DESIGN DOCUMENT

OTHELLO PROGRAM

ARUM ESTRI PERDINASARI
arum.estri@gmail.com

DAFTAR ISI

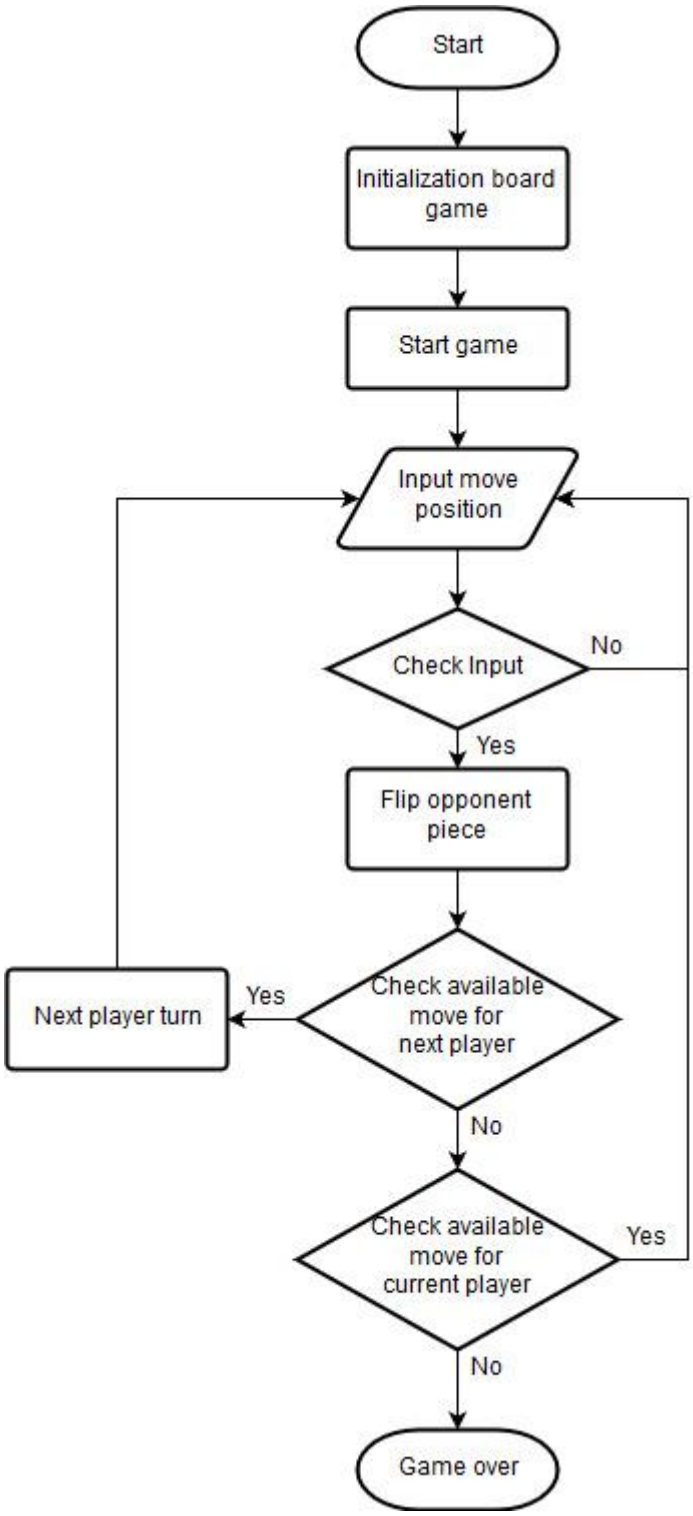
1. PENDAHULUAN

Othello adalah sebuah permainan antara dua pemain pada papan permainan yang terdiri dari 8x8 kotak. Kedua pemain tersebut masing-masing memainkan hitam dan putih. Tujuan dari permainan ini adalah memenuhi papan permainan dengan keeping koin yang dimiliki. Referensi dari peraturan dan permainan : http://www.hannu.se/games/othello/rules.htm.

Program ini bertujuan untuk menjalankan permainan Othello antara dua pemain user (*player vs player*). Kedua pemain akan menginput posisi untuk meletakkan keping pada papan permainan secara bergantian.

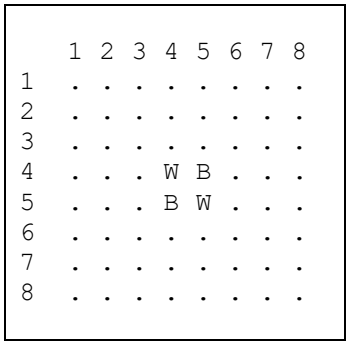
2. DESAIN ALGORITMA

Gambar dibawah ini adalah flow chart dari desain algoritma untuk membuat program Othello ini:



Dibawah ini merupakan penjelasan mengenai desain algoritma untuk menjalankan permainan Othello ini.

