan apakah pada papan akan disimpan Bidak (hitam atau putih) atau tidak. Simpan bidak ini ditentukan oleh papan (*board*[i, j]) yang menempati suatu *mobility*.



Gambar 4.4. *Flow Chart* Algoritma Simpan Bidak

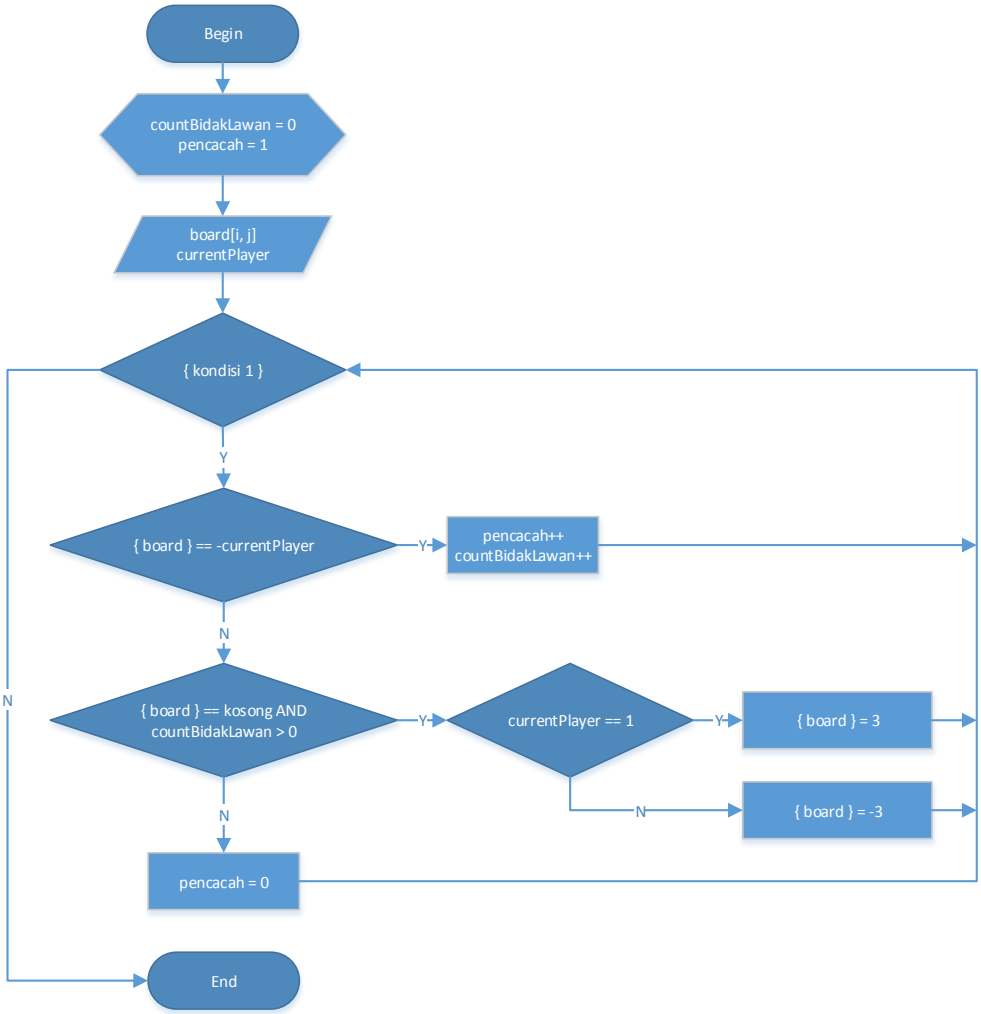
4.1.2.2. *Flow Chart* Algoritma Pemberian *Mobility* pada Setiap Langkah

Sebagaimana diketahui di awal bahwa *mobility* adalah tempat-tempat atau kotak-kotak yang dapat dijadikan langkah selanjutnya (penyimpanan bidak), dalam hal rancangan algoritmanya, pertama-tama penulis melakukan pencarian terhadap semua kotak atau tempat yang ditempati oleh bidak *current play*, kemudian dilakukan pencarian *mobility* secara horizontal ke atas, horizontal ke bawah, vertikal ke kiri, vertikal ke kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah.



Gambar 4.5. *Flow Chart* Algoritma Pemberian *Mobility* pada Setiap Langkah

Pada *flow chart* gambar 4.5 dapat dilihat bahwa ada beberapa prosedur yang mana prosedur tersebut memiliki tahapan proses yang hampir sama. Hanya dibedakan dari kondisi dan posisi pencarian ke berbagai arah dari posisi *current*, ada yang pencarian secara horizontal ke atas, horizontal ke bawah, vertikal ke kiri, vertikal ke kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah. Jika digambarkan dengan *flow chart* prosedur tersebut seperti berikut :



Gambar 4.6. *Flow Chart* Prosedur *mobility*

Pada *flow chart* gambar 4.6 tersebut yang membedakan kedelapan prosedur *mobility* pada gambar 4.5 adalah bagian { kondisi 1 } dan { board } saja. Penjelasannya sebagai berikut :

{ kondisi 1 } untuk :

a. Horizontal ke atas : *(pencacah != 0) AND (i >= 0) AND (i – pencacah >= 0)*
b. Horizontal ke bawah : *(pencacah != 0) AND (i <= 7) AND (i + pencacah <= 7)*
c. Vertikal ke kiri : *(pencacah != 0) AND (j >= 0) AND (j – pencacah >= 0)*
d. Vertikal ke kanan : *(pencacah != 0) AND (j <= 7) AND (j – pencacah <= 7)*
e. Diagonal kiri atas : *(pencacah != 0) AND (i >= 0) AND (j >= 0) AND (i – pencacah >= 0) AND (j – pencacah >= 0)*
f. Diagonal kiri bawah : *(pencacah != 0) AND (i <= 7) AND (j >= 0) AND (i + pencacah <= 7) AND (j – pencacah >= 0)*
g. Diagonal kanan atas : *(pencacah != 0) AND (i >= 0) AND (j <= 7) AND (i – pencacah >= 0) AND (j + pencacah <= 7)*
h. Diagonal kanan bawah : *(pencacah != 0) AND (i <= 7) AND (j <= 7) AND (i + pencacah <= 7) AND (j + pencacah <= 7)*

{ board } untuk :

a. Horizontal ke atas : *board[i - pencacah, j]*
b. Horizontal ke bawah : *board[i + pencacah, j]*

c. Vertikal ke kiri : *board[i, j - pencacah]*

d. Vertikal ke kanan : *board[i, j + pencacah]*

e. Diagonal kiri atas : *board[i - pencacah, j - pencacah]*

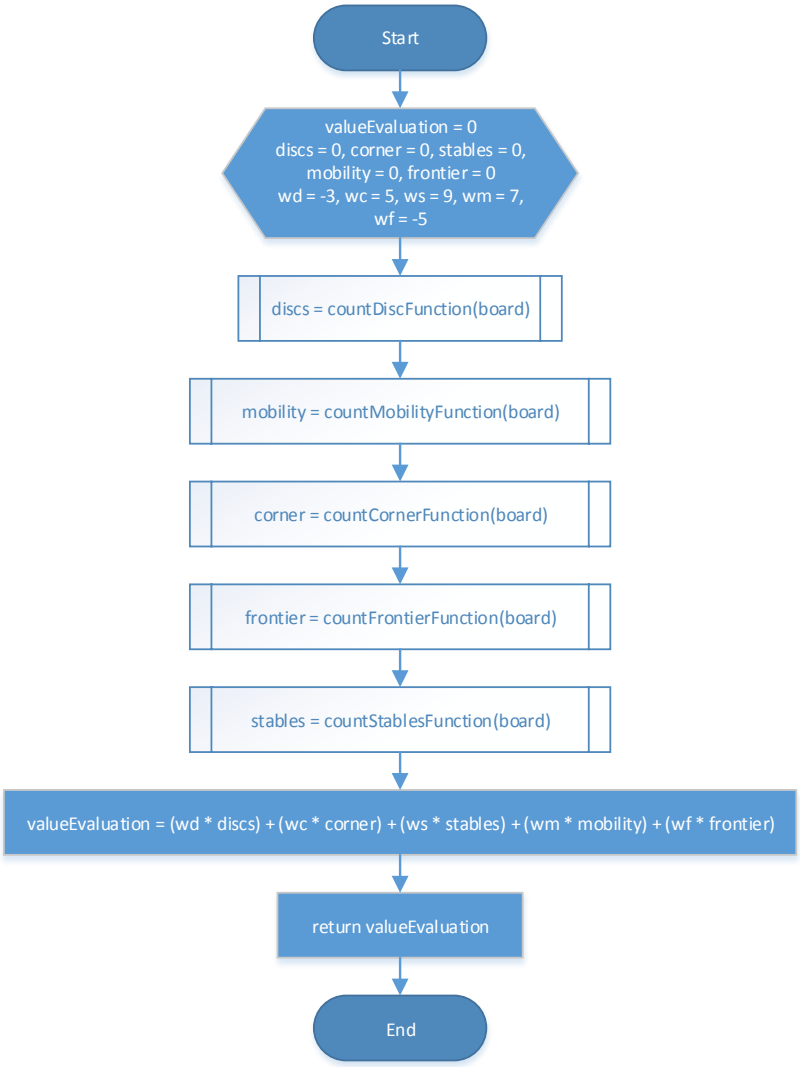
f. Diagonal kiri bawah : *board[i + pencacah, j - pencacah]*

g. Diagonal kanan atas : *board[i - pencacah, j + pencacah]*

h. Diagonal kanan bawah : *board[i + pencacah, j + pencacah]*

4.1.2.3. *Flow Chart* Algoritma Pemberian Nilai Evaluasi pada Setiap Langkah

Berikut ini merupakan *flow chart* algoritma untuk pemberian nilai evaluasi pada setiap langkah atau pada setiap kondisi papan yang terbentuk. Hal ini diperlukan karena komputer (AI) akan menemukan kondisi dari langkah terbaik dari nilai evaluasi yang dihasilkan. Algoritma ini merupakan sebuah *function* dengan parameter kondisi papan terakhir dan *function* ini akan mengembalikan sebuah nilai evaluasi bertipe *integer*.

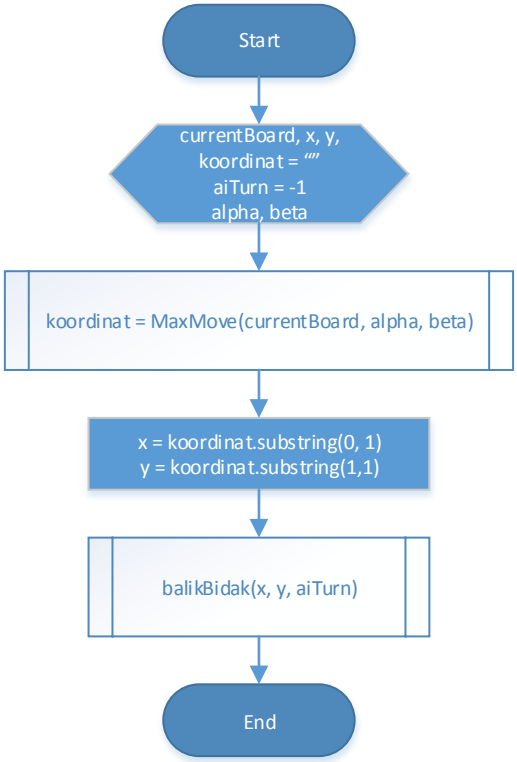


Gambar 4.7. *Flow Chart* Algoritma Pemberian Nilai Evaluasi

Pada Gambar 4.7 terdapat beberapa *function* untuk melakukan perhitungan nilai pada papan dengan menghitung jumlah bidak keseluruhan, jumlah *mobility* yang terbentuk, jumlah bidak pojok, dan jumlah bidak stabil yang kemudian akan dilakukan perhitungan dengan sebuah rumus yang telah ditetapkan sebelumnya.