1) Inisialisasi board game game menggunakan array bertipe data char yang berukuran 89.



```
   1 2 3 4 5 6 7 8
1  . . . . . . . .
2  . . . . . . . .
3  . . . . . . . .
4  . . . W B . . .
5  . . . B W . . .
6  . . . . . . . .
7  . . . . . . . .
8  . . . . . . . .
```

   a. 0 – 9 : sebagai border atas dari papan permainan yang akan digunakan untuk informasi posisi kolom.

```
for (i = 0; i<=9; i++)
```

   b. 10 – 80 modulus 10 : sebagai border kiri dari papan permainan yang akan digunakan untuk informasi posisi baris.

```
if (i%10 == 0)
```

   c. Posisi akan ditentukan dengan : (baris*10)+kolom. sebagai contoh : baris ke 6, kolom ke 5 maka array[65].

```
if (i%10 >= 1 && i%10 <= 8)
```

   d. Game akan dimulai dengan putih pada posisi 44 dan 55, hitam pada posisi 45 dan 54 seperti pada gambaw diatas.

2) Game akan dimulai dengan pemain hitam terlebih dahulu.
3) Mulai game:
   a. Program akan meminta input dari pemain pada giliran tersebut untuk meletakkan piecenya pada papan permainan.
   b. Check masukan posisi dari pemain
      i. check apakah input posisi berada pada range array

```
if ((input >= 11) && (input <= 88) && (input %10 >= 1) &&
(input %10 <= 8))
```

      ii. check apakah posisi yang diinput oleh pemain kosong atau tidak
      iii. check apakah posisi yang diinput oleh pemain dapat membalikkan koin lawan atau tidak. dengan cara memeriksa seluruh arah :

```
int  direction [8]= {-11, -10, -9, -1, 1, 9, 10, 11}
      //posisi-posisi yang mengelilingi
while(i<=7)
{     check = move + 2* direction[i];
      if (board[move + direction[i]] == opponent)
      {
            while (board[check] == opponent) check = check +
direction[i];
            if (board[check] == player) i=i+1;
            else i=i+0;
      }
      else i=i+0; }
```

c. Apabila seluruh kondisi telah terpenuhi maka seluruh koin lawan yang terlompati diubah menjadi koin pemain.

```
board[move] = player;
for (i=0; i<=7; i++)
{     bracketer = player piece position to out flank.
      if (board[direction [i]]==opponent)
      {     flip = move + direction [i];
            do {  board[flip] = player;
                  flip = flip + direction [i];
            } while (flip != bracketer);  }}
```

d. Apabila seluruh kondisi tidak terpenuhi maka program akan meminta kembali inputan kepada pemain

e. Apabila giliran pemain sudah selesai, maka akan dilanjutkan pemain lawan.

f. Check apakah pemain selanjutnya masih memiliki posisi untuk meletakkan koinnya di papan permainan. Pemeriksaan dilakukan dari kolom ke 1 dan baris ke 1 hingga kolom ke 8 hingga baris ke 8 dengan menggunakan fungsi check posisi pada poin 3.b

   i. Apabila pemain selanjutnya masih dapat meletakkan koin, maka giliran akan diberikan kepada pemain selanjutnya.

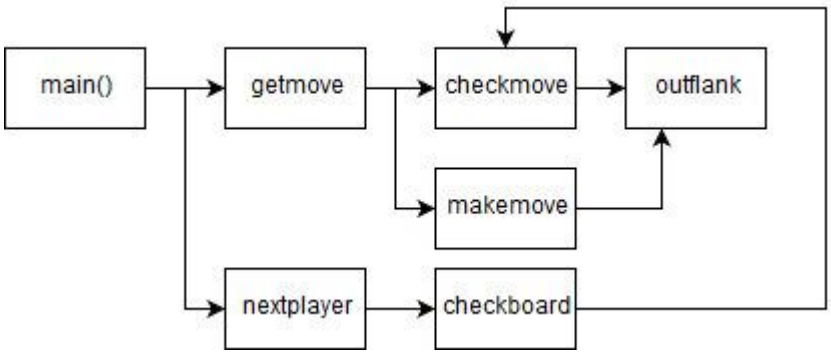
   ii. Apabila pemain selanjutnya tidak dapat meletakkan koin, maka giliran akan diberikan kepada pemain sebelumnya.

   iii. Apabila kedua pemain sudah tidak dapat meletakkan koin maka permainan selesai.

4) Pemenang akan ditentukan dengan menghitung jumlah keping yang dimiliki oleh hitam dan putih. Pemain dengan keping terbanyak yang akan menjadi pemenang. Apabila jumlah keping kedua pemain adalah sama maka hasil permainan adalah seri.

3. DESAIN MODUL

Berikut ini adalah desain diagram modul-modul.



Berikut ini adalah penjelasan mengenai desain modul-modul yang akan dibuat untuk menjalankan program permainan Othello.