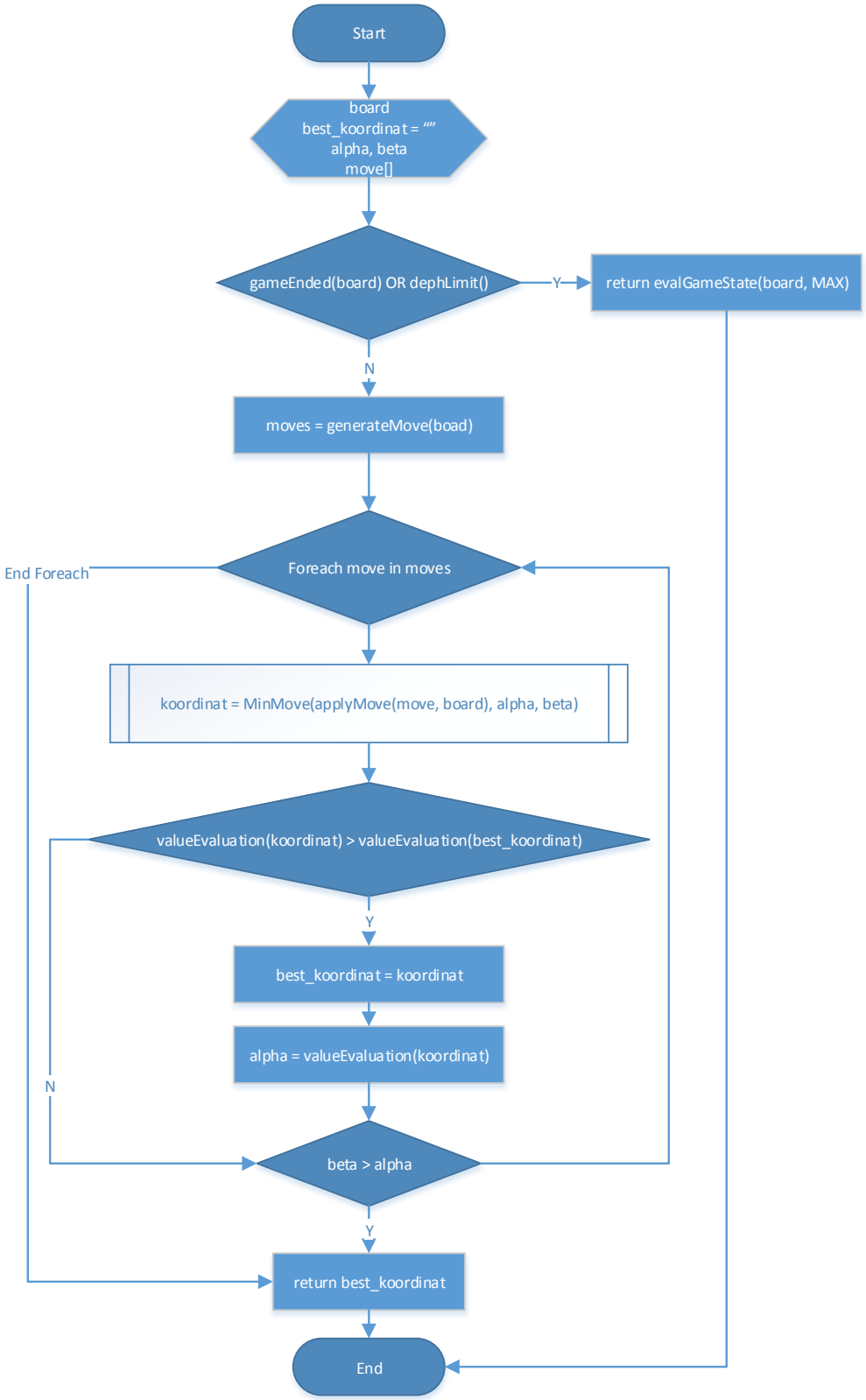
4.1.2.4. *Flow Chart* Algoritma Pencarian Langkah Terbaik

Pada *flow chart* algoritma pencarian langkah terbaik ini penulis akan menggunakan penerapan algoritma minimax sehingga terdapat tiga fungsi yang harus dibuat diantaranya fungsi **MinMax** itu sendiri, kemudian fungsi **MaxMove** dan **MinMove**. Ketiga fungsi ini mengembalikan nilai berupa suatu koordinat (*mobility*) yang terbaik dari beberapa *mobility* yang terbentuk berdasarkan posisi *current board*.



Gambar 4.8. *Flow Chart* Algoritma Pencarian Langkah Terbaik (MinMax)

Pada *flow chart* algoritma diatas terdapat pemanggilan terhadap suatu fungsi bernama **MaxMove** yang mana fungsi ini akan mengembalikan nilai berupa string. Berikut *flow chart* fungsi **MaxMove** dapat dilihat pada Gambar 4.9 :

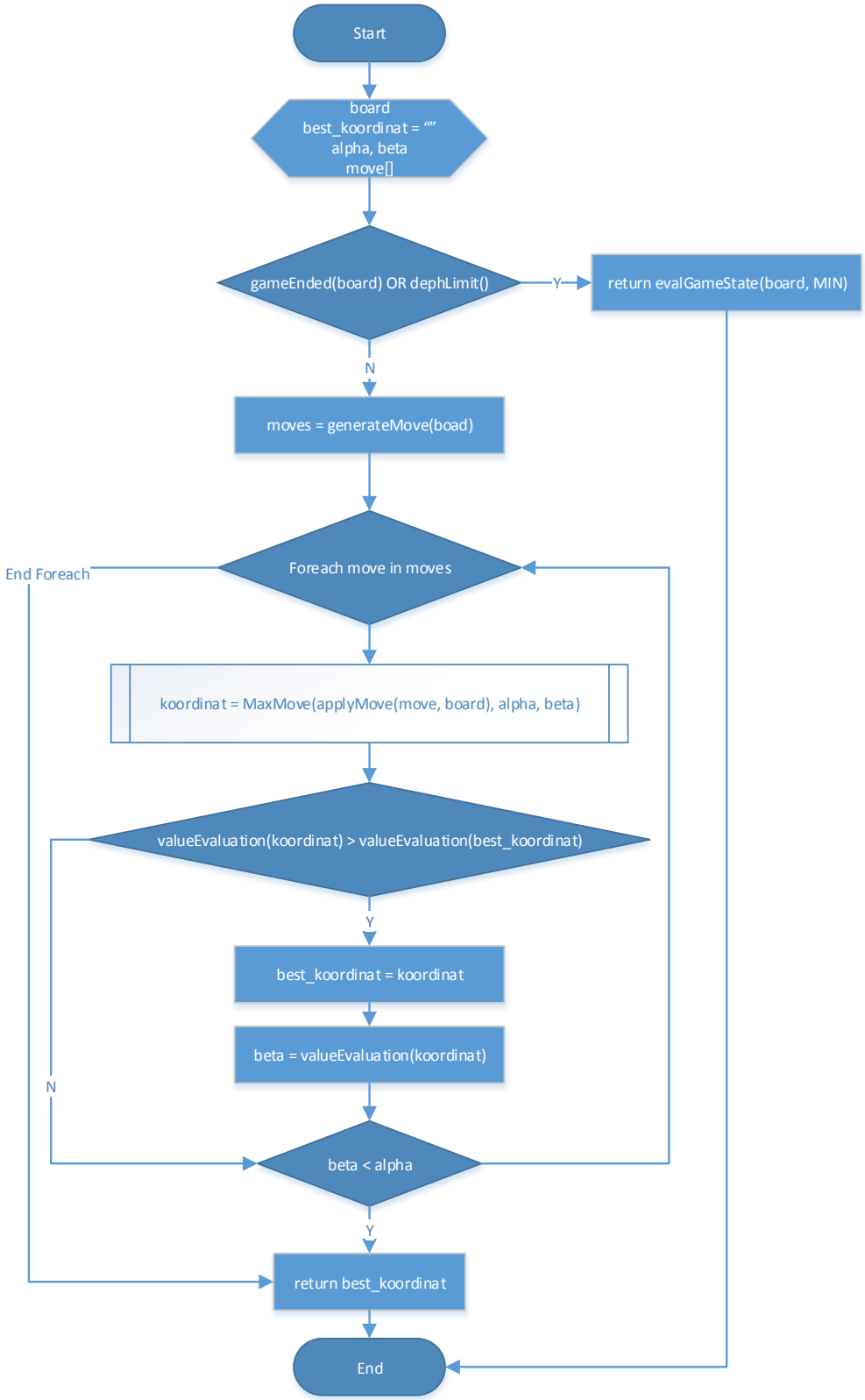


Gambar 4.9. *Flow Chart* Algoritma Pencarian Langkah Terbaik (MaxMove)

Secara garis besar *flow chart* algoritma fungsi **MaxMove** diatas yaitu pertama melakukan pemeriksaan terhadap langkah, apakah merupakan *leaf node* atau merupakan batas kedalaman yang telah ditentukan karena adanya *level* permainan (baik *beginner*, *intermediate*, atau *expert*). Kemudian jika bukan merupakan kedua hal tersebut maka akan dilakukan iterasi langkah berikutnya yang kemudian pada setiap langkah yang terbentuk akan dilakukan pemanggilan fungsi **MinMove** dikarenakan langkah **MinMove** tersebut merupakan representasi langkah *human player* dalam bermain dan harus dicari nilai seminimum mungkin. Sedangkan untuk **MaxMove** akan dicari nilai semaksimal mungkin karena merupakan representasi langkah *computer player*.

Pada fungsi **MinMove** akan dilakukan proses yang hampir sama dengan fungsi **MaxMove** yaitu pemeriksaan kondisi papan dan kedalaman kemudian akan dilakukan iterasi langkah berikutnya yang mana untuk setiap langkah yang terbentuk akan dilakukan pemanggilan kembali fungsi **MaxMove**, dan begitu seterusnya sampai menemukan kondisi akhir.