1) main

Purpose	menjalankan permainan, menginisialisasi papan permainan, memulai permainan
Prototype	int main ()
Input	-
Output	-
Calls	○ getmove ○ nextplayer
Algorithm	○ Inisialisasi papan permainan awal ○ Inisialisasi pemain awal (hitam) ○ Menjalankan loop permainan ○ Menunjukkan hasil permainan ○ Menjalankan loop untuk mengulang permainan

2) getmove

Purpose	meminta dan memeriksa masukan dari user, membalik keping lawan
Prototype	void getmove (int player, int *board)
Input	pemain pada giliran tersebut, papan permainan terakhir
Output	-
Calls	○ checkmove ○ makemove
Algorithm	○ Meminta masukan pemain sampai masukan yang diterima valid ○ Memeriksa masukan pemain dengan memanggil fungsi checkmove ○ Membalik keping lawan dengan memanggil fungsi makemove

3) checkmove

Purpose	Memeriksa masukkan posisi pemain apakah valid atau tidak
Prototype	int checkmove (int move, int player, int *board)
Input	masukan posisi, pemain pada giliran tersebut, kondisi papan permainan terakhir

Output	flag hasil pemeriksaan. valid atau tidak
Calls	○ outflank
Algorithm	○ Memeriksa apakah posisi yang dimasukkan berada dalam range papan permainan ○ Memeriksa apakah posisi yang dimasukkan kosong atau tidak

4) outflank

Purpose	Memeriksa kearah mana saja keping lawan dapat dibalikkan dari posisi tertentu
Prototype	int outflank (int move, int player, int * board, int direction)
Input	masukan posisi, pemain pada giliran tersebut, kondisi papan permainan terakhir, arah yang akan diperiksa (yang mengelilingi posisi masukan)
Output	letak keping terujung yang melompati keping lawan.
Calls	-
Algorithm	○ Memeriksa apakah keping-keping lain yang terletak disekitar posisi masukan adalah keping lawan. ○ Apabila terdapat keping lawan, modul ini akan menjalankan loop untuk memeriksa apakah terdapat keping sendiri yang mengurung keping lawan.

5) makemove

Purpose	Membalik seluruh keping lawan yang terlompati.
Prototype	void makemove (int move, int player, int * board)
Input	masukan posisi, pemain pada giliran tersebut, kondisi papan permainan terakhir.
Output	-
Calls	○ outflank
Algorithm	○ Menjalankan loop sebanyak 7 kali yang menjalankan : ○ Menerima informasi posisi keping yang mengurung keping lawan dari modul outflank ○ Apabila terdapat posisi keping yang mengurung, maka modul akan menjalankan loop untuk merubah keping lawan menjadi keping pemain dengan cara merubah nilai pada array papan permainan sampai pada posisi keping yang mengurung.

6) nextplayer

Purpose	Menentukan pemain yang akan bermain digiliran selanjutnya
Prototype	int nextplayer (int * board, int previous)
Input	kondisi papan permainan terakhir, pemain yang sebelumnya.
Output	pemain yang bermain digiliran selanjutnya
Calls	○ checkboard
Algorithm	○ Memeriksa apakah pemain selanjutnya masih memiliki posisi yang boleh diisi ○ Apabila tidak, memeriksa apakah pemain yang sebelumnya masih memiliki posisi yang boleh diisi ○ Apabila tidak ada dari kedua pemain yang dapat mengisi papan permainan lagi, maka output dari program akan menghentikan loop yang menjalankan game pada modul main.

7) checkboard

Purpose	memeriksa apakah masih terdapat posisi yang boleh diisi oleh pemain
Prototype	int checkboard (int player, int * board)
Input	pemain, kondisi papan permainan terakhir
Output	flag apakah terdapat posisi yang masih bisa diisi atau tidak
Calls	○ checkmove
Algorithm	○ Menjalankan loop untuk memeriksa setiap posisi dari kolom 1 baris 1 hingga kolom 8 baris ke 8 apakah posisi tersebut terdapat posisi yang valid.

8) printboard

Purpose	menampilkan papan permainan
Prototype	void printBoard (int * board)
Input	kondisi papan permainan terakhir
Output	-
Calls	○ count
Algorithm	○ Membersihkan layar ○ Menampilkan informasi kolom ○ Menjalankan loop untuk menampilkan informasi baris dan isi dari setiap kotak. ○ Menampilkan jumlah keping masing-masing pemain

9) count

Purpose	menghitung jumlah keping pemain tertentu yang ada pada papan permainan
Prototype	int count (int player, int * board)
Input	pemain, papan permainan yang terakhir
Output	jumlah keping dimiliki pemain
Calls	-
Algorithm	○ Menjalankan loop dari 1 sampai 88 yang memerika apakah pada array papan permainan di posisi tersebut terdapat keping pemain. ○ Mengakumulasi seluruh keping pemain.

4. MAN-HOUR

Estimasi man-hour yang dihabiskan untuk mengerjakan program ini:

Pekerjaan	Man-hour
Desain implementasi	4
Pemrograman modul :	
1. main	1
2. getmove	1
3. checkmove	2
4. outflank	2
5. makemove	2
6. nextplayer	2
7. checkboard	2
8. modul accessoris (printboard, count, etc)	1
Testing and debungging	4
Trouble shooting	4
Total	**25**