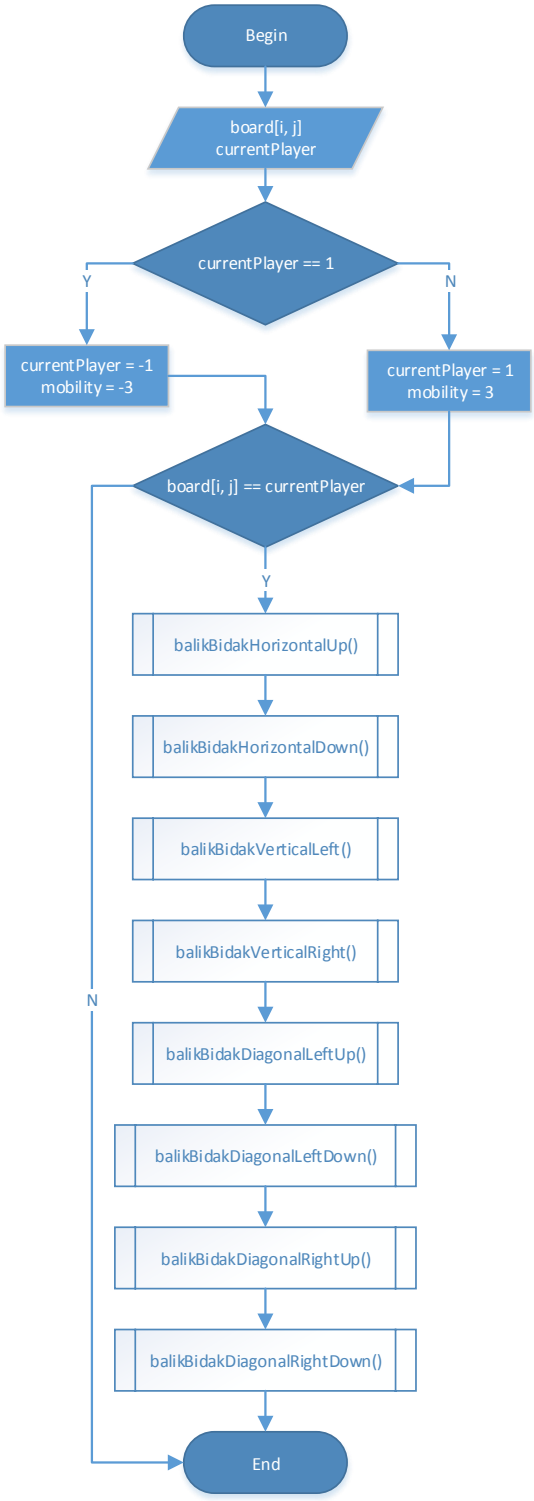
Berikut *flow chart* algoritma fungsi **MinMove** dapat dilihat pada Gambar 4.10 :



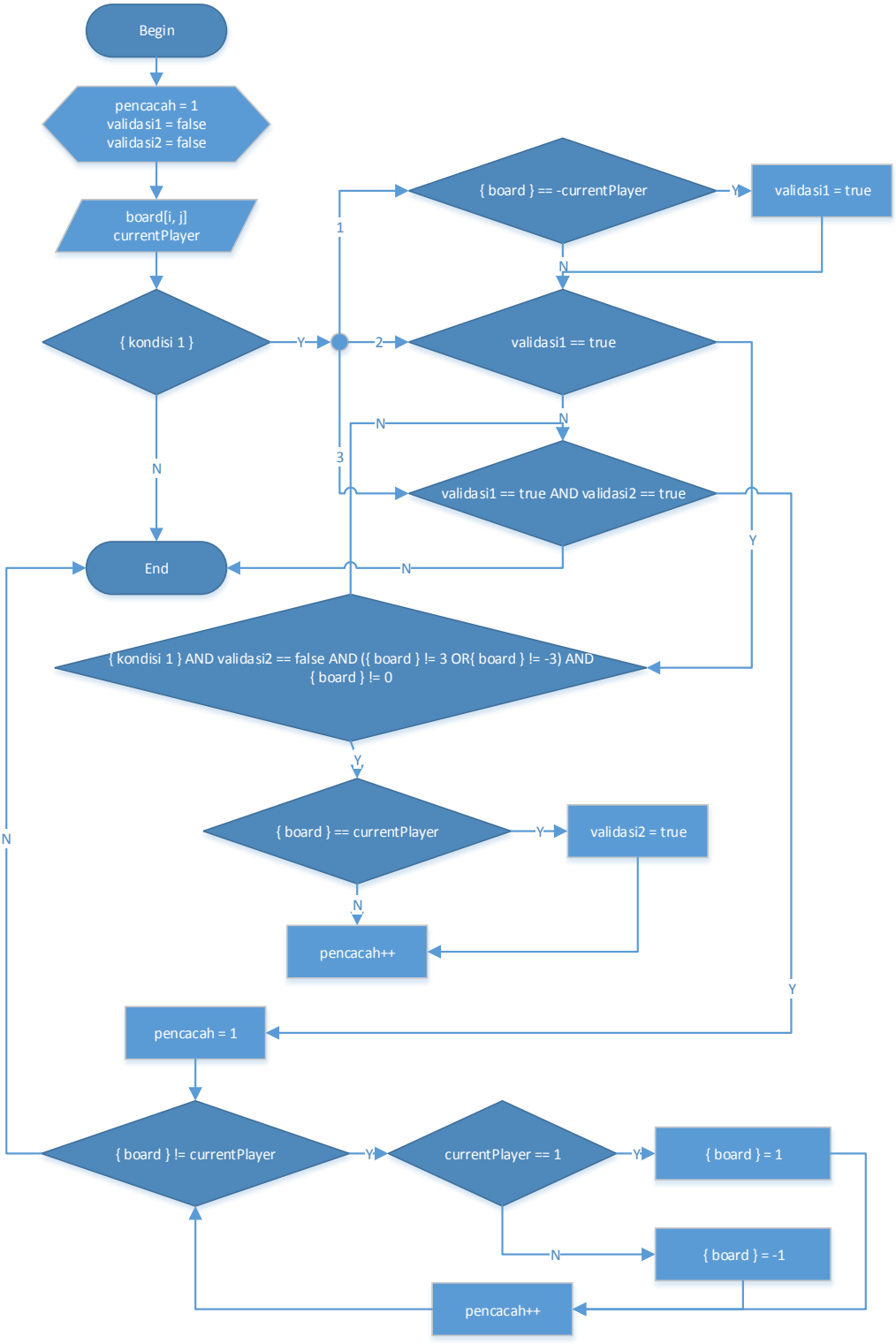
Gambar 4.10. *Flow Chart* Algoritma Pencarian Langkah Terbaik (MinMove)

4.1.2.5. *Flow Chart* Algoritma Balik Bidak Lawan

Algoritma balik bidak lawan merupakan algoritma untuk membalik (mengganti) semua bidak lawan yang terlewati dengan bidak *current play* baik ke arah horizontal atas, horizontal bawah, vertikal kiri, vertikal kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah. Inti dari algoritma ini yaitu adanya pemanggilan prosedur untuk membalik bidak secara horizontal, vertikal, dan diagonal ke berbagai arah menyeluruh dengan total terdapat delapan prosedur yaitu **horizontalUp()**, **horizontalDown()**, **verticalLeft()**, **verticalRight()**, **diagonalLeftUp()**, **diagonalLeftDown()**, **diagonalRightUp()**, dan **diagonalRightDown()**. Kedelapan prosedur tersebut pada intinya sama, yang membedakan hanya arah penelusurannya saja, untuk lebih jelas dapat dilihat pada Gambar 4.12 *Flow Chart* balik bidak dengan pemanggilan prosedurnya pada Gambar 4.11.



Gambar 4.11. *Flow Chart* Algoritma Balik Bidak Lawan



Gambar 4.12. *Flow Chart* Prosedur *balikBidak*

Pada flow chart gambar tersebut yang membedakan kedelapan prosedur *balikBidak* pada gambar adalah bagian { kondisi 1 } dan { board }. Penjelasannya sebagai berikut :

{ kondisi 1 } untuk :

a. Horizontal ke atas : *(i - pencacah) >= 0*
b. Horizontal ke bawah : *(i + pencacah) <= 7*
c. Vertikal ke kiri : *(j - pencacah) >= 0*
d. Vertikal ke kanan : *(j + pencacah) <= 7*
e. Diagonal kiri atas : *(i - pencacah) >= 0 AND (j - pencacah) >= 0*
f. Diagonal kiri bawah : *(i + pencacah) <= 7 AND (j - pencacah) >= 0*
g. Diagonal kanan atas : *(i - pencacah) >= 0 AND (j + pencacah) <= 7*
h. Diagonal kanan bawah : *(i + pencacah) <= 7 AND (j + pencacah) <= 7*

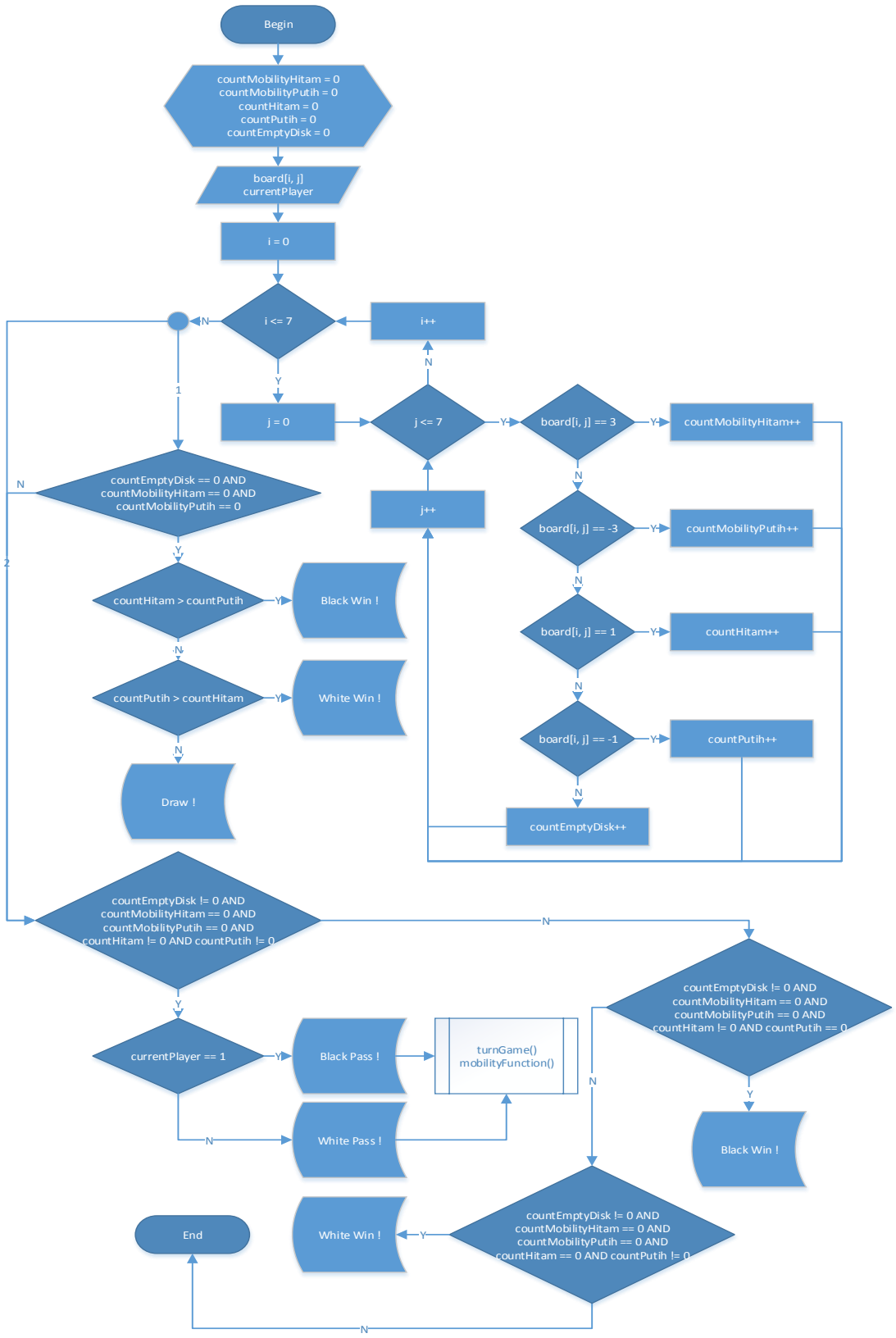
{ board } untuk :

a. Horizontal ke atas : *board[i - pencacah, j]*
b. Horizontal ke bawah : *board[i + pencacah, j]*
c. Vertikal ke kiri : *board[i, j - pencacah]*
d. Vertikal ke kanan : *board[i, j + pencacah]*
e. Diagonal kiri atas : *board[i - pencacah, j - pencacah]*
f. Diagonal kiri bawah : *board[i + pencacah, j - pencacah]*
g. Diagonal kanan atas : *board[i - pencacah, j + pencacah]*
h. Diagonal kanan bawah : *board[i + pencacah, j + pencacah]*

4.1.2.6. *Flow Chart* Algoritma Penentuan Pemenang

Adapun flow chart untuk algoritma penentuan pemenang yaitu algoritma untuk melakukan pemeriksaan terhadap bidak mana yang memenangkan permainan. Algoritma ini memeriksa setiap langkah yang dilakukan baik human player maupun computer player, algoritma ini juga yang mengatur jalannya permainan, misalnya apakah masih terdapat langkah yang mungkin untuk human player atau sebaliknya. Jika kedua pemain sudah tidak bisa melangkah dalam arti seluruh papan sudah dipenuhi koin atau sudah tidak ada lagi langkah yang mungkin untuk kedua pemain, maka permainan berakhir dan dilakukan perhitungan terhadap bidak atau koin yang dihasilkan.

Berikut flow chart algoritma penentuan pemenang dapat dilihat pada Gambar 4.13 :



Gambar 4.13. *Flow Chart* Algoritma Penentuan Pemenang

4.2. Rancangan Antarmuka Pengguna

Berikut ini merupakan rancangan antarmuka (UI) untuk aplikasi permainan reversi ini, diantaranya yaitu rancangan antarmuka masukan dan rancangan antarmuka keluaran.

4.2.1. Rancangan Antarmuka Masukan

Rancangan antarmuka atau tampilan aplikasi pada saat awal *running* terdapat beberapa *button* yang merupakan menu utama aplikasi.



Gambar 4.14. Tampilan Antarmuka Awal Aplikasi

Terdapat 4 *button* yang ada pada tampilan menu awal aplikasi ini, yaitu :

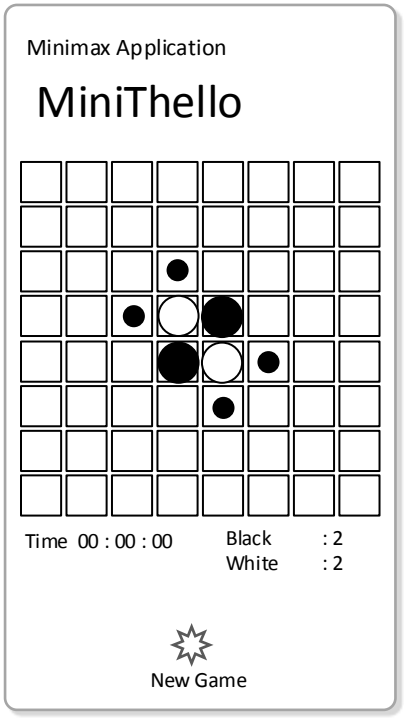
a) Play merupakan *button* untuk memulai permainan, jika mengalami perubahan status pada *button* tersebut, maka akan menampilkan tampilan antarmuka

papan permainan seperti pada gambar 4.15. Permainan dan *timing* akan dimulai ketika *human player* meletakan bidak pada papan yang terdapat *mobility* dan berakhir jika papan permainan telah ditentukan pemenangnya atau pada tengah-tengah permainan menekan *icon* New Game pada *application bar*.

b) About merupakan *button* untuk mengetahui tentang aplikasi permainan Reversi / Othello ini, untuk mengetahui versi dan *developer* aplikasi.

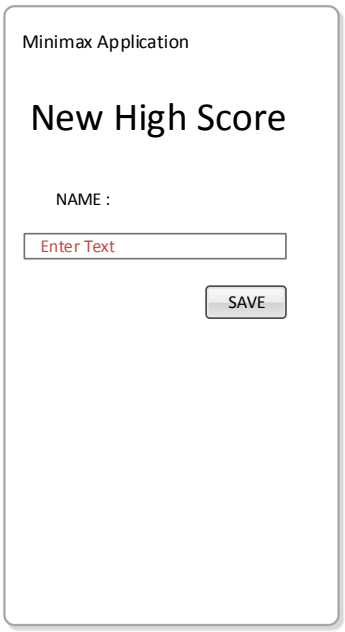
c) High Scores merupakan *button* untuk mengetahui top 15 besar peringkat waktu (*timing*) yang diperoleh.

d) Help merupakan *button* bantuan tentang penggunaan aplikasi permainan ini.



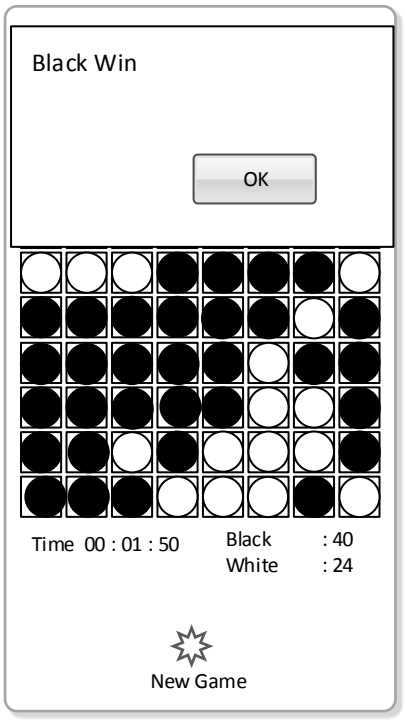
Gambar 4.15. Tampilan Antarmuka Papan Permainan

Pada gambar 4.15 diatas merupakan tampilan antarmuka papan permainan reversi, dapat dilihat ada komponen *label* yang merupakan *timing* dan *count* untuk bidak hitam dan putih. Permainan dimulai dan *timing* akan mulai berjalan jika *human player* meletakan bidak warna hitam pada *mobility* warna hitam (*mobility* direpresenasikan dengan bidak yg lebih kecil) kemudian berakhir jika pemenang sudah ditentukan atau dapat dengan menekan *icon* New Player pada *application bar*. Jika permainan berakhir dan dimenangkan oleh *human player* (*black win*), maka jika *timing* yang dihasilkan merupakan *timing* yang paling cepat dibandingkan dengan *timing players* sebelumnya, maka dapat menginputkan nama untuk masuk pada daftar top 15 *high scores*.



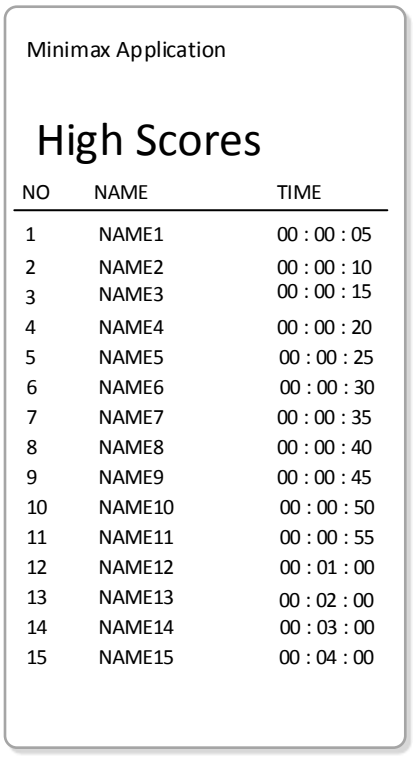
Gambar 4.16. Tampilan Antarmuka *Input New High Score*

4.2.2. Rancangan Antarmuka Keluaran



Gambar 4.17. Tampilan Antarmuka Pemberitahuan Pemenang

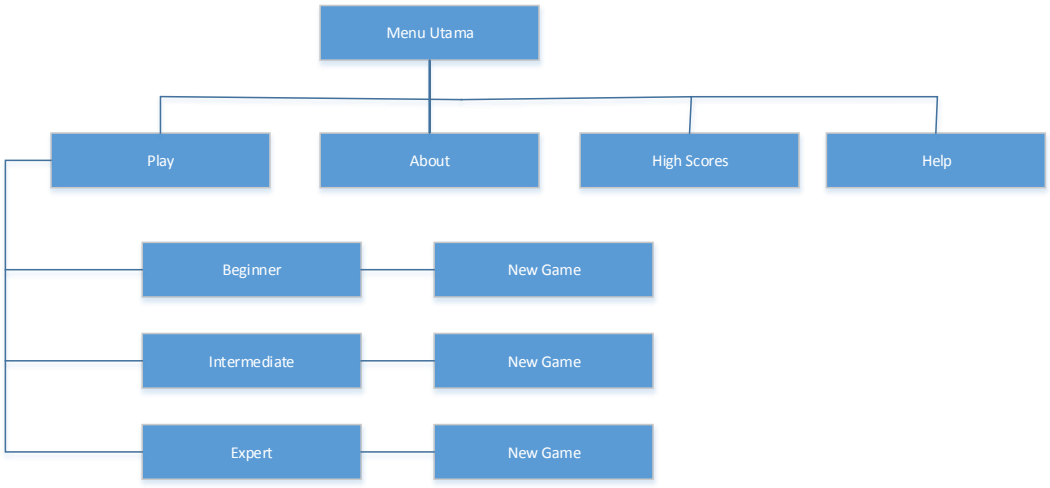
Pada gambar 4.17 merupakan tampilan antarmuka keluaran berupa pemberitahuan pemenang (menang, kalah, atau seri).



Gambar 4.18. Tampilan Antarmuka Informasi *High Scores*

4.3. Struktur Menu

Berikut ini adalah struktur menu pada aplikasi permaian Reversi :



Gambar 4.19. Struktur Menu Program Aplikasi

4.4. Perancangan Basis Data

Oleh karena pada aplikasi ini hanya akan ada satu tabel pada penyimpanan data, jadi penulis tidak akan menjelaskan konsep perancangan basis data secara normal yang mencakup normalisasi, ERD, dsb. Tetapi hanya dijelaskan mengenai satu tabel yang mempunyai spesifikasi sebagai berikut :

Nama File : Score

Media : Windows Phone Isolated Storage Settings

Isi : Data Score User

Primary Key : Id

Struktur File :

No	Nama Field	Jenis	Lebar	Desimal	Keterangan
1	Id	Numeric	4	0	Primary Key
2	Name	Character	20	0	
3	Time	Character	12	0	Timing
4	Second	Numeric	6	0	Time in Seconds (not show in application)